EVIDEN

Identity and Access Management

DirX Identity

Troubleshooting Guide

Version 9.0.2, 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. Introduction 6
1.1. General Remarks 6
1.2. Avoiding Problems 6
1.3. Following Problem-Solving Guidelines 7
1.4. Opening Tickets 8
1.5. Getting Logging Message Information 8
2. Guidelines for Specific Components 10
2.1. Analyzing Global Aspects 10
2.2. Analyzing Installation and Configuration Problems 11
2.3. Analyzing Manager Problems 13
2.4. Analyzing Object Description Problems 14
2.5. Analyzing Web Center Problems 15
2.6. Analyzing Policy Problems 16
2.7. Analyzing Services Problems 17
2.8. Analyzing C++-based Server Problems 18
2.9. Analyzing Java-based Server Problems 19
2.10. Analyzing Java-based Workflow Problems 21
2.11. Analyzing Tcl-based Workflow Problems 23
2.12. Analyzing Request Workflow Problems 24
2.13. Using LDAP Session Tracking to Analyze Component Problems 26
3. Installation / Configuration Problems 29
3.1. Installation Problems 29
3.1.1. Determining which Agents are Installed 29
3.1.2. Master Setup and JRE on 32-bit Windows 29
3.1.3. Windows Installation Complete but Errors Occurred (Read-Only Files) 30
3.1.4. Installation Complete but Errors Occurred (Other Reasons) 31
3.2. Configuration Problems 31
3.2.1. Admin or Password Not OK 32
3.2.2. Bind Error during Schema Configuration 33
3.2.3. Creation of DirX Identity C++-based Service Fails 33
3.2.4. Long lasting configuration 33
3.2.5. Initial Configuration Loops 34
3.2.6. Configuration on Linux Virtualized on VMWare ESX 34
3.2.7. Schema Setup Problems after Upgrading DirX 34
3.3. Schema Installation Problems 35

3.3.1. Invalid Parameter Passed . . . 35
3.4. Uninstall and Unconfiguration Problems . . . 35
3.4.1. Some Libraries Cannot be Removed . . . 35
3.4.2. Objects Remain after Unconfiguration . . . 36
3.4.3. Unable to Remove Registry Key . . . 36
3.4.4. Uninstallation Hangs . . . 37
3.4.5. Uninstallation on 64-bit Windows . . . 37
3.5. Update and Upgrade Installation Problems . . . 37
3.5.1. Overwritten Agent Batch Files . . . 37
3.5.2. Overwritten SSL Certificate Stores during Upgrade . . . 38
3.5.3. Overwritten cert8.db File . . . 38
3.5.4. Windows Services Temporarily not Visible . . . 39
4. Problems with Provisioning . . . 40
4.1. Approval does not Work . . . 40
4.2. Changed Rules for Attributes . . . 40
4.3. Consistency Check Workflow is Aborted due to OutOfMemoryError . . . 41
4.4. Enabling Deleted Users . . . 41
4.5. Incorrect Approval Behavior . . . 42
4.6. Incorrect Privilege Resolution . . . 42
4.7. Invalid Query Reference . . . 42
4.8. Old Values in Membership Attributes . . . 43
4.9. Web Center or Java-based Server not Working Correctly . . . 44
4.10. Cleanup Rule for Imported SharePoint Group Members . . . 45
4.11. Inconsistent Group Membership . . . 45
5. Problems with Connectivity . . . 47
5.1. General Problems . . . 47
5.1.1. Imported Groups Cannot Be Assigned . . . 47
5.1.2. Insufficient Memory for Java Agents . . . 47
5.1.3. SSL Flag at Service Objects for LDAP Connections . . . 47
5.1.4. Workflow does not End . . . 48
5.2. Java-Based Workflow Problems . . . 48
5.2.1. Attribute is not Synchronized even though Specified in Mapping . . . 48
5.2.2. Starting Transport Workflows on Java-based Server . . . 49
5.2.3. SharePoint Connector Problems . . . 49
5.2.3.1. Content is not allowed in prolog.: 404 FILE NOT FOUND . . . 49
5.2.3.2. Org.xml.sax.SAXException: Premature end of file . . . 50
5.2.3.3. Java.net.SocketException: Connection reset . . . 50
5.3. Tcl-Based Workflow Problems . . . 51
5.3.1. ADS Agent . . . 51
5.3.1.1. Secure Authentication in Bind Profiles . . . 51
5.3.2. Exchange 5.5 Agent . . . 51
5.3.2.1. Stored Mailbox Resources . . . 51

5.3.3. Meta Controller . . . 52
5.3.3.1. Exit Code 4 . . . 52
5.3.3.2. GUID Generation Fails . . . 53
5.3.3.3. metacp and Additional LDAP Attribute Names . . . 53
5.3.3.4. metacp and Operational Attributes . . . 54
5.3.3.5. metacp Hangs . . . 54
5.3.3.6. Tcl Comment Problems . . . 55
5.3.3.7. Warnings are not Logged . . . 56
5.3.3.8. Workflows Run Too Slowly . . . 56
5.3.3.9. XML Parse Error . . . 57
5.3.4. Lotus Notes Agent . . . 57
5.3.4.1. Password Reading from File . . . 57
5.3.5. ODBC Agent . . . 57
5.3.5.1. Errors in Old ODBC Workflows . . . 57
5.3.5.2. MS Access: Too Few Parameters . . . 59
5.3.6. SAP R/3 HR Agent . . . 60
5.3.6.1. Extras / Copy Export . . . 60
5.3.7. SAP R/3 UM Agent . . . 60
5.3.7.1. Other Languages . . . 60
5.3.8. SAP EP UM Agent . . . 60
5.3.8.1. Setting User Passwords in SAP EP . . . 60
5.4. Windows Password Listener Problems . . . 61
5.4.1. User Data Empty Error Message in Event Viewer . . . 61
5.4.2. Service not Installed . . . 62
6. Problems with DirX Identity Manager . . . 63
6.1. General Issues . . . 63
6.1.1. Abbreviation Field Change Not Recognized . . . 63
6.1.2. Copied Report does not Work . . . 63
6.1.3. Instantiation Exception . . . 63
6.1.4. java.lang.NullPointerException . . . 64
6.1.5. java.lang.OutOfMemoryError . . . 65
6.1.6. Reset in Sub-Dialogs of Wizards . . . 65
6.1.7. Target System Report does not Work . . . 66
6.1.8. UTF-8 as Default Character Set . . . 66
6.1.9. Inconsistent Mapping in Tcl-based Workflows . . . 66
6.1.10. Wrong Order in Tree after Changing dxmDisplayName in Data View . . . 67
6.1.11. Displaying the LDAP Attribute Displayname . . . 67
6.2. Monitor View . . . 69
6.2.1. Duplicate Status Entries . . . 69
6.2.2. National Characters Are Not Displayed . . . 70
6.2.3. No HTML Viewer . . . 70
6.2.4. No File Viewer for UNIX . . . 70

6.2.5. No Statistics When Compression Mode is On . . . 71
6.3. Connectivity View . . . 71
6.3.1. Cannot Set Notify on Modify . . . 71
6.4. Provisioning View . . . 72
6.4.1. Target System Creation Fails . . . 72
6.5. Workflow Runtime Window . . . 72
6.5.1. SHM9550 . . . 72
6.5.2. Refresh Takes Too Much Time . . . 73
6.5.3. Wait for ACK of Create Instance Error . . . 73
7. Problems with Web Applications . . . 74
7.1. Access to an Application Fails With HTTP Status 404 . . . 74
8. Problems with Web Center . . . 75
8.1. Broken Pages after Session Upgrade . . . 75
8.2. Broken Pages Due to Missing X Display Server . . . 75
8.3. CAPICOM Error Message with Enabled Client Signature . . . 76
8.4. Change/Reset Password Failure . . . 76
8.5. Display Privileged Account Password Failure . . . 77
8.6. Error Messages when Approving Request Workflow Tasks . . . 77
8.7. Incorrect Login Link . . . 77
8.8. Login Fails for Any User . . . 78
8.9. Login Fails for a Specific User . . . 78
8.10. Request Workflow Access Failure . . . 79
8.11. Request Workflow Access Failure in Case of Single Sign-On . . . 80
8.12. Session Timeout . . . 80
8.13. Size Limit Exceeded . . . 81
8.14. Too Many Sessions . . . 81
8.15. Wrong Role Parameter Display . . . 82
9. Problems with Web Admin . . . 84
9.1. No Connection to Java-based Identity Server . . . 84
9.2. Node for C++-based Identity Server is not Displayed . . . 84
10. Problems with Web Services (SOA) . . . 86
10.1. Exception NoClassDefFoundError when Running Web Service Clients . . . 86
10.2. Memory Problems with Web Services . . . 86
11. Problems with Request Workflows . . . 87
11.1. Request Workflow Uses the Wrong Initiator . . . 87
12. Problems with Identity Services . . . 88
12.1. Starting and Stopping Services . . . 88
12.2. Warnings due to Missing Definitions within Object Descriptions . . . 89
12.3. C++-based Identity Server Errors . . . 90
12.3.1. Authentication Failed . . . 90
12.3.2. Can’t Connect to LDAP Server . . . 90
12.3.3. Can’t Extract Serial Number from keyOwnerPSE . . . 90

12.3.4. Can’t Retrieve ATS Server Data . . . 91
12.3.5. Configuring a Maximum of More than 128 Threads Fails . . . 91
12.3.6. CryptoException: Export keys expired! . . . 92
12.3.7. DirX Identity Service Does Not Start Correctly . . . 92
12.3.8. Transferring File Fails (File Service) . . . 92
12.3.9. High Load during Restart . . . 93
12.3.10. Internal Windows NT Error . . . 93
12.3.11. Service does not start with encryption enabled . . . 93
12.3.12. No Statistics with Compression Mode . . . 94
12.3.13. Service Not Available . . . 94
12.3.14. Service Stops Directly after Start . . . 95
12.3.15. SHM9550 . . . 95
12.3.16. The C++-based Server Does Not Run Properly Due to a Hostname Change . . . 95
12.4. Java-based Identity Server Errors . . . 96
12.4.1. Cannot Claim Ownership Problem . . . 96
12.4.2. Class Loading Problems . . . 97
12.4.3. Class Loading Problems due to Customer-specific Job Deployment . . . 98
12.4.4. Corrupted Repository Files . . . 98
12.4.5. Escalation not entered for Request Workflows . . . 98
12.4.6. Java-based Server Does Not Stop on UNIX . . . 100
12.4.7. Java-based Server Processing Stopped after Re-Boot . . . 100
12.4.8. java.io.EOFException . . . 101
12.4.9. No Automatic Restart of the Server after Crash on Linux . . . 101
12.4.10. Repository Exceptions for Duplicate Workflows . . . 101
12.4.11. Running Java-based Agents with SSL to the LDAP Server . . . 102
12.4.12. UNIX: Problem Stopping Java-based Identity Server Running with SSL . . . 103
12.4.13. Unable to Run More than One Java-based Server per Domain . . . 103
12.4.14. "No Ldap Entry available for" Messages During Startup . . . 104
12.4.15. ActiveMQ Warning about Missing Certificates . . . 105
13. Problems with LDAP Servers . . . 107
13.1. "No Such Attribute or Value" Error . . . 107
13.2. Service does not Operate Properly after Directory Restore . . . 107
13.3. "Unwilling to Perform" Error . . . 108
13.4. DirX Directory . . . 108
13.4.1. Check-pointing does not Work . . . 108
13.4.2. DIRX_LOGCFG_FILE Variable not allowed . . . 109
13.4.3. Insufficient Size Limit for Searches . . . 109
13.4.4. Uninstallation and Installation . . . 110
13.4.5. "Unwilling to Perform" Error . . . 111
13.4.6. Wrong use of DAP . . . 112
14. Problems with Utilities . . . 113
14.1. Locale Dependency Problem With logANT . . . 113

14.1.1. Run Workflow Tool Hangs . . . 113
14.1.2. dirxgenpse Cannot Contact LDAP Server . . . 113
15. Problems in Complex Scenarios . . . 115
15.1. Password Synchronization . . . 115
15.1.1. Initial Checks . . . 115
15.1.2. Identifying the Component . . . 116
15.1.3. Checking Windows Password Listener . . . 116
15.1.4. Analyzing Java-based Server Operation . . . 117
15.1.5. Analyzing Event Manager Operation . . . 118
15.1.6. Analyzing Target System Workflow Operation . . . 119
15.1.7. Analyzing Account Password Manager Operation . . . 120
16. Logging Messages . . . 121
16.1. Logging Messages for the C++-based Server Components . . . 121
16.1.1. Logging Messages (ACL) . . . 122
16.1.2. Logging Messages (AGC) . . . 122
16.1.3. Logging Messages (AMQ) . . . 130
16.1.4. Logging Messages (CDB) . . . 136
16.1.5. Logging Messages (CMR) . . . 160
16.1.6. Logging Messages (CSR) . . . 163
16.1.7. Logging Messages (CUT) . . . 166
16.1.8. Logging Messages (CWR) . . . 167
16.1.9. Logging Messages (DXC) . . . 168
16.1.10. Logging Messages (F1_) . . . 169
16.1.11. Logging Messages (FSV) . . . 170
16.1.12. Logging Messages (JDBF) . . . 173
16.1.13. Logging Messages (MSS) . . . 174
16.1.14. Logging Messages (SDR) . . . 202
16.1.15. Logging Messages (SHM) . . . 210
16.1.16. Logging Messages (STT) . . . 213
16.1.17. Logging Messages (UTL) . . . 216
16.1.18. Logging Messages (WFE) . . . 220
16.2. Logging Messages for the Common Java Components . . . 227
16.2.1. Logging Messages (CFG) . . . 227
16.2.2. Logging Messages (JMSD) . . . 248
16.2.3. Logging Messages (JMXSOAP) . . . 252
16.2.4. Logging Messages (MSG) . . . 253
16.2.5. Logging Messages (PSF) . . . 253
16.2.6. Logging Messages (RPL) . . . 260
16.2.7. Logging Messages (STG) . . . 264
16.2.8. Logging Messages (SVCMGMT) . . . 275
16.3. Logging Messages for the Connectivity Components . . . 276
16.3.1. Logging Messages (AAD) . . . 278

16.3.2. Logging Messages (ADS) . . . 284
16.3.3. Logging Messages (AEP) . . . 288
16.3.4. Logging Messages (ALN) . . . 289
16.3.5. Logging Messages (ANF) . . . 291
16.3.6. Logging Messages (ASC) . . . 292
16.3.7. Logging Messages (ASI) . . . 306
16.3.8. Logging Messages (C1_) . . . 322
16.3.9. Logging Messages (CSP) . . . 323
16.3.10. Logging Messages (EEC) . . . 323
16.3.11. Logging Messages (GAC) . . . 323
16.3.12. Logging Messages (JDB) . . . 327
16.3.13. Logging Messages (JMS) . . . 331
16.3.14. Logging Messages (LCO) . . . 333
16.3.15. Logging Messages (O365) . . . 342
16.3.16. Logging Messages (ODB) . . . 344
16.3.17. Logging Messages (OICF) . . . 350
16.3.18. Logging Messages (PVC) . . . 352
16.3.19. Logging Messages (RACF) . . . 354
16.3.20. Logging Messages (RWC) . . . 355
16.3.21. Logging Messages (SAMP) . . . 356
16.3.22. Logging Messages (SIPORT) . . . 357
16.3.23. Logging Messages (SLSF) . . . 358
16.3.24. Logging Messages (SPCO) . . . 359
16.3.25. Logging Messages (SPUH) . . . 365
16.3.26. Logging Messages (SPV1V2) . . . 367
16.4. Logging Messages for High Availability . . . 371
16.4.1. Logging Messages (ADC) . . . 371
16.5. Logging Messages for the Java-based Server Components . . . 374
16.5.1. Logging Messages (ACKUPD) . . . 375
16.5.2. Logging Messages (BCK) . . . 376
16.5.3. Logging Messages (CCC) . . . 377
16.5.4. Logging Messages (CPGCTRL) . . . 391
16.5.5. Logging Messages (CPGSVC) . . . 393
16.5.6. Logging Messages (DCFG) . . . 398
16.5.7. Logging Messages (EBR) . . . 398
16.5.8. Logging Messages (ELM) . . . 410
16.5.9. Logging Messages (ICF) . . . 413
16.5.10. Logging Messages (IDSJ) . . . 419
16.5.11. Logging Messages (IDSJADP) . . . 458
16.5.12. Logging Messages (IDSJAPI) . . . 466
16.5.13. Logging Messages (IDSJBS) . . . 469
16.5.14. Logging Messages (IDSJCFG) . . . 470

16.5.15. Logging Messages (IDSJJOB) . . . 473
16.5.16. Logging Messages (IDSJUTL) . . . 482
16.5.17. Logging Messages (IOJ) . . . 483
16.5.18. Logging Messages (IPO) . . . 491
16.5.19. Logging Messages (JAL) . . . 491
16.5.20. Logging Messages (JINI) . . . 494
16.5.21. Logging Messages (JINIAPI) . . . 506
16.5.22. Logging Messages (JOIN) . . . 507
16.5.23. Logging Messages (LCP) . . . 531
16.5.24. Logging Messages (REQWF) . . . 532
16.5.25. Logging Messages (RSKCTRL) . . . 562
16.5.26. Logging Messages (RTWF) . . . 563
16.5.27. Logging Messages (SCE) . . . 563
16.5.28. Logging Messages (SCKJOB) . . . 565
16.5.29. Logging Messages (TCK) . . . 567
16.5.30. Logging Messages (WEBADM) . . . 578
16.6. Logging Messages for the DirX Identity Manager . . . 579
16.6.1. Logging Messages (GUI) . . . 579
16.7. Logging Messages for the DirX Identity Services . . . 580
16.7.1. Logging Messages (APM) . . . 581
16.7.2. Logging Messages (AUD) . . . 583
16.7.3. Logging Messages (CAG) . . . 586
16.7.4. Logging Messages (CHM) . . . 591
16.7.5. Logging Messages (GID) . . . 598
16.7.6. Logging Messages (HIS) . . . 598
16.7.7. Logging Messages (LAN) . . . 604
16.7.8. Logging Messages (LMT) . . . 607
16.7.9. Logging Messages (LNC) . . . 610
16.7.10. Logging Messages (MCO) . . . 612
16.7.11. Logging Messages (POA) . . . 615
16.7.12. Logging Messages (PSV) . . . 629
16.7.13. Logging Messages (PWDMGMT) . . . 653
16.7.14. Logging Messages (PWRST) . . . 661
16.7.15. Logging Messages (RBA) . . . 664
16.7.16. Logging Messages (RPC) . . . 669
16.7.17. Logging Messages (RWF) . . . 677
16.7.18. Logging Messages (SEC) . . . 682
16.7.19. Logging Messages (SRT) . . . 685
16.7.20. Logging Messages (SVC) . . . 689
16.7.21. Logging Messages (SVO) . . . 745
16.7.22. Logging Messages (WDM) . . . 755
Legal Remarks . . . 759

Preface

The *DirX Identity Troubleshooting Guide* describes how to solve problems that can occur in DirX Identity installations.

DirX Identity Documentation Set

Version: 9.0.2 | Date: 2026-11-02 | Timestamp: 13:57:36 UTC

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 Resolution Service Setup Guide*. Use this book to set up the DirX Identity Resolution Service as a standalone service.
- *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.
- *DirX Identity REST Service 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. Introduction

The DirX Identity Troubleshooting Guide describes how to solve problems that can occur in DirX Identity installations and gives hints for general system handling. The Guide describes how to:

- General remarks
- Get information about the state of the system
- Solve installation and configuration problems
- Solve Provisioning problems
- Solve Connectivity problems
- Solve DirX Identity Manager problems
- Solve Web Applications problems
- Solve Web Center problems
- Solve Web Admin application problems
- Solve Web Services problems
- Solve request workflow problems
- Solve DirX Identity Server problems
- Solve LDAP Server problems
- Solve problems with utilities
- Solve problems in complex scenarios
- Handle error messages

Please note that small enhancements and bug fixes for DirX Identity are delivered from time to time as service packs or hot fixes.

1.1. General Remarks

DirX Identity is a comprehensive product with a huge set of features.Identifying and solving problems in such an environment is not an easy task.

Some general remarks in this section are intended to help avoid problems and explain some general guidelines or principles on how to locate and solve problems.

1.2. Avoiding Problems

You can minimize the necessity to analyze a problem if you follow some guidelines.This strategy helps to reduce problems or even avoid them completely.

- **Implement your solution stepwise** (incremental) and test it to reduce the number of possible problem causes.
- Be sure to **understand the theory of a feature**.This knowledge eases and speeds up

problem analysis enormously.

- **Understand the data flow**.Check (and break) the data flow where necessary.
- **Use off-line tests** wherever possible, especially for workflow design.
- **Build (semi-) automatic transport mechanisms** if you move data within staged environments.
- **Test your application thoroughly** and in the correct environment (as far as possible).
- **Run check tasks regularly** (LinkChecker, ...) in a production environment and solve the displayed problems.Do not ignore them because this could result in a much bigger problem in the future (for example, a filled-up Message Server that stops processing).
- **Do not copy jar (Java archive) files** without changing the extension to something other than **jar**. Use, for example, **jar_old**, **old** or **bak**. This is common Java development knowledge.

  If you do not follow this rule, the Java machine uses both jar files (for example, original.jar and original_copy.jar). Both jar files can contain the same methods. It is not predictable which method is used. Thus strange and hard-to-find errors can occur.
- **Check and adapt default time condition values** according to your system requirements and system load. In the Connectivity view group, check the following important parameters:

Configuration → **DirX Identity Servers** → **C Servers** → *yourServer*:

- JMX Access: JMX Timeout
- AgtSvr Configuration: Max number of threads
- AgtSvr Configuration: KeyGet timeout

Configuration → **DirX Identity Servers** → **Java Servers** → *yourServer*:

- Limits: Tomcat: Max No. of Threads
- Be sure **to write all Lotus Notes parameters correctly** regarding case sensitivity.

1.3. Following Problem-Solving Guidelines

If a problem occurs, try to follow these guidelines to locate and solve it:

- Read the error messages carefully.Many tickets that reach us suffer from improper reading.The cause is often obvious from the message.Trust the message as far as possible.
- Look up the error message in this troubleshooting guide.Read the explanation and follow the solution sketches.
- Locate the problem.Use the following questions:
- Where did the error occur?
- Are there any other errors that are related to this one?If yes, which one is the important one?

- Which component caused the issue?
- If the error occurred in a complex environment, try to reduce / isolate the problem:
- Reduce the amount of data and check if the error still occurs.
- In a chain of procedures or components, try to isolate or find the part that causes the error and then try to reproduce the error.
- Try to approach the problem from different sides.Does the error occur under different conditions?
- If you cannot solve the problem, check the DirX Wiki to see if someone else has already found and solved the issue.
- 'Walk around the building', this might help open your mind to alternative approaches.

If nothing has helped up to here: open a ticket and send it to support.See the next section for details on this process.

1.4. Opening Tickets

You can open tickets for various purposes on DirX Ticket Portal:

- **Request consulting support** - use this type of ticket if you are not sure how to solve a customer requirement.Use it also to clarify the right alternative if you are not sure which feature of DirX Identity to use.
- **Report an Incident** - use this type of ticket to create an incident ticket. Use the DiagTool to collect the necessary basic information.Add a short but precise description of the problem. Use screenshots to explain the issue if appropriate.Add additional information if this makes sense.
- **Request a License** - use this type of ticket to request License-Files. Fill out the form with all mendatory information.
- **Request a change** - use this type of ticket to request new functionality. Describe your idea shortly but clearly. Explain the motivation of this request and why you cannot realize your requirement with the existing functionality.

In case you do not have access to the above Ticket Types, please request access first.

Do not forget to set the correct severity for your incident:

- Severity 1-Critical: Bug capable of triggering complete system shutdown.
- Severity 2-Major: Bug capable of collapsing large parts of the system.
- Severity 3-Minor: Results in some unexpected or undesired behavior, but not enough to disrupt system function.
- Severity 4-Low: Bug won’t result in any noticeable breakdown of the system.

1.5. Getting Logging Message Information

If you receive a logging message from DirX Identity, you can look up information about it in

the DirX Identity Manager's help system.

1. Start DirX Identity Manager.
2. Click **Help** → **Help** (the message **loading the help topics** is displayed).
3. Click the **Search** tab and type the error message number in the **Find** field (for example, **CDB4001**).Alternatively, you can type a part of the message text into the **Find** field of the **Search** tab; for example, **wrong schema version**).
4. Enter **Return**.

The help system displays error information that corresponds to the error message you specified.This information consists of the **message** itself, an **explanation** and possible **actions**.

Alternatively you can use the PDF documentation to look up information about logging messages. See the Acrobat Reader help for information on how to search issues.

2. Guidelines for Specific Components

This chapter describes where to find valuable information for important DirX Identity components (for example, log files) and where to set important switches to influence the logging process. It also provides hints about typical problems.

Be aware that you can use the **Windows Event Viewer** on Windows to view logging information. The information depth can be configured with the logging configuration files or the filters provided with the Event Viewer.

For the **Java-based Server**, logging can be configured for syslog.

You can find information for the following components:

- Installation and Configuration
- Manager
- Object Descriptions
- Web Center
- Policies
- Services
- C++-based Server
- Java-based Server
- Java-based Workflows
- Tcl-based Workflows
- Request Workflows

2.1. Analyzing Global Aspects

This section explains issues related to many components and provides hints on how to analyze and solve these issues.It is divided into three parts:

- Did you follow the design guidelines?
- Perform link checking regularly
- Analyze and solve performance problems

Did you follow the Design Guidelines?

You can avoid problems if you try to follow these guidelines:

- From the very beginning, design your system having performance aspects in mind. Be aware of possible bottlenecks and collect requirements on system load.
- Select built-in features instead of building your own special solutions. This avoids unnecessary effort and eases migration.
- Use only the features that you really need. Disable the rest. Especially important are

    - Domain object: Inherit role parameters: delays privilege resolution
    - Domain object: Offline resolution (also available as separate flag in Web Center)
    - Target System object: Reference group from account (avoids huge groups)
    - Target System object: Ignore nested groups (avoids checking complex group structures)
- Use the latest versions and service packs (which typically have higher performance).
- Use cluster workflows if you want to provision sets of similar target systems.
- Follow the guidelines for access policy filters.
- Test your system with realistic load early.

Performing Regular Link Checking

An Identity management system is a dynamic system. Even if you have well-designed processes and check procedures, data quality is never 100%. One problem is broken links.

Use the link checker regularly to analyze the situation and to optimize your system. Read more about the link checker and its operation in the *DirX Identity User Interface Guide*.

Be aware that the link checker comes with a default configuration file. Keep this file up to date with the configured links in your Identity system.

Analyzing Performance Problems

If you run into performance problems, try to identify the component and the problem cause and then try to solve it.The following hints can help to manage performance:

- Missing indexes in the directory database can lower performance enormously.To find these problems, enable auditing at the directory server.Run the relevant use cases and then check the operations with the highest duration.Evaluate especially the LDAP filter conditions of these operations.If missing indexes are the cause, add these indexes and run your use cases again to verify that the problem is gone.
- Read the section "Tuning the Provisioning System" in the *DirX Identity Provisioning Administration Guide* and follow these guidelines to optimize Manager and Web Center operation.
- Use the performance relevant switches in the **dxi.cfg** file to optimize Manager operation.For details see the section "Customizing the Property File (dxi.cfg)" in the *DirX Identity User Interface Guide*. Use especially **monitorview.refresh**, **processtable.refresh** and **resolution.mode**.
- Be aware that the use of the **Segregation of Duties** flag at the domain object in combination with policy execution lowers performance significantly.

2.2. Analyzing Installation and Configuration Problems

This section explains where to find information to analyze installation and configuration-

related problems and provides hints on how to analyze and solve these issues.

Where to find information?

These two files in the folder *install_path* are relevant for installation and configuration:

install_history.txt - this file contains global information about installed versions, service packs and hot-fixes.It is written and maintained by the product installer, the service pack installer and the hot-fix installer.

configuration.ini - this file is instantiated by the installer (see the **Installed.Components.***component* lines). It is the interface to the configuration tool that extends the file according to the user input (see the *Configured.Components.**component* lines as well as the related parameters). The silent installation uses the *option.**component* lines.

basic.input.tcl - this file is used as a parameter source for various scripts (for example, schema extensions). Parameters are set via the Configuration Wizard.

component.input.tcl - this file is used as a parameter source for extension of C++-based Server objects regarding agent information. Parameters are set via the Configuration Wizard.

For more information about installation and configuration, see the *DirX Identity Installation Guide*.

The *install_path* directory contains this set of log files that is relevant for the installer:

Atos_DirX_Identity_V*x.x*_Install_***timestamp***.log* - this file contains the installer logging and *timestamp* represents date and time of the related installation launch. Check this file for error messages if you run into problems.

The *install_path**\logs* directory contains this set of log files that is relevant for the installer:

dirxidty_debug.txt - contains additional information together with the following file.

dirxidty-installer_debug.txt - contains additional information.

The *install_path**\logs* directory contains this set of log files that is relevant for the Configuration Wizard:

ConfigurationWizard_<date>.log - contains the input parameters and other details logged by the Configuration Wizard.

Deployment_<date>.log - contains the output of the configuration tasks.

Analysis-relevant switches and options

There are no specific switches available for debugging the installation or configuration.

Procedures and hints

Be aware that you cannot install the directory schema remotely.Perform this step for Connectivity and Provisioning directly on the corresponding machine.

2.3. Analyzing Manager Problems

This section explains where to find information to analyze DirX Identity Manager-related problems and provides hints on how to analyze and solve these issues.

Where to find information?

Find the relevant log files in the folder *install_path***gui\logs**

system.*nnn*.**log** - this file contains all messages the DirX Identity Manager produces.Standard behavior is that the file is deleted during start of the Manager.

dximanager.*nnn*.**log** - this file is normally empty. It contains only messages if errors and warning occur.

ldap.DirXmetaRole.*nnn*.**log** - this file contains all LDAP requests and responses for the Provisioning view.

ldap.DirXmetahub.*nnn*.**log** - this file contains all LDAP requests and responses for the Connectivity view.



in contrast to the LDAP audit information of the DirX Directory server, both LDAP files contain the complete response of the request. Because this can be a huge amount of information, switch on LDAP logging only if it's really required.

Analysis-relevant switches and options

The following switches are helpful for problem analysis and logging. For a complete description of the switches see the chapter "Customizing the Property File (dxi.cfg)" in the *DirX Identity User Interface Guide*.

assign.InitialSearch
ldap.trace
trace.level
trace.maxline
trace.timestamp.format

The following switches influence performance:

aclmgr.refresh.interval
assign.InitialSearch
ldap.switch2offline
monitorview.refresh
processtable.refresh

report.sizelimit
resolution.mode

Procedures and hints

Use the log files to check correct operation of the DirX Identity Manager:

- Search the **system**.*nnn*.**log** files for the strings **ERR(** and **WAR(** especially before you start production.There should not be any errors or warnings.
- Depending on the log level, you can find additional information in the **dximanager**.*nnn*.log files.
- Do not forget to switch off all analysis switches before you start in a production system because they could lower performance in the DirX Identity Manager.

If these procedures aren't sufficient to identify the problem and your DirX Identity installation uses the DirX Directory server, you can use LDAP session tracking as an additional method to identify the problem.See the section "Using LDAP Session Tracking to Analyze Component Problems" for details.

2.4. Analyzing Object Description Problems

This section explains where to find information to analyze object description related problems and provides hints how to analyze and solve these issues.

Where to find information?

Find the relevant log files in the folder *install_path***gui\logs**

system.*nnn*.**log** - contains all messages the DirX Identity Manager produces.Problems in object or component descriptions are visible in these log files (startup phase of the Manager).

Analysis-relevant switches and options

No specific switches are necessary

Procedures and hints

Perform this basic procedure

- Create and change object descriptions incrementally.
- Test from time to time whether errors or warnings show up in the log files:
- Start the Manager
- Check the log files for "ERR(" and "WAR(" strings to be sure that everything is correctly defined.
- If messages occur, analyze them and correct the problem cause.

2.5. Analyzing Web Center Problems

This section explains where to find information to analyze Web Center related problems and provides hints on how to analyze and solve these issues.

Where to find information?

Web Center runs on Tomcat, which means that you can find all relevant log information in the Tomcat installation area.

Find all log files in the Tomcat log folder (for example: *tomcat_install_path***Apache Software Foundation\Tomcat 11.0\logs**).Check these files in case of problems:

localhost.date.log - Tomcat and Web Center startup log information

stdout_date.log - log files with relevant runtime information

Analysis-relevant switches and options

Find Web Center configuration files in the DirX Identity installation in the folder *install_path***web\webCenter-domain\WEB-INF**

web.xml - general Web Center properties

config\webCenter.properties - application specific properties for Web Center

Procedures and hints

Use the Web Center **logViewer** to analyze Web Center log files. Perform these steps:

- Unpack the tool to the Tomcat log directory.
- Copy a log file to the **logViewer\log** folder
- Rename it to **log.txt**
- Run the **logViewer\run.bat** file. After a view seconds the result is created in the log folder
- Start the **summary.html** page

The tool displays all requests (struts actions) in a list (use the tool tips to understand the columns and items).

Clicking a row opens a separate window that displays the complete logging part for this specific request.

For a detailed description of the logViewer, see the *DirX Identity User Interface Guide*.

If these procedures aren't sufficient to identify the problem and your DirX Identity installation uses the DirX Directory server, you can use LDAP session tracking as an additional method to identify the problem.See the section "Using LDAP Session Tracking to Analyze Component Problems" for details.

2.6. Analyzing Policy Problems

This section explains where to find information to analyze policy-related problems and provides hints on how to analyze and solve these issues.

In this section we concentrate on access policy problems.The typical question is: "Why does a user have this access right?"

The next paragraphs provide hints on how to answer this question.

Where to find information?

You set up all access policies in the DirX Identity Manager in **Provisioning → Policies → Access Policies** folder.

View the result of the access policies in Web Center: **Delegation → Show access rights**. For each operation and object type you can find a row in the table. Click the button at the end of the line to view the details; that is, the objects on which the user is allowed to perform the named operation.

Analysis-relevant switches and options

If you have problems in a test environment, use these switches at the **Domain** object (tab **Policies**) to check whether the problem comes from access policies:

Enable menu policies - disable this switch to find out whether menu policies influence the accessibility of a specific function.

Disable access policies - enable this switch to switch off access policies completely. This helps to decide if a problem comes from access policies or not.

Procedures and hints

To analyze the influence of access policies on the actions a specific user can perform on a specific set of objects, you need to understand that the access rights of a specific user are the combined result of all relevant access policies. This is the reason why it is difficult to identify the causing access policies in a complex project.

Try to follow these guidelines to solve access policy problems:

- Work incrementally if you create access policies. Check intermediate results.
- Try to reduce the problem.

A general approach is:

- Use the queries in the DirX Identity Manager to find the relevant access policies.Find default queries in **Provisioning** → **Policies** → **Access Policies** → **_Queries**.The set-up queries work by object type or by operation.Do not hesitate to set up your specific queries.The result is a set of access policies that you should check further to see if they cause your problem.

- Verify the access rights of a user with **Delegation → Show Access Rights** in Web Center.
- Deactivate an access policy to check whether it has an influence.Be aware that there is an access policy cache that is refreshed every ten minutes by default.To make changes effective immediately, perform a log out / log in sequence in Web Center or restart the Manager.

If these procedures aren't sufficient to identify the problem and your DirX Identity installation uses the DirX Directory server, you can use LDAP session tracking as an additional method to identify the problem.See the section "Using LDAP Session Tracking to Analyze Component Problems" for details.

2.7. Analyzing Services Problems

This section explains where to find information to analyze services related problems and provides hints on how to analyze and solve these issues.

Where to find information?

The DirX Identity service layer runs either standalone as a Java-based job as part of a Tcl-based workflow or it runs in any component that uses the service layer (Manager, Web Center, event-based workflows in the Java-based Server).

Find information about a problem in the relevant log files.

Analysis-relevant switches and options

If the problem is not yet obvious from the error message, enhance the trace or log levels to get more information. Repeat the test.

Procedures and hints for Privilege Resolution

1) Complex privilege resolution problems

Privilege resolution is a complex process that is not easy to manage if problems occur in a complex environment where thousands of identities are resolved via the service agent.

Try to reduce the problem while using this procedure in the DirX Identity Manager:

- Find out which identity or related object causes the problem.
- Set the log level to 9 (dxi.cfg file).
- Start the Manager and try to reproduce the problem (for example, assign or unassign a privilege by hand).

Check the log file carefully (see the section "Analyzing Manager Problems" for more information). This should show the error cause in detail; for example, incorrect object descriptions.

2) Unwilling to perform messages

If you encounter "Unwilling to perform" from the LDAP server during Privilege Resolution, check that the **Paging Policy Timeout** and the **Session Timeouts** of the LDAP server are configured correctly.

If you don’t want to increase these values, change the configuration as described in the *Application Development Guide*: **Using the Maintenance Workflows → Understanding the Tcl-based Maintenance Workflows → Privilege Resolution Workflow → Privilege Resolution Workflow Optimization**.

If these procedures aren’t sufficient to identify the problem and your DirX Identity installation uses the DirX Directory server, you can use LDAP session tracking as an additional method to identify the problem.See the section "Using LDAP Session Tracking to Analyze Component Problems" for details.

2.8. Analyzing C++-based Server Problems

This section explains where to find information to analyze C++-based Server-related problems and provides hints on how to analyze and solve these issues.

Where to find information?

Find the log files in the folder

*install_path***server\log**

Analysis-relevant switches and options

The startup of the server is controlled by the file

*install_path***server\conf\dxmmsssvr.ini**

You can configure the logging levels and the logging behavior in the folder

*install_path***server\conf**

Use the **dirxlog.cfg** file to adjust the logging (see the *DirX Identity Meta Controller Reference* for details).

Procedures and hints

1) Startup problems

Note that the startup phase is controlled by the **dxmmsssvr.ini** file parameters). The server registers only if file and LDAP parameters are consistent (**dnServerName** and **host**). Correct these parameters if they are not correctly configured.

Due to the fact that the LDAP server can be installed on another machine, the server repeats LDAP binds. See the parameters **repeat** and **timeout** in the **dxmmsssvr.ini** file. Adjust these parameters if not sufficient!

Additionally, the watchdog repeats server starts. Configure the parameter **server_restart** accordingly.

2) Using logging

Use the log files as described above and configure logging in the **dirxlog.cfg** file as needed. (see the *DirX Identity Meta Controller Reference* for details).

Use the command **metahubdump** to convert the binary log files to readable format (see the *DirX Identity Meta Controller Reference* for details).

Analyzing errors and warnings:

- Search for the string '-- error' to find errors in the file
- Search for the string '-- warning' to find warnings in the file
- Look up error numbers in the documentation. Read the provided reason and solution paragraphs carefully.

On Windows platforms, you can use the Windows Event Viewer alternatively or additionally.View the log entries in the Windows Event Viewer and use the filter function of the Event Viewer.See the Microsoft documentation for more information.

3) Using the Get Server State method

You can use the DirX Identity Manager **Get Server State** command to request the state of each **C++-based Identity Server**.See the "Get Server State" topic in the *DirX Identity User Interface Guide* for more information.

If these procedures aren't sufficient to identify the problem and your DirX Identity installation uses the DirX Directory server, you can use LDAP session tracking as an additional method to identify the problem.See the section "Using LDAP Session Tracking to Analyze Component Problems" for details.

2.9. Analyzing Java-based Server Problems

This section explains where to find information to analyze Java-based Server-related problems and provides hints on how to analyze and solve these issues.

Where to find information?

The Java-based server provides a comprehensive set of log files in the folder

*install_path***ids-j-***domain*-**S***n***logs**

You can open the log files from this directory or from the Web Admin application in the section **Logging → View Log Files**.

These files are available:

start.log - provides information about the server start-up phase.

classloader.txt - helps to analyze class loading problems.

stderr.*nnn*.log* - delivers messages that otherwise cannot be logged (logging not yet initialized).

server-*timestamp*-*counter*.txt* - contains all log messages ordered per timestamp. Set logging levels in Web Admin to influence the logging depth.

warnings-*timestamp*-*counter*.txt* - contains a collection of all errors and warnings that occurred (find full information is in **server-*timestamp*-*counter***.txt* files)

serverstate.txt - provides the complete state of server is written every hour.

server.xml - contains the complete loaded server XML configuration. This is the resolved content of the corresponding Java-based Server entry in the Connectivity view (**Content** tab).

Analysis-relevant switches and options

Set up logging levels for the **Java-based Identity Server** Web Admin application in the section **Logging → Set Log Levels**.

For more information about the Web Admin application, see "Using Web Admin" in the *DirX Identity User Interface Guide*.

Procedures and hints

1) Analyzing log files

Use the logANT tool to produce log files for individual workflows. For more information see the *DirX Identity User Interfaces Guide*.

Analyze errors and warning by searching the string

'*** WARNING' for warnings

'*** SEVERE' for errors

2) Analyzing class loading problems

You can analyze class loading problems in the Java-based Server in two ways.

Checking whether a specific JAR-file is loaded

- Check the file **classloader.txt** whether a specific jar file is loaded.
- Check also the location of the loaded jar file and whether it is maybe loaded multiple times (maybe the wrong version is loaded?).

Checking whether a specific class is loaded

- To check loading of a specific class, set for this class in the file _ install_path_**\ids-j-***domain*-**S***n***\bin\clcfg.properties**

the property **class=true**

- Restart the server.
- Check the **classloader.txt** file whether the class is correctly loaded.

3) Viewing the server status

Use the Web Admin application to verify the status of each **Java-based Identity Server**.For more information about the Web Admin application, see "Using Web Admin" in the *DirX Identity User Interfaces Guide*.

Check the **server.xml** file in the folder *install_path***ids-j-***domain**-S**n***logs** to find out what the actual configuration of this server is (all references are resolved in this file).

If these procedures aren't sufficient to identify the problem and your DirX Identity installation uses the DirX Directory server, you can use LDAP session tracking as an additional method to identify the problem.See the section "Using LDAP Session Tracking to Analyze Component Problems" for details.

2.10. Analyzing Java-based Workflow Problems

This section explains where to find information to analyze Java-based workflow related problems and provides hints on how to analyze and solve these issues.

Where to find information?

You can view the complete logging information for these workflows in each Java-based Server instance in the folder

*install_path***ids-j-***domain**-S**number***logs**

The files of interest are:

server-*timestamp-counter***.txt** - all log messages ordered per timestamp. Set logging levels in Web Admin to influence the logging depth.

warnings-*timestamp-counter***.txt** - a collection of all errors and warnings that occurred (find full information is in **server-***timestamp-counter***.txt** files)

Analysis-relevant switches and options

Set up logging levels for the **Java-based Identity Server** Web Admin application in the section **Logging → Set Log Levels**.

For more information about the Web Admin application, see "Using Web Admin" in the *DirX Identity User Interface Guide*.

Procedures and hints

These hints can help to isolate and reduce problems in complex environments:

1) Use extended Join logging

Analyze problems in real-time workflows while you set the Join logging (class com.siemens.dxm.join) via Web Admin to 'Finest' or 'All'. This action reveals a lot of details and helps to understand what is really going on.

2) Perform stand-alone tests

To isolate complex problems, perform stand-alone real-time workflow tests that run outside the Java-based Server. Use this method also during creation of new workflows and connectors.

3) Monitoring Provisioning Workflows

The Monitor View presents a lot of information about running or completed provisioning workflow runs. See the "Monitor View" and "Using the Monitor View" topics in the *DirX Identity User Interfaces Guide* for basic information.

There are several ways to use the Monitor View:

- Open the folder in the tree pane with the name of the desired workflow. Click the status entry from the workflow that you'd like to explore (hint: the last entry in the list is the newest one). To view a status entry with a specific date and time, click on the entries and view the **Start Time** field, which displays the local time (you can use the cursor keys to move between the entries in the tree).
- Use the predefined filters or (set up your own) to view specific status entries. See the 'Query Folder' topic in the *DirX Identity User Interfaces Guide* for information on how to set up filters. For example, you can use the **Result=Not OK** filter to get all workflow status entries that did not run correctly. Use relative and absolute time variables to refine your filter definitions, for example display the workflows of the last 3 hours.

To analyze a status entry:

- View the workflow status entry.Look at the **Result** field to check for the status of the entry.
- If the status is not **closed.completed.ok**, the **Remark** field should display additional messages.If the remark field is empty, view the corresponding activity status entries (remark field and statistics).The remark field contains a line for each performed action with all necessary information about the result.It displays also error and warning information.If you need more information, look up the Java-based Server log files.
- You can use the additionally provided information in the *DirX Identity Trouble Shooting Guide* to analyze the messages displayed here.If the messages are not completely displayed, use the button at the bottom on the left side of the field to read the messages in your preferred text editor.

Note that for real-time workflows including password changes, no files are stored in the status area.All available information is present in the status entries.If this is not enough information, you need to check the Java-based Server log files.

If these procedures aren't sufficient to identify the problem and your DirX Identity

installation uses the DirX Directory server, you can use LDAP session tracking as an additional method to identify the problem.See the section "Using LDAP Session Tracking to Analyze Component Problems" for details.

2.11. Analyzing Tcl-based Workflow Problems

This section explains where to find information to analyze Tcl-based workflow related problems and provides hints on how to analyze and solve these issues.

Where to find information?

The central point at which to find information is the **Connectivity → Monitor** area.Here you can find status entries that contain all information about a specific workflow run.

These status entries can relate to files stored in the status area of the C++-based Server.

Note also that in some cases the log files of the C++-based Server can reveal additional problems. See the corresponding section.

Analysis-relevant switches and options

You can control the log levels for the meta controller (**metacp**) in the subdirectory

*install_path***client\conf**

See the description of the **dirxdumplog** command in the *DirX Identity Meta Controller Reference* for further details.

Procedures and hints

The Monitor View presents a lot of information about running or completed provisioning workflow runs. See the "Monitor View" and "Using the Monitor View" topics in the DirX Identity Manager help for basic information.

There are several ways to use the Monitor View:

- Open the folder in the tree pane with the name of the desired workflow. Click the status entry from the workflow that you'd like to explore (hint: the last entry in the list is the newest one). To view a status entry with a specific date and time, click on the entries and view the **Start Time** field, which displays the local time (you can use the cursor keys to move between the entries in the tree).
- Use the predefined filters or (set up your own) to view specific status entries. (See the 'Query Folder' topic in the DirX Identity Manager help for information on how to set up filters). For example, you can use the **Result=Not OK** filter to get all workflow status entries that did not run correctly. Use relative and absolute time variables to refine your filter definitions, for example display the workflows of the last 3 hours.

To analyze a status entry:

- View the workflow status entry. Look at the **Result** field to check for the status of the

entry.

- If the status is not **closed.completed.ok**, the **Remark** field should display additional messages. You can use DirX Identity Manager help to analyze the error messages displayed here. If the messages are not completely displayed, use the button at the bottom on the left side of the field to read the messages in your preferred text editor.

The information provided for C++-based status entries in the **Remark** field is usually not very detailed.Consequently, you should look at the activity or the activities that failed:

- When there are several activities, click the workflow entry in the tree pane.In the list pane (top pane at the right side), you can then view the activities that have a result that is not **closed.completed.ok**.
- Click these activities and view the **Remark** field.This field should contain more detailed information about the specific problem that occurred with this activity.You can use DirX Identity Manager help to analyze the error messages displayed here.
- In many cases, you will find a text message like "See process info file for more information" or "See report or trace file for more information".The message indicates that the **File** tab should contain additional information about the error.
- Click the **File** tab.Then view the **Process Info** file.It can contain the **stdout** and **stderr** information which should give you a more detailed view of the problem.Please note that this information is highly dependent on the information the agent provides.
- If the information in the **Process Info** does not help, you can try to view **trace** or **report** files to find out more.

If these procedures aren’t sufficient to identify the problem and your DirX Identity installation uses the DirX Directory server, you can use LDAP session tracking as an additional method to identify the problem.See the section "Using LDAP Session Tracking to Analyze Component Problems" for details.

2.12. Analyzing Request Workflow Problems

This section explains where to find information to analyze request workflow-related problems and provides hints on how to analyze and solve these issues.

Where to find information?

You can view the complete logging information for these workflows in each Java-based Server instance in the folder

*install_path***ids-j-***domain*-**S***n***logs**

The files of interest are:

server-*timestamp-counter***.txt** - all log messages ordered per timestamp. Set logging levels in Web Admin to influence the logging depth.

warnings-*timestamp-counter***.txt** - a collection of all errors and warnings that occurred (find full information is in **server-***timestamp-counter***.txt** files)

Analysis-relevant switches and options

Set up logging levels for the **Java-based Identity Server** Web Admin application in the section **Logging → Set Log Levels**.

For more information about the Web Admin application, see "Using Web Admin" in the *DirX Identity User Interfaces Guide*.

Procedures and hints

These hints can help to isolate and reduce problems in complex environments:

1) Perform stand-alone tests

To isolate complex problems, perform stand-alone real-time workflow tests that run outside the Java-based Server. Use this method also during creation of new workflows and connectors.

2) Monitoring Request Workflows

Open **Provisioning → Request Workflows → Workflows → Monitor** to view the instances of request workflows. Here you can find all information about running or completed workflows.

To analyze the status entries:

- Open the folder structure and navigate to the folder for the day where you expect the workflow instance.Alternatively you can use the predefined queries or you can define your own queries.
- Check first the workflow entry.The **Status Information** tab presents all important information.Especially the **Error Messages** field presents all related error messages.Analyze these messages to find out the problem cause.
- If you want to view the activity entries, check in the **General** tab in the graphical view which activities failed or where the workflow is waiting for further processing.
- Open the workflow instance entry to display the list of activities.Click the activities and then the **Status Information** tab to view more status information.
- If this is not enough information, check the log files in the Java-based server log folder.You can access these files either in the file system (*install_path**\ids-j-***domain***-S* *n**\logs*) or via the Web Admin application in the logging section.
- If you still do not have enough information and your problem is reproducible, change the log levels via the Web Admin and start another request workflow of the same type.

If these procedures aren’t sufficient to identify the problem and your DirX Identity installation uses the DirX Directory server, you can use LDAP session tracking as an additional method to identify the problem.See the section "Using LDAP Session Tracking to Analyze Component Problems" for details.

2.13. Using LDAP Session Tracking to Analyze Component Problems

If you find that the component-specific logging described in the previous sections does not provide you with a clear picture of the problem you are experiencing and your DirX Identity installation uses the DirX Directory server, you can use LDAP session tracking as an additional way to identify the problem.

This section describes where to find LDAP session tracking information and provides hints on how to use it to analyze and solve component problems.

Where to find information?

If your DirX LDAP server's audit configuration enables LDAP session tracking, all DirX Identity components that access LDAP - DirX Identity Manager, Web Center, Policy Agent, Provisioning and Request workflows and other Connectivity and Provisioning components - augment the DirX LDAP server's audit records with session-tracking related information that allows you to identify, for an LDAP operation:

- The DirX Identity component that invoked the operation
- The user bound to the invoking component
- The computer on which the operation was invoked

You use the DirX Directory commands **dirxauddecode** and **dirxaudstatistics** to evaluate the LDAP audit records and obtain this information.

Analysis-relevant switches and options

If LDAP auditing is not yet enabled in your DirX Directory server environment, use the **dirxcp** command to run the script **ldap_auditcfg.cp**, as described in the *DirX Administration Guide* → "Using LDAP Auditing" → "Enabling LDAP Auditing". This script creates a default LDAP audit configuration subentry for the DirX LDAP server that enables session tracking for LDAP audit records, among other default values. See the section "Attributes for LDAP Server Audit Configuration" in the chapter "DirX Attributes" in the *DirX Administration Reference* for a description of the attributes related to LDAP server audit configuration. See the description of the **dirxcp** command in the *DirX Administration Reference* for usage information on the command.

Procedures and hints

Use the DirX Directory commands **dirxauddecode** and **dirxaudstatistics** to evaluate the DirX Identity component session tracking information in the LDAP audit records to analyze DirX Identity component operations. The most important items in the evaluated LDAP audit records are:

- SID-Name - the name of the invoking component
- SID-Info - the DN of the bind user
- SID-IP - the IP address of the machine on which the component runs

These items are analogous to the sessionSourceName, sessionTrackingIdentifier and sessionSourceIP items described in the section "Using Session Tracking" in the *DirX Administration Guide* and the Internet draft "LDAP Session Tracking Control" (draft-wahl-ldap-session-03.txt).

DirX Identity prefixes every SID-Name with **DXI**. The next part of the SID-Name identifies the component. When possible, the "Application Name" of the LDAP audit record is used. If available, an identifier like the workflow instance ID is appended. The following table describes the SID-Names for DirX Identity components and provides examples of their use:

Component	**SID Name**	**Example**
Identity Manager	DXI Manager ([Connectivity|Provisioning])	DXI Manager(Connectivity)
Identity Web Center	DXI WebCenter *session_ID*	DXI WebCenter DirXmetaRole-591814-2
Policy Agent	DXI PolicyEngine *workflow_instance_ID*	DXI PolicyEngine a5d19c2—12828251-1490dc3300a—7fec#20141014090742Z
MetaRole Service Agent	DXI ConsistencyAgent *workflow_instance_ID*	DXI ConsistencyAgent a5d19c2—669bb17d-14aecca6e00—7fe8#2015011
Web Services	DXI ProvisioningService ProvisioningWebServices-*session_ID*	DXI ProvisioningService ProvisioningWebServices-209319-1
Java-based Provisioning Workflows	DXI [JoinFromDXI | JoinToDXI] *workflowID workflow_name*	DXI JoinFromDXI 1495c804a6e$-724d Validate_JDBC_RealtimeOra
Entry Change Workflows	DXI EventBasedResolution *workflow_instance_ID*	DXI EventBasedResolution 149f06b6032$-7987
Ticket Workflows	DXI ProcessTickets *workflow_instance_ID*	DXI ProcessTickets 149f06b6032$-7988
Request Workflows	DXI ApplyChange *workflow_instance_ID activity_name*	DXI ApplyChange 1495c804a6e$-724d Apply Changes
Campaign Controller	DXI CampaignControl *workflow_instance_ID*	DXI CampaignControl 155057c69a9$-7ea1
Read Configuration	DXI Read[Provisioning|Connectivity]Configuration-S*n*	DXI ReadProvisioningConfiguration-S1
Request Workflow Service (including bulk approval)	DXI WorkflowService-S*n session_ID*	DXI WorkflowService-S1 DirXmetaRole

Component	SID Name	Example
Various Java-based Server components, like Request Workflow Engine	DXIJavaServerInternal-S*n*	DXI JavaServerInternal-S1
Full-Check Java Server	DXI FullCheck-S*n*	DXI FullCheck-S1
WFStatusLogHandler (RT Monitor)	DXI WfStatusLogHandler-S*n*	DXI WfStatusLogHandler-S1
Ldapreader Java Server	DXI LdapReader-S*n*	DXI AbstractLdapResourceManager-S1
Approval REST Services	DXI ApprovalRestService-*sessionID*	DXI ApprovalRestService ApprovalRestService-560312-1
JobSessionFactory Java Server	DXI JavaServer JobSessionFactory-S*n*	DXI JavaServer JobSessionFactory-S1
domcfg/Loader Java Server	DXI JavaServer DomcfgLoader-S*n*	DXI JavaServer DomcfgLoader-S1
domcfg/DomainSessionAccessor Java Server	DXI JavaServer DomainSession-S*n*	DXI JavaServer DomainSession-S1

Use the **-C** option to the **dirxauddecode** and **dirxaudstatistics** commands to specify a SID-Name as a string. For example, specifying the string **DXI WfStatusLogHandler-S1** to the **-C** option directs the commands to generate an output file that contains only those operations issued by an LDAP client whose session-tracking control has the SID-Name **WfStatusLogHandler-S1**. See the descriptions of these commands in the *DirX Administration Reference* for detailed information about their use.

You can use DirX Identity tools to obtain the desired SID-Name string input depending on the component you're trying to track. For example, you can use the Monitor View to obtain a real-time workflow's name and instance ID for input to **dirxauddecode**/**dirxaudstatistics**. You can also use the Monitor View to obtain the name of the Java-based Server on which a real-time workflow ran (for example, **My-Company-S1-MyHostName**); this information is contained in the entry for the real-time workflow in the Monitor View. For other components, you can examine the logs described in "Analyzing Java-based Server Problems" to determine on which Java-based server they're running. You can obtain the session ID for a component by examining the informal messages provided in the component logs for the component; for example, Web Services, REST Approval Service, and so on.

3. Installation / Configuration Problems

This section describes installation and configuration-related problems.

3.1. Installation Problems

This section handles server or agent installation related problems.

3.1.1. Determining which Agents are Installed

Indication:

You are not sure which agents are installed on the different servers of a distributed DirX Identity domain.

Reason:

- Complex changes in your configuration.
- An agent in a distributed workflow does not run on the target machine.

Solution:

- In the DirX Identity Manager Expert View, select **Configuration → DirX Identity Servers**.
- In the relevant DirX Identity server object, click the **Agents** tab and check the installed agents.
- Start an update installation on the machine where the problem occurs. Go to the step in the installation where the agents are selected. The installation step shows the agents that are installed on this machine.
- Compare the installation list with the list in the DirX Identity server object.
- If something differs, perform an update installation on the relevant server and select the correct agents.

3.1.2. Master Setup and JRE on 32-bit Windows

Indication:

Installing the JRE from installation medium on a Windows machine with 32-bit architecture does not work using the related master setup menu item.

Reason:

The menu item is supported for 64-bit architecture only.

Solution:

Use Windows Explorer in order to navigate into the subfolder Installation\Windows\Oracle of the installation medium, and then select the suitable JRE to be installed.

3.1.3. Windows Installation Complete but Errors Occurred (Read-Only Files)

Indication:

The product installation (upgrade or update installation) ends with the following message:

"The installation of Atos DirX Identity is complete, but some errors occurred during the install. Please see the installation log for details."

Sample installation log:

```
...
Summary
---------
Installation: Successful with errors.
... SUCCESSES
0 WARNINGS
... NONFATAL ERRORS
0 FATAL ERRORS
...
Install File:             D:\Program Files\DirX
Identity\bin\libdirxapi.dll
                          Status: ERROR
                          Additional Notes: ERROR - Flexeraakt: Target
file was read only or did not have write permissions and was not
overwritten
...
Install File:             D:\Program Files\DirX
Identity\bin\libdirxapi_md.dll
                          Status: ERROR
                          Additional Notes: ERROR - Flexeraakt: Target
file was read only or did not have write permissions and was not
overwritten
...
Install File:             D:\Programme\DirX Identity\bin\metacp.exe
                          Status: ERROR
                          Additional Notes: ERROR - Flexeraakt: Target
file was read only or did not have write permissions and was not
overwritten
```

Reason:

Some files were read-only and have not been overwritten. As a consequence, said files are not up-to-date in your installation.

Solution:

Review the installation log and search for the string **ERROR** (case-sensitive). If the number of errors is 15 or higher, inform DirX Identity technical support about the problem. If related errors can be fixed by removing the read-only flag from said files or by similar actions regarding access right flags perform the indicated actions. Repeat the installation.

3.1.4. Installation Complete but Errors Occurred (Other Reasons)

Indication:

The product installation ends with the following message:

"The installation of Atos DirX Identity is complete, but some errors occurred during the install. Please see the installation log for details."

Sample installation log:

```
...
Summary
---------
Installation: Successful with errors.
... SUCCESSES
0 WARNINGS
... NONFATAL ERRORS
0 FATAL ERRORS
```

Reason:

Errors occurred during installation.The installation log file provides details about the errors occurred.

Solution:

Review the installation log file and search for the string **ERROR** (case-sensitive).If the error reasons are obvious from the related error messages fix them.If the indicated problems resulted in failures regarding creating or updating files or folder repeat the installation.If the problem persists, inform DirX Identity technical support about the problem.

3.2. Configuration Problems

This section describes problems that occur during DirX Identity configuration and how to solve them.

3.2.1. Admin or Password Not OK

Indication:

The message **Admin or password not ok** is displayed when the Directory Administrator account is tested.

Reason 1:

- (indication) the error occurs only on Windows Server machines where Active Directory is installed in parallel.
- When Active Directory and the Meta Directory Server use the same port (**389**), the account test will go to the Active Directory and the indicated error message results.

Solution 1:

- Reconfigure the LDAP server of the Meta Directory Server to another free port.
- Try the DirX Identity configuration again with the new port value.

Reason 2:

The Directory Administrator account contains an umlaut.

Solution 2:

- Change the LDAPConfiguration attributes LCCQ and LCCS to LATIN1:

  Run **dirxcp**

  Bind as **admin** via DAP and type the following commands

  modify */cp/***CN=LDAPConfiguration -rem LCCQ –add LCCQ=LATIN1**
  -modify* */cp/***CN=LDAPConfiguration -rem LCCS –add LCCS=LATIN1**

  where *cp* is the context prefix; for example **o=pqr**.

  If the message **Attribute or attribute value doesn't exist** is displayed, do not use the **–rem** option in the commands

- Configure DirX Identity
- Change the LDAPConfiguration attributes LCCQ and LCCS to UTF-8:

  Run **dirxcp**

  Bind as **admin** via DAP and type the following commands

  modify */cp/***CN=LDAPConfiguration -rem LCCQ –add LCCQ=UTF8**
  modify */cp/***CN=LDAPConfiguration -rem LCCS –add LCCS=UTF8**

3.2.2. Bind Error during Schema Configuration

Indications:

- The following error occurs during configuration of the DirX Identity schema: **BIND '/dxmC=DirXmetahub/cn=admin' failed: 'Invalid parameter passed as an argument.'**
- The error happens only with DirX.

Reason:

metacp reports a problem (for example, duplicate attribute definitions in the DirX abbreviation files).

Solution:

- You can obtain the specific error message if you start **metacp** with the same command by hand.
- Check the **stdout** message and resolve the problem.

3.2.3. Creation of DirX Identity C++-based Service Fails

Indication:

- The creation of the C++-based service fails during configuration.
- The error occurs only on Windows servers.

Reason:

An open **Services** or **Event Viewer** window prevents the creation of the service.

Solution:

- Close all Services and Event Viewer windows.
- Configure the C++-based Server again.

3.2.4. Long lasting configuration

Indication:

The configuration lasts a long time (20 min and more).

Reason:

If the directory contains 100,000 entries and more, the installation of the configuration database requires a long time.

Solution:

Be patient until the configuration has finished.

3.2.5. Initial Configuration Loops

Indication:

The Initial Configuration loops.

Reason:

The environment variable DIRX_IDM_PDU_TRACE is set and the directory already contains configuration data (for example from a previous aborted configuration).

Solution:

Reset the variable and start the Configuration Wizard again.

3.2.6. Configuration on Linux Virtualized on VMWare ESX

Indication:

When running the Configuration Wizard on RedHat Linux ES 5, virtualized on VMware ESX, some input fields, like password and path for key store or trust store are read-only what blocks completion of the configuration.

Reason:

Software problem during initial creation of the relevant input fields.

Solution:

The solution depends on the related configuration step:

- If you cannot enter a password in the step “Directory Server for Connectivity”, then click the input field for the port, then the input field for the password.
- If you have selected SSL in the step “Java-based Server” and cannot enter a password here, first un-check “Use SSL”, then check it again.
- If you cannot enter a trust store password in the step “Java-based Worker” first open the file-dialogue for “Trust Store File” and close it after suitable file selection.

3.2.7. Schema Setup Problems after Upgrading DirX

Indication:

DirX Identity does not start due to schema error problems indicated by the message **unknown abbreviation**.

Reason:

You have upgraded your old DirX installation to another DirX version.It was necessary to uninstall the old DirX version before installing the new one because the database of the new DirX version is incompatible with the old one.

DirX Identity uses project-specific abbreviation files (**dirxabbr-ext***) that are copied to the DirX installation subdirectory **$DIRX_INST_PATH/client/conf** when performing DirX Identity configuration.When uninstalling a version, DirX removes all files from its installation directory $DIRX_INST_PATH.

Solution:

Copy all **dirxabbr-ext*** files from the directory **$DIRX_METAHUB_INST_PATH/client/conf** to the directory **$DIRX_INST_PATH/client/conf** or re-run the Configuration Wizard to ensure that the files are copied.

3.3. Schema Installation Problems

This section handles schema installation problems that occur during DirX Identity Server or DirX Identity Schema Extension parts.

3.3.1. Invalid Parameter Passed

Indication:

The configuration fails on a DirX directory server during the schema configuration procedure.The corresponding message during the first call of **metacp** is:

BIND '/dxmC=DirXmetahub/cn=admin' failed: 'Invalid parameter passed as an argument.'

Reason:

A duplicate definition of an attribute abbreviation exists in the schema.

Solution:

- To find out which abbreviation is duplicated, start **metacp** interactively with the necessary scripts and parameters.This action results in a **stderr** output to the console that reports the problem, as follows:

  "Error: Duplicate Attribute abbreviation "*abbreviation*".- Error in schema file *file*"
- Correct the problem and restart the configuration procedure.

3.4. Uninstall and Unconfiguration Problems

This chapter handles uninstalling/unconfiguration problems that occur during DirX Identity uninstallation and unconfiguration.

3.4.1. Some Libraries Cannot be Removed

Indication:

When uninstalling DirX Identity on Windows platforms, some libraries, for example, **libdirxutil.dll**, cannot be removed.

Reason:

DirX Manager is open and uses the libraries due to the order specified in the PATH environment variable.

Solution:

Close DirX Manager.

3.4.2. Objects Remain after Unconfiguration

Indication:

After an unconfiguration, the root node **dxmc=DirXmetahub** together with the attribute DSET='' remains in the configuration database.

The error occurs only on DirX.

Reason:

You cannot delete the node **dxmc=DirXmetahub** via LDAP.

Solution:

To delete root node **dxmc=DirXmetahub**:

- Start **dirxadm** and enter the command:

  modify *ldap_root_dn* **-removeattr LNCO={/DXMC=DirXmetahub}**

  For example: `modify /O=pqr/CN=LdapRoot -removeattr LNCO={/DXMC=DirXmetahub}`

  Alternatively, you can use DirX Manager to delete **/DXMC=DirXmetahub** in the attribute **LDAP-Naming-Contexts**.
- Perform the **dirxadm** command:

  modify / -removeattr DADM={/DXMC=DirXmetahub/CN=admin}.

  Alternatively, you can use DirX Manager to delete **/DXMC=DirXmetahub/CN=admin** in the attribute **DirX-Administrators**

Now the node **/DXMC=DirXmetahub** is completely removed.

3.4.3. Unable to Remove Registry Key

Indication:

Message:

"Unable to remove registry key
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session

Manager\Environment"

Reason:

The installer could not remove the Identity specific parts of the PATH variable.

Solution:

You can ignore this message if you intend to reinstall the product.

Alternatively, delete the DirX Identity-specific parts of the PATH variable by hand.

3.4.4. Uninstallation Hangs

Indication:

Uninstallation hangs in 'Registry' section.

Reason:

The file **C:\Program Files\Zero G Registry\.com.zerog.registry.xml** is available due to a previously erroneous operation.

Solution:

- Delete the file **C:\Program Files\Zero G Registry\.com.zerog.registry.xml**
- Start the uninstallation again.

3.4.5. Uninstallation on 64-bit Windows

Indication:

Uninstall fails while trying to stop running DirX Identity services.

Reason:

This issue is related to 32-bit custom code during uninstallation.

Solution:

Stop all DirX Identity services, including the one for the ActiveMQ message broker manually before uninstalling.

3.5. Update and Upgrade Installation Problems

This section describes problems that relate to DirX and DirX Identity update and upgrade installations and how to solve them.

3.5.1. Overwritten Agent Batch Files

Indication:

Workflows no longer run after update / upgrade installation.

Reason:

During the DirX Identity installation the agent batch files in the bin directory of the installation path are overwritten. If you changed something in these files (for example more memory for the Java processes) this information is lost.

Solution:

To avoid this problem, you should copy the agent batch file before you perform the changes. To use the copied and changed files, create new agent objects that use the new agent batch files and set links from all relevant jobs to these agent objects in the Connectivity configuration database.

In a distributed environment be sure that all DirX Identity servers where this agents shall run have copies of the changed files.

3.5.2. Overwritten SSL Certificate Stores during Upgrade

Indication:

After an upgrade installation and configuration, the SSL messaging no longer runs. A possible indication can be that the self registration processes fail.

Reason:

The upgrade installation overwrote the SSL Certificate stores.

Solution:

Re-import the server certificates to *dxi_java_home***/lib/security/cacerts**

3.5.3. Overwritten cert8.db File

Indication:

SSL connections no longer work after upgrade or update installation and configuration.

Reason:

During an update or upgrade installation the cert8.db file is written again. If you used this default file for your certificates they are lost.

Solution:

To prevent such type of errors in the future, use another filename for your certificates and set the corresponding environment variable.

3.5.4. Windows Services Temporarily not Visible

Indication:

After an upgrade installation some or all of the Windows Identity services are not visible.

Solution:

Simply perform a reboot.

4. Problems with Provisioning

This section describes problems that relate to all provisioning services like policy execution, privilege resolution, consistency checks and approval workflows used with DirX Identity and how to solve them.

4.1. Approval does not Work

Indication:

Approval does not work.Entries with title UNDER CONSTRUCTION exist in the top level folder under **Provisioning → Approval Workflows → Monitoring**.
Note: this topic is only valid for the approval engine up to DirX Identity V7.0.

Reason:

One or more login or configuration parameters might be incorrect or the approval engine could be configured to another domain.

Solution:

Perform the following checks:

- Check the 'LDAP Parameters' section in the file **server.xml** in the folder:_ install_path_**\web\approvalEngine\WEB-INF**.
- Check especially root, host, port and user.
- If parameters are not correct, correct it and restart Tomcat.

If login still not works:

- Open the file **password.properties** in the folder:_ install_path_**\web\approvalEngine\WEB-INF**.
- Replace the encrypted password value behind 'ldap=' by the correct password of the technical user to log in (normally the DomainAdmin).
- Restart Tomcat.
- Check, that the file contains an encrypted value again (the server reads the clear text password and encrypts it immediately).

4.2. Changed Rules for Attributes

Indication:

You changed rules for attribute creation, for example to create attributes for a new or changed account.Now the attributes of many of these objects are not correctly built.

Solution:

Write a consistency rule in Java or JavaScript that adapts the values of all objects according

to your new rule.

4.3. Consistency Check Workflow is Aborted due to OutOfMemoryError

Indication:

A Consistency Check Workflow is aborted due to OutOfMemoryError.

Solution:

If a Consistency Check Workflow is aborted due to OutOfMemoryError perform the following steps to solve the problem:

- **Increase the maximum virtual memory for the ServiceAgent**:
  Edit the file **MetaRoleServiceAgent.bat** (on Windows systems) or **MetaRoleServiceAgent.sh** (on Linux systems) and increase the default setting **-Xmx512m** to a higher value.The maximum value depends on the system architecture (32bit/64bit) and OS, the system's total memory, and the memory requirements of the other processes being run on the system.On a 32bit Windows system, there is an upper limit of about 1500m through 1600m.
- **Reduce the setting of the Object Cache MRU Size (cache.mru.size):**
  Reducing the Object Cache MRU Size considerable (for example. to a value of 10) may help in such a situation.This will give the garbage collector access to objects that may be required in a later stage of the resolution process.So we gain stability at the cost of a lower performance, since those objects are restored from LDAP when requested again.

See "Using the Maintenance Workflows" → "Understanding the Tcl-based Maintenance Workflows" → "Consistency Check Workflow" → "Service Agent Configuration File" and "Consistency Check Workflow Optimization" in the *DirX Identity Application Development Guide* for details.

4.4. Enabling Deleted Users

Indication:

A user that is already in the TBDEL state is enabled again.All accounts are already in the DELETED state.When enabling the user again, account creation is not possible because the already deleted account is there.Other problems might also result from this situation.

Reason:

DirX Identity distinguishes clearly between the state TBDEL and DISABLED.The state DISABLED should be used for:

- Temporarily disabling a user.
- Setting a time period after the user has left the company to ensure that it is possible to enable the user again if necessary.

The state TBDEL is a final state that should not be used to enable the user again after some waiting time.

Solution:

If you 'redefine' these states in your project, you might run into a lot of problems.We strongly recommend using the DirX Identity states as originally intended.This strategy also guarantees upward compatibility.See the chapter "Managing States" in the *DirX Identity Provisioning Guide* for more information.

4.5. Incorrect Approval Behavior

Note: this issue is only valid for the approval engine supported with pre V8.0 versions.

Indication:

The approval process does not work correctly after configuration changes.

Reason:

When you modify an existing object description or add new ones, for example, by creating a new target system, please make sure to that the Approval Engine services read the changed data.

Solution:

The usual way to achieve this is to restart the appropriate Tomcat Web server(s).

4.6. Incorrect Privilege Resolution

Indication:

If a group contains more than one value for a privilege parameter, the privilege resolution does not assign this group.

Reason:

If the attribute is defined as single value, but keeps more than one attribute value, the privilege resolution takes only the first value and performs the resolution based on this data.

Solution:

Check for all parameters that the definition for the attribute in the Groups.xml file is set to multi value.If not, change it.

4.7. Invalid Query Reference

Indication:

The LDAP server returns the message

Invalid query reference; LDAP server is unwilling to perform

while DirX Identity-related services (privilege resolution, consistency check, policy execution) are working.

Reason:

If paging is used, each request for an additional page contains a cookie. Obviously consecutive page requests need longer than the configured paging timeout. The searches as well as the necessary calculation of for example attribute values consume too much time for a page of user entries.

Solution:

1. Check the LDAP audit of the server for long lasting searches.These are mainly caused by attributes that have not got an index.Check the filter of such searches whether all attributes are indexed correctly.Check also all other methods that are described in the section titled "Tuning the Provisioning System" in chapter titled "Managing the Provisioning System" in the *Provisioning Administration Guide*.
2. Check if you use long lasting JavaScript routines for attributes when creating an account.Try to use as many naming rules as possible (naming rules are handled directly in Java code which is definitely much faster than JavaScript).
3. Increase the paging timeout at LDAP server side (default is 5 minutes); for example, to 10 minutes.
4. Change the limits of the corresponding workflows.Decrease the **LDAP Page Size** stepwise to values like 100, 50, 10. Note that the corresponding parameter name in the policy execution workflow is **User LDAP Size**.
5. If all actions taken do not help, inform technical support about the problem.

4.8. Old Values in Membership Attributes

Indication:

The membership attributes still contain the old values if you changed the 'Referenced Property' attribute of a target system (see the 'Advanced' tab of a target system object).

Reason:

There are the following restrictions:

- The Service Agent adapts the membership attributes when this attribute is changed:
- The old value is deleted from the membership attributes.
- The new value is added to the membership attributes.
- If the cn or dn is used as 'Referenced Property' attribute, changes of the cn or dn attriute via "Move Object" or other pure LDAP operations do not result in membership changes because these operations are not handled by the services (they are not able to

detect the change).

- Nested groups are currently not supported.

Solution:

Respect the restrictions above.

4.9. Web Center or Java-based Server not Working Correctly

Indication:

You cannot login to the Web Center or your Java-based Server does not work correctly (errors during startup).

Reason:

The Web Center and / or the Java-based Server are not configured to the correct domain.

Solution:

1. Check the Web Center configuration in the file *install_path***web\webCenter\WEB-INF\web.xml**:

   Check the parameters in the <!-- Ldap parameters --> section, especially the **host**, **port**, **ssl**, **user**, **baseDN** and **anyone** values.

2. Check the Java-based Server configuration in the files *install_path***ids-j-***domain*-**S***n***bin\bindcredentials.xml** and *install_path***ids-j-***domain*-**S***n***bindprofiles\private\domain.xml**:

   Check the following parameters: **host**, **port**, **ssl, user**, **baseDN** and **domain** values.

3. If the parameters are not correct, you have the following options:
   - Run the Configuration (on Windows from **Start** → **DirX Identity V***x***.x** → **Configuration**).

     Set the options **Java-based Server Configuration** and **Web Center Configuration**.

     Set all relevant attributes especially the required **Domain** parameter and run the configuration.

     Your Web Center and your Java-based Server will be re-configured.
   - Edit the files by hand:
     - Shut down the Java-based Server and the Web Center (Tomcat).
     - Edit the above checked files and set the correct parameters.You must also edit the corresponding password files (see the parameter in these files).Set the passwords in clear text.The services will encrypt the values during startup.

       Start the Web Center (Tomcat) and the Java-based Server.

4.10. Cleanup Rule for Imported SharePoint Group Members

Indication:

A customer-defined validation cleanup rule with the purpose of deleting imported SharePoint group members from a group because those members shall not be members in DirX Identity does not work and instead logs **Imported member** "**cn=*_user_name_ ,cn=users,cn=_domain_" *is not affected by Accept/Cleanup rules**.

Reason:

The cleanup rule's search base and filter were defined to search for users, but the Policy Agent's rule processing coding expects account objects to be found with the defined search, not with user objects.For a SharePoint target system, the accounts are those in the related peer target system.

Solution:

Even though the members of SharePoint groups in Identity consist of user DNs, not account DNs, as is usually the case, the search base and filter in the cleanup rule must be defined to search for the accounts in the related peer target system, not for the users.

4.11. Inconsistent Group Membership

Indication:

The group membership attributes **dxrGroupMemberOK** and **dxrGroupMemberImported** hold the same value.

Reason:

If many realtime workflows for the same account or group are running in parallel, then there might be situations that one realtime workflow (on its way back to synchronize the Identity Store) already collects membership attribute values from the Connected System that the other realtime workflow just created.This situation results in creating members in the **dxrGroupMemberImported** attribute whereas the second realtime workflow created the same value in **dxrGroupMemeberOk**.

Solution:

The situation described above is not really critical. The next time when the realtime workflow synchronizes that account or group the membership attributes are updated correctly.

Nevertheless, in order to avoid confusion (because next update on the account or group may happen only in the far future) a consistency rule **ResolveInconsistentGroupMembership** is provided that will drop these inconsistencies.

This rule is executed by the workflow "Connectivity Configuration data / Workflows /

Default / Identity Store / ResolveInconsistentGroupMembership". We recommend you to create a schedule for that workflow (for example running the workflow once a day).

5. Problems with Connectivity

This section describes DirX Identity connectivity errors that can occur and the steps to take to solve the problem.

5.1. General Problems

This section describes problems that can occur with different agents and the steps to take to solve the problem.

5.1.1. Imported Groups Cannot Be Assigned

Indication:

After running an initial load or validation workflow in Tcl-based technology, the imported groups cannot be assigned to users.

Reason:

The Tcl-based workflows do not set the "User Assignment Possible" flag by default. In contrast, the Java-based workflows set this flag by default.

Solution:

Change the mapping of the Tcl-based workflows to set this flag.

5.1.2. Insufficient Memory for Java Agents

Indication:

Running a Java-based application can result in an "Insufficient memory" message.

Reason:

Each Java application starts up with a fixed maximum memory definition (if none is specified 64 MB are used by default). DirX Identity uses 256 MB as the default.

Solution:

If you run into this problem, adjust the corresponding parameter **-Xmx** of the Java application.

Specify an appropriate value for the parameter **-Xmx**. For the definition of this parameter, see the file **Xusage.txt**. (Search this file in the DirX Identity installation.)

5.1.3. SSL Flag at Service Objects for LDAP Connections

Indication:

The SSL flag in service objects is not used.

Reason:

The SSL flag is not used in service objects for LDAP connections.

Solution:

If you need to configure an SSL connection to an LDAP server, set the SSL flag in the corresponding bind profile and be sure that the secure port in the service object is set.

5.1.4. Workflow does not End

Indication:

Workflows hang and sometimes run into time out. A specific activity does not end correctly.

Reason:

A buffer overrun occurred either in **stdout** or **stderr**.DirX Identity can handle output on these two channels only up to 32 KB.If larger output is generated on these channels, communication between the agent controller and the agent is broken.A result is that the agent controller waits until timeout.Nevertheless the agent completed its tasks and ended correctly.

Solution:

The channels **stdout** and **stderr** are not designed to be used for a large amount of output data.Use regular files instead for this type of output.A common problem is often to use **puts** in Tcl to output some debug information to **stdout** or **stderr**.

To check that this is the real error cause, copy all files of the erroneous run from the status area back to the work area and start the agent by hand.Redirect the output to a file and check whether the file is too large (> 32 KB).

Rewrite your application so that it does not produce large amount of output to **stdout** or **stderr**.For meta controller applications, open a regular file and write this information to it.

5.2. Java-Based Workflow Problems

This section describes problems that can occur with Java-based workflows and the steps to take to solve the problem.

5.2.1. Attribute is not Synchronized even though Specified in Mapping

Indication:

You have added a mapping entry in a channel, for example a direct mapping from source attribute **displayName** to target attribute **description** in an account channel to an LDAP target system, but the attribute is not synchronized to the target system.

Reason:

The join engine can only perform the mapping if the source attribute **displayName** is contained in the selected attributes list of the corresponding Identity account channel. If **displayName** would have already been contained as target attribute in the mapping to Identity, which is true for most attributes, this error wouldn't have been occurred.

Solution:

To put the attribute **displayName** to the source selected attributes list, follow the link to the corresponding Identity account channel and add a dummy mapping - if no real mapping is wanted for that attribute - to **displayName** as target attribute, for example map a constant value to **displayName**, and set the flag **readOnly**.

5.2.2. Starting Transport Workflows on Java-based Server

Indication:

It is not possible to start transport workflows on a Java-based Server. The error message is:

ERROR: 'The first argument to the non-static Java function '...' is not a valid object reference.'
FATAL ERROR: 'Could not compile stylesheet'

Reason:

The TransformerFactory FEATURE_SECURE_PROCESSING flag enabled on a Java-based Server prevents the use of XSLT extension functions.

Setting of the flag is driven by presence of a custom security manager. Whenever it is set (System.setSecurityManager()), the flag is turned on and cannot be switched back. This "limitation" comes with Java 7.

Solution:

Start transport workflows from batch.

5.2.3. SharePoint Connector Problems

This section describes problems that relate to SharePoint Connector operation and how to solve them.

5.2.3.1. Content is not allowed in prolog.: 404 FILE NOT FOUND

Indication:

The logging of the SharePoint Connector, which is contained in the Java server log file if run inside a workflow, states that even though a connection to the SharePoint server was made successfully, the search on the specified site fails with the above error code.

Reason:

The reason for this error code is an incorrect server URL (property "Site URL" of the SharePoint Provisioning ClusterTarget System "Server Connection" tab).

Solution:

Depending on your Windows domain DNS configuration it is important that you specify the server URL in the appropriate form, which can be either a short server name form, like

http://shp_server1/sites/SiteCollection/SC_SiteA

or a fully qualified server name form, like

http://shp_server1.mydomain.com/sites/SiteCollection/SC_SiteA

or just a form without any server name, like

http://DXI_TestSiteCollection/SC_SiteA

You must try and find out which form fits with your environment.

5.2.3.2. Org.xml.sax.SAXException: Premature end of file

Indication:

The logging of the SharePoint Connector, which is contained in the Java-based server log file if run inside a workflow, states that even though a connection to the SharePoint server was built up successfully, the search on the specified site fails with the error code above.

Reason:

The reason for this error code is an incorrect password (property "bind account" of the SharePoint Provisioning ClusterTarget System "Server Connection" on a clustered SharePoint target system or the password of the Bind Profile of the Connected Directory on a non-clustered target system).

Solution:

Specify the correct password.

5.2.3.3. Java.net.SocketException: Connection reset

Indication:

The SharePoint Connector log, which is contained in the Java-based Server log file if run inside a workflow, states that the connection during a validation or synchronization workflow has been reset due to a java.net.SocketException and the workflow was aborted.

Reason:

The reason for this error is that the "Unlimited Strength Java Cryptography Extension Policy Files" JRE extension has been installed in the JRE the Java-based Server runs with.

Solution:

If JRE version 8 is used, the Unlimited Strength Java Cryptography Extension (JCE) package should not be installed.Only the limited JCE should be installed; this version is installed by

default when installing a JRE.If JRE version 7 is used, both limited and unlimited variants of JCE can be used.

5.3. Tcl-Based Workflow Problems

This section describes problems that can occur with Tcl-based workflows and the steps to take to solve the problem.

5.3.1. ADS Agent

This section describes problems that relate to ADS agent operation and how to solve them.

5.3.1.1. Secure Authentication in Bind Profiles

Indication:

When you set the **Use Secure Authentication** flag in the bind profile, you can set user passwords, but binding to the deleted objects container fails if the agent runs on a Windows NT machine.

Reason:

Due to a Microsoft bug in the ADSI Version on Windows NT, a bind to the deleted objects container fails when the **Use Secure Authentication** flag is set.

Solution:

If you want to be able to set user passwords and also get deleted objects, use different bind profiles for ADS Import and Export when running on Windows NT. Set the flag in the bind profile used for import (note that if you set this flag, the username must be specified in the form *domainname***username* on Windows NT) and do not set it in the bind profile used for export.

5.3.2. Exchange 5.5 Agent

This section describes problems that relate to Exchange 5.5 agent operation and how to solve them.

5.3.2.1. Stored Mailbox Resources

Indication:

When a mailbox is deleted, the stored resources are not deleted and become orphaned.

Reason:

Due to a Microsoft bug, the stored resources are not deleted if an Exchange 5.5 mailbox is deleted using the ADSI/LDAP API. They are only deleted using the DAPI API or the Exchange Administrator Tool.

Solution:

- To identify orphaned mailbox resources, run the consistency adjuster with the Exchange Administrator Tool. Select: **YourServer→Properties→Advanced→Consistency Adjuster.**
- Check all private information store checkboxes and the filter to all inconsistencies, then click **OK**.
- The Consistency Adjuster runs and recreates the orphaned objects under their original Recipients container. You can now delete them with the Exchange Administrator Tool. You can also recreate an orphaned object in its original context by creating the same deleted mailbox with the Exchange Agent and then deleting it with the Exchange Administrator Tool.

5.3.3. Meta Controller

This section describes problems that relate to meta controller operation.

5.3.3.1. Exit Code 4

Indication:

During workflow execution, a **metacp** job exits with exit code 4, and no further information or not enough information about the reason for the problem is visible in the report, trace, or process info file of the related status entry.

Reason:

There is a serious syntax error in a Tcl script involved in the job, for example, a missing opening brace in a script.

Solution 1:

Run the workflow again. If this action does not provide more information, try solution 2.

Solution 2:

- For each Tcl script in the related job ensure that the **Copy to Status Area** flag in the **File item** tab is checked.
- Run the workflow again.
- Copy the generated files from the status area into the work area (the directory: *workpath*/*workflowDisplayName*/*activityDisplayName*)
- Open an MS/DOS command prompt window (or a UNIX shell) in the work area directory and rerun the Tcl scripts:

  metacp
  metacp>**source control.tcl**

  error output will be displayed here

  metacp>**exit**

- Analyze the error output. If the error output is related to the DirX Identity default scripts rather than to your customizations, inform your DirX Identity support service about the problem. Otherwise, fix the problem in your scripts. Proceed with solution 3.

Solution 3:

Set **debug_scripts = 1** in the central control script, rerun the workflow and view the file *workpath/workflowDisplayName/activityDisplayName*/dump.txt*.

If the error output in this file is related to the DirX Identity default scripts rather than to your customizations, inform your DirX Identity support service about the problem. Otherwise, fix the problem in your scripts.

Don't forget to reset the **debug_scripts** switch in the central control script after you have fixed the problem.

Solution 4:

If none of methods just described provide enough information, you need to insert additional debug statements (like puts..., meta writetrace..., trace_out...) in order to find and fix the problem.

5.3.3.2. GUID Generation Fails

Indication:

During GUID generation (adding an entry) this message is displayed:

```
# {GUID not set by mapping => create it now !}
# {*** object cn=uid-a127359-6ecbe6a6-10499aaf3d5--7ff5,dxmC=uid-
a127359-6ecbe6a6-10499aaf3d5--7ff6,dxmC=uid-a127359-6ecbe6a6-
10499aaf3d5--7ff7,dxmC=uid-a127359-6ecbe6a6-10499aaf3d5--
7ff8,dxmC=Connected Directories,dxmC=DirXmetahub not found ***}
```

Reason:

The start value to generate the next GUID value is not set.

Solution:

Specify the attribute dxmActualGUIDvalue at the target connected directory to the required value. (Specify **1** if this is the first entry, otherwise specify the *highest number* **+ 1** of existing GUIDs in the directory.)

5.3.3.3. metacp and Additional LDAP Attribute Names

Indication:

During execution of **metacp** jobs in Tcl-based workflows, **metacp** does not recognize

additional attribute names if the number of attributes to be synchronized is greater than 64. In this case, LDAP attributes which have been defined in the schema so that they have not only a primary name but additional attribute names are not handled in a correct way.

Reason:

In order to tune performance, **metacp** simply requests all attributes instead of specifying longer attribute lists explicitly. The search result contains the primary LDAP attribute names which are not necessarily identical to related additional attribute names.

Solution:

Use primary LDAP attribute names (attribute configuration, mapping) in these jobs instead of additional attribute names. As a short-term workaround in a production environment, perform the following change in the file **init.metacp**: set the Tcl variable **_md_req_attr_limit** to the value **-1** instead of **64**. The file is located in *install_path***/bin** (Windows) or *install_path***/client/conf** (UNIX), respectively. Remember to check this setting again whenever you repeat the installation of the release or a related service pack.

5.3.3.4. metacp and Operational Attributes

Indication:

During execution of **metacp** jobs in Tcl-based workflows, **metacp** does not recognize operational attribute names if the number of attributes to be synchronized is greater than 64.

Reason:

In order to tune performance, **metacp** requests simply all attributes instead of specifying longer attribute lists explicitly. Therefore, the search result doesn't contain operational attributes.

Solution:

As a short-term workaround in a productive environment perform the following change in the file **init.metacp**: set the Tcl variable **_md_req_attr_limit** to the value **-1** instead of **64**. The file is located in *install_path***/bin** (Windows) or *install_path***/client/conf** (UNIX), respectively. Remember to check this setting again whenever you repeat the installation of the release or a related service pack.

5.3.3.5. metacp Hangs

Indication:

The Initial Configuration hangs in one of the steps:

- "Connectivity schema and data configuration"
- "Provisioning schema and data configuration"
- "Sample domain configuration"

- “Create customer domain ...”

or when running Tcl workflows in the MSS, the workflow hangs.

Reason:

When running **metacp** using a Tcl script, **metacp** hangs if the meta controller flushes messages either to **stdout** or **stderr** by means of a **fprintf** operation. Such an **fprintf** operation conflicts with the native Tcl I/O handling by means of **puts**. This situation appears if certain environment variables are set to enforce **metacp** to output trace information on **stderr**.

Solution:

Run **metacp** in a DOS box or UNIX shell and check whether **metacp** flushes messages to **stderr**. If true, then you should unset the environment variables. Check all the environment variables that start with prefix **DIRX_** and unset those that are not absolutely necessary.

5.3.3.6. Tcl Comment Problems

Indication:

Sometimes people find that Tcl behaves differently than they expect (often because of the way some other language acts in a similar situation). They then think this unexpected behavior is a bug. Probably the most common occurrence of this is in regard to comments.

Reason:

In Tcl, everything passed to the parser must have proper list structure, even comments (and comments are passed to the parser, unlike in some languages, where they are stripped at an earlier stage). Generally, this means you need to make sure your braces are evenly matched, even though they may be on a line that is commented out.

Solution:

- Unbalanced braces:
  For example, you have an **if** statement that tests a certain condition, but you want to try testing a different condition. You comment out the old condition and type a new **if** statement. This code will cause an error that there is a missing close-brace:

  ```
  ## WRONG
  if { $newflag } {
  # if { $oldflag } {
  puts hello
  }
  ```

  In this case, you have to balance the braces, for example:

```
## CORRECT
if { $newflag } {
# if { $oldflag } {
puts hello
# }
}
```

- Comment Continuation:
  Another interesting point about comments in Tcl is that the line continuation mechanism still applies, so:

```
# This is a comment line that ends with a backslash \
and this line is still part of the comment
```

5.3.3.7. Warnings are not Logged

Indication:

Warnings are not logged at all.

Reason:

Logging of warnings is dependent on the trace level **Variable trace**.

Solution:

Select a trace level that comprises **Variable trace**.

5.3.3.8. Workflows Run Too Slowly

Indication:

The meta controller activity, especially import workflows, seems to run very slowly compared with the amount of entries to be produced.

Reason:

A possible reason can be searches for attributes that are not indexed.

Solution:

- Check your filter condition for attributes that are not indexed.
- Set the indexes in the directory.

5.3.3.9. XML Parse Error

Indication:

A 'parse error' message is displayed when the meta controller (**metacp**) reads an XML file.

Reason:

The export parser has detected an incorrect XML file.

Solution:

Use an XML editor or viewer to check the file before you read it with the meta controller. For example, use Microsoft Internet Explorer.

5.3.4. Lotus Notes Agent

This section describes problems that relate to Lotus Notes agent operation and how to solve them.

5.3.4.1. Password Reading from File

Indication:

Lotus Notes / Domino Agent shall read the password automatically from a file.

Reason:

Administrative requirement.

Solution:

Perform this administrative step:

Specify the extension manager by extending the section **[Notes]** in the file **notes.ini**:

[Notes]
EXTMGR_ADDINS=nextpwd.dll

The location of **notes.ini** depends on the version of your Lotus Notes client:

- Version 4.6.3: the file is located in the Windows directory.
- Version 5.x and 6.x: the file is located in the Lotus Notes client directory.

5.3.5. ODBC Agent

This section describes problems that relate to ODBC agent operation and how to solve them.

5.3.5.1. Errors in Old ODBC Workflows

Indication:

ODBC Agent enhancements may cause errors in old ODBC workflows

Reason:

In older versions it was sometimes necessary to specify column specification using *tablename*.*columnname* format. Using abbreviations was not possible. The new version of the agent provides full support of abbreviations.

If you have used SQL columnnames identical to an abbreviation this might result in strange SQL error messages. The columname is interpreted as abbreviation and the long form of the abbreviation is inserted in the SQL statement. If you turn on SQL tracing you will see the generated SQL statement.

Solution:

Use abbreviations in your configuration.

Example:

a. Part of attribute configuration:

```
Abbr:AccountID Name:Accounts.ID Prefix:'AccountID:' Suffix:'\012'
Rec-Sep:'' Mrule:- Encryption:N
Abbr:GroupID Name:Groups.ID Prefix:'GroupID:' Suffix:'\012' Rec-
Sep:'' Mrule:- Encryption:N
Abbr:MembershipAccountID Name:Memberships.AccountID
Prefix:'MembershipAccountID:' Suffix:'\012' Rec-Sep:'' Mrule:-
Encryption:N
Abbr:MembershipGroupID Name:Memberships.GroupID
Prefix:'MembershipGroupID:' Suffix:'\012' Rec-Sep:'' Mrule:-
Encryption:N
Abbr:MembershipID Name:Memberships.ID Prefix:'MembershipID:'
Suffix:'\012' Rec-Sep:'' Mrule:- Encryption:N
```

b. The old from clause:

```
from (Groups LEFT JOIN Memberships ON
Memberships.GroupID=Groups.ID) LEFT JOIN
    Accounts ON Accounts.ID=Memberships.AccountID
```

worked fine in former versions. Now the above configured "from clause" results in the following SQL "from clause":

```
from (Groups LEFT JOIN Memberships ON
```

```
Memberships.Groups.ID=Groups.ID) LEFT JOIN
     Accounts ON Accounts.ID=Memberships.Accounts.ID
```

which results in an SQL Error: "too few parameters".

c. Change the "from clause" to:

```
from (Groups LEFT JOIN Memberships ON MembershipGroupID=Groups.ID)
LEFT JOIN
     Accounts ON Accounts.ID=MembershipAccountID
```

Here Groups.ID and Accounts.ID have still table.column format.

Alternatively you can use:

```
from (Groups LEFT JOIN Memberships ON MembershipGroupID=GroupID)
LEFT JOIN
     Accounts ON AccountID=MembershipAccountID
```

5.3.5.2. MS Access: Too Few Parameters

Indication:

The error occurs during export of data from the ODBC database. A ODBC Microsoft Access Driver is used. The following message is logged:

```
SQLExecDirect failure
state=07002
native error=-3010
error message=[Microsoft][ODBC Microsoft Access Driver] Too few
parameters. Expected 1.
```

Reason:

The SQL query requests a field that is not present in the database. The field is defined in the attribute configuration of the ODBC database object and used in the source selected attributes.

Solution:

Define the necessary field(s) in the ODBC database.

5.3.6. SAP R/3 HR Agent

This section describes problems that relate to SAP R/3 HR agent operation and how to solve them.

5.3.6.1. Extras / Copy Export

Indication:

Activating the menu action 'Extras/Copy Export' on an R/3 release 4.7 system results in a pop-up info box that has to be confirmed.

Reason:

The info states that the SAP R/3 HR agent uses an old but otherwise still supported R/3 download functionality. Nevertheless, the Copy Export function works.

Solution:

You can ignore this message. This is an R/3 issue.

5.3.7. SAP R/3 UM Agent

This section describes problems that relate to SAP R/3 UM agent operation and how to solve them.

5.3.7.1. Other Languages

Indication:

Languages other than English or German the export of the text description of profiles or roles in a CUA environment does not work correctly. The default workflow does not export this attribute.

Reason:

Incomplete implementation.

Solution:

Change the workflow to export this attribute if you work with English or German texts.

5.3.8. SAP EP UM Agent

This section describes problems that relate to SAP EP UM agent operation and how to solve them.

5.3.8.1. Setting User Passwords in SAP EP

Indication:

In DirX Identity 7.0B and 7.0B service packs, the handling for setting and modifying a user

password in SAP EP through the “Ident_SAPEPUM_Sync” workflow (setting a password on account creation) and the “setPassword in SAP EP UM” workflow (modifying a password) is not easy.If not handled correctly you can get the error message:
“**could not update user WRONG_PASSWORD**” in the java servers logfile.

Reason:

In SAP EP, a password can only be modified when the old password is also passed in the modify request.Therefore the setPassword workflow for a password modification passes the password value the Sync workflow has set when creating the ts account as old password.This is not easy because, in 7.0B and patches, the Sync workflows for all target systems use the account attribute “dxrPassword” to set the password on account creation in the target systems.On the other hand the setPassword workflows use the dxmPassword attribute of the account for password modification.Starting from version 7.0C, the Sync workflows also use the dxmPassword attribute.

Solution:

When an account is created for a user in Identity through for example an assigned privilege, ensure that the dxrPassword, which is calculated through a java script on account creation, has the same value or is set afterwards to the same value as the dxmPassword attribute of account and user is set to.Then start the Sync workflow which creates the account in SAP EP with the password in dxrPassword.If then the user password is requested to be changed, for example by the WebCenter, an event is generated with the old password taken from dxmPassword of the user - matching the password now existing in SAP EP - and the newly requested password, which results in a correct update of the users password in SAP EP.

5.4. Windows Password Listener Problems

This section describes Windows Password Listener errors that can occur and the steps to take to solve the problem.

5.4.1. User Data Empty Error Message in Event Viewer

Indication:

In rare situations, the Event Viewer can show this message:

One or all parts of the user data are empty.

Reason:

This error situation is temporary. The incomplete file is resolved during the next processing cycle of the Windows Password Listener service.

Solution:

Ignore this type of error.

5.4.2. Service not Installed

Indication:

In rare situations, the service could not be installed.

Reason:

Not known.

Solution:

Use this command to repeat the installation:

dxmWinPwListener.exe -install

6. Problems with DirX Identity Manager

This section describes DirX Identity Manager problems that can occur and the steps to solve them.

6.1. General Issues

This section discusses general problems that can occur with the DirX Identity Manager.

6.1.1. Abbreviation Field Change Not Recognized

Indication:

Capitalization changes in the abbreviation field of a topic in the attribute configuration are not recognized in the generated mapping.

Reason:

When you add a new attribute in the attribute configuration file with an incorrect capitalization in the abbreviation field, then add this field to the selected attribute list and use it in the mapping, the mapping will use the wrong capitalized abbreviation in the generated mapping script. When you correct the capitalization, the mapping does not recognize the correction.

Solution:

When you detect an incorrect capitalization in the mapping, you should change to another attribute, change to the original one and save the mapping. The generated script should now contain the correct capitalization.

6.1.2. Copied Report does not Work

Indication:

A copied report does not work.

Reason:

All report templates contain relative reference links in the **Format** tab to common templates, for example "_Shared Report Routines". If you copy a report to the Customer Extensions folder, you must copy the _Shared Report Routines, too.

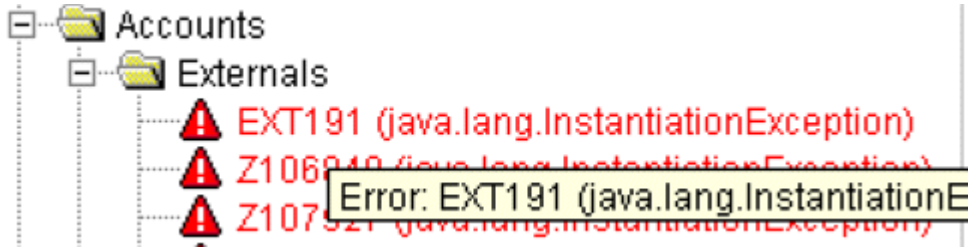
Solution:

Copy the common template, for example _Shared Report Routines, too. If necessary, adapt the links accordingly.

6.1.3. Instantiation Exception

Indication:

Objects or folders in the Identity Manager are shown in red together with the error message java.lang.InstantiationException:



In the Identity Manager log, you will find parsing error messages like:

```
DBG(STG100): parsing file
'storage://DirXmetaRole/cn=TSAccount.xml,cn=Object

Descriptions,cn=default,cn=TargetSystems,cn=Configuration,cn=ZFMD?con
tent=dxrObjDesc':
DBG(STG100): local var
TSdefaultDN=cn=Default,cn=TargetSystems,cn=Configuration,cn=ZFMD
DBG(STG100): local var
TSdefaultDN=cn=Default,cn=TargetSystems,cn=Configuration,cn=ZFMD
DBG(STG100): Configured SvcTSAccount
DBG(STG100): [Fatal Error] :6:1: Content is not allowed in prolog.
ERR(STG800): **** Parsing error: org.xml.sax.SAXParseException:
Content is not allowed in prolog.
DBG(STG100): null at line 6 column 1
```

Reason:

The related object description is erroneous.

Solution:

Correct the object description (mostly incorrect XML formatting). In the log file, you will find the object description that causes the error.

6.1.4. java.lang.NullPointerException

Indication:

This message can occur in many situations. Consequently, there is no well-defined solution that can be described here.

Reason:

The LDAP server might be configured to a specific size limit (maximum number of records

to be retrieved during a search operation). In this case, DirX Identity Manager can obtain messages that indicate severe errors (the database might become inconsistent!). For DirX Extranet, the size limit is set to 2048, which might be too low.

Solution:

We recommend setting the size limit to 10000 or higher.

6.1.5. java.lang.OutOfMemoryError

Indication:

This message can occur in many situations.

Reason:

The Java engine memory limit has been exceeded.

Solution:

Perform the following steps to increase the memory size for the Java engine:

- Stop DirX Identity Manager.
- Open the file **run.bat** in the directory *install_path***gui\bin** with a text editor.
- Set the memory limit in the line that starts with 'start javaw' to a higher value, for example:

```
start javaw -Xmx256m …
```

- Save the file.
- Restart DirX Identity Manager.

Here is an excerpt from the Java documentation that describes -Xmx usage:

-Xmx*n*

Specify the maximum size, in bytes, of the memory allocation pool. This value must a multiple of 1024 greater than 2MB. Append the letter k or K to indicate kilobytes, or m or M to indicate megabytes. The default value is 64MB. Examples:

-Xmx83886080
-Xmx81920k
-Xmx80m

6.1.6. Reset in Sub-Dialogs of Wizards

Indication:

If you open a sub dialog in a wizard that was shown as wizard step as well and you changed values during this wizard step, pressing Reset will reset your changes.

Reason:

Malfunction of software.

Solution:

Press OK instead and it will work fine.

6.1.7. Target System Report does not Work

Indication:

A target system report does not work.

Reason:

You can call target system specific reports only from the node **Target Systems** or from sub nodes.

Solution:

Call the report from a correct location.

6.1.8. UTF-8 as Default Character Set

Indication:

After import of LDIF files, some special characters are not correctly displayed.

Reason:

With DirX Identity V7.0 the default character set for all import and export operation of LDIF files has been adapted to the standard definition. Instead of Latin1 we use now UTF-8.

Solution:

Convert your file from Latin1 format to UTF-8 before performing the import operation.

6.1.9. Inconsistent Mapping in Tcl-based Workflows

Indication:

Arguments are displayed as inconsistent.attributename in pink color.

Reason:

Often the corresponding mapping items refer to a non-existent channel. This may occur after you have linked the job with another channel. All mapping items created before are referencing the old channel. All mapping items created afterward refer to the new one. When the old channel is now deleted the old mapping items are now inconsistent. Also, if you export your workflow and import it on another machine the old channel may not exist there.

Solution:

You must edit your mapping (on the source side if you intend to copy it to another machine) and select all arguments again. Here all attributes of the actual channels are offered. So it could happen that attributes from the old channel don’t appear here.

6.1.10. Wrong Order in Tree after Changing dxmDisplayName in Data View

Indication:

After changing the dxmDisplayName in the Data View and then returning to the Connectivity View, the tree resorted.

Reason:

Unknown.

Solution:

Perform a **Reload Object Descriptors** operation. The tree elements will then appear in the correct order.

In general, changes made using the Data View are not automatically displayed in the Connectivity or Provisioning View without any action.

6.1.11. Displaying the LDAP Attribute Displayname

Indication:

Identity Manager handles the attribute "displayname" in a special way. When requesting **displayname** the value of the attribute defined by "displayattribute" is displayed. In the following TSAccount.xml snippet **cn** is defined as display attribute and therefore the cn value will be displayed.

```
<definition
     cluster="$(./../../..)"
     targetsystem="$(./../../..)"
     class="siemens.dxr.service.nodes.SvcTSAccount"
     namingattribute="cn"
     displayattribute="cn"
     label="TSAccount"
     icon="icons/tsaccount.gif"
     candelete="true"
     canrename="true"
     cancopy="true"
     haschildren="false"
```

```
     helpcontext="tsaccount"
     displayname="Account"
     filterobjectclass="dxrTargetSystemAccount"
     accesscontrol="true" />
```

Solution:

If you want to display the real LDAP displayname define a pseudo property ($myDisplay) which returns the value of the LDAP attribute; for example the snippet TSAccount.xml:

```
<propertysheet>
   <propertypage name="displayname" insertafter="AccountGeneral"

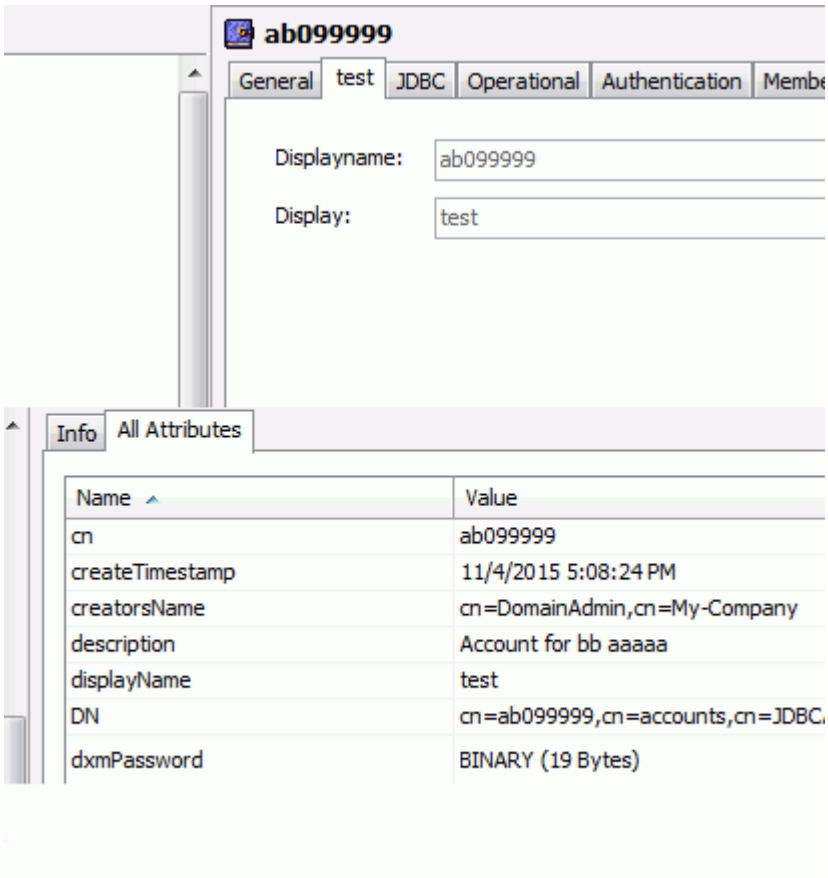
class="siemens.dxr.manager.nodes.customizer.GenericPropertyPage"
                 title="test"
                 layout="displayname, $myDisplay"
                 helpcontext="mr_ts_account_tsspecific"/>

</propertysheet>

<!-- JDBC specific properties -->
   <properties>
        <property name="displayName" type="java.lang.String"
label="Displayname" multivalue="false" mandatory="false"/>

        <property name="$myDisplayName"
                label="Display"
                                      >
                <script name="getProperty" return="value">
                           <![CDATA[
                                var obj = scriptContext.getObject();
                                var value =
obj.getProperty("displayname");
                               ]]>
                </script>
        </property>
```

In the sample **displayname** displays the value of the attribute **cn** (ab099999) and **myDisplayName** displays the value of the LDAP attribute **displayname** (test):



6.2. Monitor View

This section describes problems and issues that relate to using the DirX Identity Manager monitor view.

6.2.1. Duplicate Status Entries

Indication:

You start a workflow once interactively in the DirX Identity Manager.Afterwards you detect two status entries from this workflow with a slight delay (a few seconds).

Reason:

First you installed a DirX Identity Server on a configuration database. Then you installed a second DirX Identity Server on another machine but with the same configuration database.

This results in two **main** servers, which confuses the system.

Solution:

In this case, you must set up the system again from scratch:

- Save any work you did in this database (for example scenarios, workflows, jobs...) by exporting this information.
- Uninstall both, **main** Server installations and all Agent Installations.

- Install all servers again. Be sure that you set up only one **main** server (you select **Server** from the installation options). Do not select the **Server** option for the agent servers.

6.2.2. National Characters Are Not Displayed

Indication:

National characters are not displayed correctly when viewing trace or report files from the meta controller (**metacp)**.

Reason:

The meta controller always uses UTF-8 code to write trace and report files.

Solution:

Configure an editor that can display UTF-8 characters for viewing files. For example, you can use the freely available **lister** program.

6.2.3. No HTML Viewer

Indication:

HTML files (suffix **.htm**) cannot be viewed in a status entry.

Reason:

- Reason 1: No viewer is defined.
- Reason 2: The defined viewer cannot be found during execution because the operating system's path variable does not contain the location of the viewer.

Solution:

- Solution 1: Define a viewer in the **dxi.cfg** file. By default, the Internet Explorer is defined (should work on Windows platforms). Choose another one.
- Solution 2: Define either an absolute path to your viewer in the **dxi.cfg** file or extend the operating system path variable with the correct path.

6.2.4. No File Viewer for UNIX

Indication:

Files cannot be viewed in a status entry.

Reason:

The defined viewer cannot be found during execution because because it does not exist.

Solution:

Define the properties **file.editor** and **clipboard.viewer** in the **dxi.cfg** file so that the

properties are of the form **sh -c "*editor* *'$file'"** where *editor* is an absolute path to your viewer. The default editor is **/usr/bin/gedit** (Linux). Choose another one.

6.2.5. No Statistics When Compression Mode is On

Indication:

Statistics are missing for a workflow status entry.

Reason:

If the compression mode setting for a workflow is different from "Detailed" in the workflow entry or "None" in the central configuration object (which is the default) the statistics info for the workflow status object is not available.It is only available for the activity status objects.

Solution:

Choose another compression mode.

6.3. Connectivity View

This section describes problems that relate to DirX Identity Manager's Connectivity view and how to solve them.

6.3.1. Cannot Set Notify on Modify

Indication:

In the **Import** Tab of a Realtime Workflow Channel inside the **Notify** boarder you cannot set the **On Modify** flag.After save it stays empty as before.

Reason:

The **notifyOnModify** attribute in the **dxmImport** LDAP attribute is missing. You can check this in the Data view. For example, the content of **dxmImport** looks like:

```
<import createDespiteOfMultipleMatch="false"
            idInAddRequest="true" notifyOnAdd="false"
notifyOnDelete="false"/>
```

Solution:

In Data View extend the value with **notifyOnModify** attribute.For example:

```
<import createDespiteOfMultipleMatch="false"
            idInAddRequest="true" notifyOnAdd="false"
```

```
notifyOnDelete="false" notifyOnModify="false"/>
```

6.4. Provisioning View

This section describes problems that relate to DirX Identity Manager's Provisioning view and how to solve them.

6.4.1. Target System Creation Fails

Indication:

Target system creation with the target system creation wizard fails.

Reason:

You tried to create a target system that requires a custom schema extension.

Solution:

- Delete the partially created target system (both within Connectivity and Provisioning views).
- Perform the schema extension.
- Create the target system again.

6.5. Workflow Runtime Window

This section describes problems related to the workflow runtime window and how to solve them.

6.5.1. SHM9550

Indication:

Workflow executions fail.ProcessInfo and/or trace files show error message SHM9550: timeout to get semaphore (in dxmGetShm).

Reason:

DirX Identity Server is in operation with an encryption mode that is different from NONE, and heavy system load causes some workflows to fail with this error message.

Solution:

Increment the **KeyGet** timeout of the related DirX Identity server using DirX Identity Manager (Expert View: Configuration/DirX Identity Servers/C++-based Servers/*Server object*).

6.5.2. Refresh Takes Too Much Time

Indication:

Automatic Refresh in the Monitor View takes too much time.

Reason:

When DirX Identity has produced a lot of status entries, the automatic refresh needs some time. Changing from the Monitor View to other views and back to it refreshes the view even if nothing has changed.

Solution:

Turn off the automatic refresh:

1. Open the file *install_path***Gui\bin\dxi.cfg**
2. Change the value **monitorview.refresh** to **off**.
3. Save the file and restart DirX Identity Manager.

6.5.3. Wait for ACK of Create Instance Error

Indication:

Workflow runtime window hangs at 'Wait for ACK of create instance'.

Reason:

The DirX Identity 7.0 service is not started.

Solution:

- Check the log files and/or the event log.
- Start the DirX Identity service and try again.

7. Problems with Web Applications

This section lists common problems related to DirX Identity web applications deployed into a Tomcat server, like the Business User Interface or Web Center, and shows how to solve them.

7.1. Access to an Application Fails With HTTP Status 404

Indication:

After having deployed a DirX Identity web application into Tomcat, any attempt to access the application fails with HTTP status code 404 - Not Found.

Tomcat log file **catalina**.*date*.**log** reports an error message like

SEVERE [main] org.apache.catalina.startup.HostConfig.deployDescriptor Error deploying deployment descriptor [...*path of deployment descriptor*...]

java.lang.IllegalStateException: Error starting child

...

Caused by: java.lang.IllegalArgumentException: The main resource set specified [...*path of web application folder*...] is not valid

Reason:

The path of the web application folder specified in the deployment descriptor file is incorrect, or not accessible by the log on account for the Tomcat service.

Solution:

Make sure that the path of the web application folder specified in the deployment descriptor is correct.

Make sure that the web application folder and the files in folder are accessible by the log on account for the Tomcat service.

8. Problems with Web Center

This section lists common problems that related to Web Center, and shows how to solve them.



After any update to a Web Center configuration or password file, or to the DirX Identity connectivity or provisioning configuration, you must restart Tomcat for the changes to take in effect.

8.1. Broken Pages after Session Upgrade

Indication:

After upgrading to a new Web Center release, the browser displays broken Web Center pages or Javascript error messages.

Reason:

The browser has still cached pre-upgrade versions of some Web Center resources like static HTML pages and Javascript source files, and does not automatically update its cache.

Solution:

Have each user delete his browser cache:

- Internet Explorer: Tools/Internet Options/General/Delete Files/Delete all offline content.
- Firefox: Tools/Clear Private Data/Cache.

Or make Web Center available under a different HTTP context path than before.This way, pre-upgrade cache data does not conflict with the new release.The context path is identical to the name of the context descriptor file (without extension).

8.2. Broken Pages Due to Missing X Display Server

Indication:

When running under Tomcat on UNIX or Linux, Web Center pages are occasionally broken.Tomcat's log file contains an error message like "Can't connect to X11 window server using '0:0' as the value of the display".

Reason:

Some Web Center functionality depends on features implemented in the **java.awt** package.On UNIX or Linux systems, the package tries to connect to an X display server.The problem shows up if no X display server is available.

Solution:

Since access to an X display server is not really required we recommend to run Tomcat in

headless mode by setting the Java system property **java.awt.headless** to true, for example in file **TOMCAT_HOME/bin/catalina.sh**:

JAVA_OPTS=-Djava.awt.headless=true

8.3. CAPICOM Error Message with Enabled Client Signature

Indication:

You have enabled client signature.You want to perform an operation for example assigning a privilege to a user.When clicking the button for saving the data, you get the follwing error message:

This application requires a newer version of CAPICOM.

Reason:

You have not installed the correct CAPICOM version.

Solution:

Perform the steps described in "Setting up Client Signature" in "Setting Up Audit Trail" in chapter "Managing Auditing" in the *DirX Identity Provisioning Administration Guide*.

8.4. Change/Reset Password Failure

Indication:

Attempts to change or reset user passwords fail.

Reason:

The message server is down.

Solution:

Start the message server.

Reason:

Message server and Tomcat server run on different hosts, and the message server port is blocked by a firewall.

Solution:

Unblock the port for access from the Tomcat server host.

8.5. Display Privileged Account Password Failure

Indication:

Web Center does not display passwords of privileged accounts.

Reason:

Password encryption is enabled, but the pin needed to decrypt passwords is wrong or missing in file **WEB-INF/password.properties**.

Solution:

Add the clear text pin (with key **pin**) to the file, or replace the current one.

8.6. Error Messages when Approving Request Workflow Tasks

Indication:

When approving assignments with start or end date in the detailed representation, Web Center displays an error message that the related request could not be processed.

Reason:

Software problem.

Solution:

As a workaround for this problem perform the following steps:

- Check the tasks in your task list.
- Select **approve** from the context-sensitive menu.
- Perform your accept and/or reject actions from the subsequent pages.

8.7. Incorrect Login Link

Indication:

The link in default request workflows can contain an incorrect HTTP address.

Reason:

Per default there is either a fixed text configured (for example: http://localhost:8080/webCenter) or a dynamic definition delivers an incorrect value (for example ${workflow.approvalLink}).

Solution:

Set up the configuration in the request workflow section correctly.Using several Web Center applications results in different HTTP addresses.Set up different links as nationalization items for the respective messages and use them in the corresponding mail texts.

8.8. Login Fails for Any User

Indication:

Attemps to log in to Web Center fail for any user.

Reason:

A login configuration parameter is wrong.

Solution:

Correct the directory access configuration parameters in the deployment descriptor **WEB-INF/web.xml**, especially **host**, **port**, **ssl**, **user** and **baseDN**.

Reason:

The directory server is down.

Solution:

Start the directory server.

Reason:

Directory server and Tomcat server run on different hosts, and access to the directory server port is blocked by a firewall.

Solution:

Unblock the port for access from the Tomcat server host.

Reason:

Wrong or missing technical user password in file **WEB-INF/password.properties**.

Solution:

Add the clear text password (with key **ldap**) to the file, or replace the current one.

8.9. Login Fails for a Specific User

Indication:

Attempts to log in to Web Center fail for a specific user while other users can log in.

Reason:

The user entry is not enabled.

Solution:

Open the user entry in the data view of the DirX Identity Manager, and set its attribute **dxrState** to **ENABLED**.

Reason:

The user entry doesn't have a password (attribute **userPassword**).

Solution:

Open Web Center and reset the user's password, or open the user entry in the data view of the DirX Identity Manager and assign a new password.

Reason:

The user enters an incorrect password.

Solution:

Reset the user's password as in the previous case.

8.10. Request Workflow Access Failure

Indication:

All attempts to perform request workflow related tasks fail.

Reason:

A login configuration parameter is incorrect.

Solution:

Correct the request workflow configuration parameters in the deployment descriptor **WEB-INF/web.xml**, especially **endpoint** and, in case of connecting to the Java-based server via SSL, **keystoreName** and **keyAlias**.

Reason:

The Java-based server is down.

Solution:

Start the Java-based server.

Reason:

Java-based server and Tomcat server run on different hosts, and the port of the Java-based server is blocked by a firewall.

Solution:

Unblock the port for access from the Tomcat server host.

Reason:

In case of connecting to the Java-based server via SSL: Wrong or missing passwords in file **WEB-INF/password.properties**.

Solution:

Add the clear text passwords (with key **webcenterKeystore** and **webcenterKey**) to the file, or replace the current ones.

8.11. Request Workflow Access Failure in Case of Single Sign-On

Indication:

All attempts to perform request workflow related tasks fail with error message "You cannot access request workflows in your current session since you have authenticated against Web Center without a password but the workflow service is not set up for single sign-on."

Reason:

Web Center authentication is setup for single sign-on but the connection between Web Center and the request workflow server is not.

Solution:

See the section "Setting up Single Sign-On to the Request Workflow Server" in the *DirX Identity Web Center Reference* for details.

8.12. Session Timeout

Indication:

Web Center displays the login page or a start page instead of the requested one, along with a message that the requested session has expired..

Reason:

The request was sent with an invalid session cookie.

Web Center sessions do not live forever.They may time out due to inactivity, and do not survive a restart of the web server.So when a client sends an HTTP request to Web Center, the session referred to in the request’s session cookie may no longer exist.

Solution:

There’s no way to entirely avoid that problem.You can increase the session time out in Web

Center's deployment descriptor **WEB-INF/web.xml**, but that may result in other problems.

8.13. Size Limit Exceeded

Indication:

When searching for objects you get the message "Size limit exceeded.Please use stronger search criteria".In fact the result is only a few entries or may be no entry at all.The defined **Search Size Limit** of your domain configuration entry is much higher.

Reason:

Searches for objects are filtered in two steps:

1. The first filter is used to search for the objects in the LDAP database.
2. A second filtering is done on the retrieved search result in memory.

The search limit is reached during the search with the first filter in the LDAP database, which causes the message. The second filter removes a part of the entries from the retrieved list. This is dependent on the defined filters in the access policies of this user.

Access policies are XACML compliant.It is not always possible to convert an XACML filter to one LDAP filter.Thus filtering must sometimes be done in two steps, which causes this effect.

Solution:

The best way to avoid the problem is to use only access policy filters that can be mapped to LDAP filters.

Do not set the Search Size Limit of your domain to a value that's too low.Try to find a value that works for most cases.

The user must refine his search filter to overcome the problem of the first search.Add a hint on all web pages that allow searching for objects to explain the user that he must refine the search.

8.14. Too Many Sessions

Indication:

A user attempts to log on to the system and the procedure fails.

Reason:

Too many users have currently logged on to the system.These may be just a few, showing that the system is configured improperly.The reason is that there a too many sessions open which are unused and were not closed yet by the server.

Solution:

The solution is two-fold:

1. Reduce the "timeout" value for the HTTP session; that is, the time period after which the session is closed and deleted automatically if it was not used within that period:
   - Go to the Tomcat installation directory and open the file **conf/web.xml**.
   - Look up the section

```
<!-- ================= Default Session Configuration
============== -->
<! You can set the default session timeout (in minutes) for all
newly >
<!-- created sessions by modifying the value below. -->
<session-config>
<session-timeout>30</session-timeout>
</session-config>
```

   - Change the timeout value to the recommended value of "5" (minutes).
   - Save the file and restart Tomcat.
2. Increase the memory pool for Tomcat.
   a. Microsoft Windows
      - Go to the Start menu and execute the "Configure Tomcat" command in the Tomcat's start menu.
      - Open the "Java" tab and set appropriate values for "Initial memory pool" and "Maximum memory pool".Recommended values are "256" and "512" (MB).
   b. UNIX/Linux
      - Go to the Tomcat installation directory and edit the **bin/catalina.sh** file.
      - To the **TOMCAT_OPTS** environment variable, add appropriate values for the memory settings: -Xms256M -Xmx512M.

8.15. Wrong Role Parameter Display

Indication:

A role parameter value is not displayed correctly.Assignment of another role parameter value is not possible.

Reason:

"!AS!" is per default used as attribute separator and "!NI!" as parameter value separator.If you use the parameter value AS or NI, these problems can occur.

Solution:

Set other separators in the renderers-config.xml file (for example YY and ZZ):

```
<renderer id="parameters"
type="com.siemens.webMgr.identityAPI.Parameter[]"
...>
...
<renderer-property name="attrSeparator" value="!YY!"/>
...
</renderer>
...
<renderer id="paramsValuesAndMore"
...>
...
<renderer-property name="attrSeparator" value="!ZZ!"/>
...
</renderer>
```

9. Problems with Web Admin

This section describes problems that relate to the Web Admin application used with DirX Identity and how to solve them.

9.1. No Connection to Java-based Identity Server

Indication:

You cannot connect with Web Admin to the Java-based Server even if the Java-based Server is running.

Reason:

There is no thread left in the integrated Tomcat (part of Java-based Server).

Solution:

Increase the number of threads for the integrated Tomcat:

- Log in with DirX Identity Manager to the **Connectivity** view group.
- Open **Configuration → DirX Identity Servers → Java Servers →** *Your_Server*.
- Select the **Limits** tab.
- Increase the maximum number of threads for Tomcat.
- Restart the Java-based Server.
- Check that Web Admin works now.
- If not, repeat the procedure.

9.2. Node for C++-based Identity Server is not Displayed

Indication:

The node for the C++-based Server is not displayed.

Reason:

The connector server connection to the C++-based Server is not working.

Solution 1:

Enable the connector server connection:

- Edit the file *install_path***/server/conf/dxmmsssvr.ini**.
- Set the switch **cconnserver** to **1**.
- The resulting section is:

; optional: start SOAP SPML connector service (requires a configured port)
cconnserver=1

- Restart the C++-based Server.

Solution 2:

- Check that the port **5315** (or the configured port) is blocked by a firewall.
- Open the firewall if this is the case.

Solution 3:

Test the connection:

- Enter the address "http://localhost:5315/" in any web browser.
- The returned SOAP answer should be:

  <SOAP-ENV:Envelope>
  <SOAP-ENV:Body id="_0">
  <SOAP-ENV:Fault>
  <faultcode>SOAP-ENV:Client</faultcode>
  <faultstring>Server custom error</faultstring> <SOAP-ENV:Code/> <SOAP-ENV:Detail> Error receiving the JMX request data, msg: gSOAP error while receiving data, socket = 796, errcode = 14 </SOAP-ENV:Detail> </SOAP-ENV:Fault> </SOAP-ENV:Body> </SOAP-ENV:Envelope>

- If this is not the answer, check the C++-based Server log files for error messages.

10. Problems with Web Services (SOA)

This section describes problems that relate to the Web Service applications used with DirX Identity and how to solve them.

10.1. Exception NoClassDefFoundError when Running Web Service Clients

Indication:

Running the sample client (described in section “Runtime Operations” of the Web Services manual) fails with errors of the form:

FTL(PSFPTPC002): Exception java.lang.NoClassDefFoundError in provisioning request *Request* for requestID *request*: org/apache/commons/codec/binary/Base64

Reason:

You are probably using a customized copy of the sample client of DirX Identity V8.0B, and you have updated your runtime environment with the jar files of the sample client shipped with your installation.This problem cannot occur when using the sample client as shipped with your installation and installed in *install_path***/provisioningServices**.

Solution:

Ensure that the classpath in your runtime environment is computed so that the file **commons-codec.jar** is on the classpath.For details about suitable classpath computation, see *DirX Identity Migration Guide*, section “Manual migration”, subsection “General Aspects”, subsection “Web Services Clients”.

10.2. Memory Problems with Web Services

Indication:

Running web services on the embedded Tomcat of the Java-based Server can cause memory problems.

Reason:

Up to now, web service clients cannot be informed about low memory conditions.

Solution:

In a production environment, always run web services on a separate Tomcat installation. This strategy prevents memory problems in the Java-based Server.

11. Problems with Request Workflows

This section describes problems related to request workflows.

11.1. Request Workflow Uses the Wrong Initiator

Indication:

The workflow initiator is not set correctly when you have implemented your own "ApplyChanges" job class and another request workflow is started by that job class.

Reason:

You have not set the workflow initiator explicitly in your job class.

Solution:

You must provide the workflow initiator in the SvcSession object. You need to integrate the following snippet into your job class (see the sample Job "SampleJobStartNewRequestWorkflow", too):

```
protected void run(TaskContext taskCtx,
                   Map modifications,
                   SvcSession session,
                   Order subject,
                   Order[] resources,
                   Map parameters) throws Exception {
...
     Order request = (resources != null && resources.length != 0)
                                      ? resources[0] : subject;
      if (request != null) {
             // read creator from order and set initiator for new
             // request workflow (that is
             // initiated e.g. when processing
             // "SvcUsser.chackAndSave()" and attribute policies are
             // active).
              session.setInitiator(request.getCreator());
      }
...
}
```

12. Problems with Identity Services

This section describes Identity server-related problems.

12.1. Starting and Stopping Services

You may need to take this step after

- Reconfiguring parameters in parts of the configuration database
- Changing the password or other parameters in the **dxmmsssvr.ini**, **idmsvc.ini**, **runServer.sh**, **ssl/password.properties** or **bindcredentials.xml** file
- Errors

Message Broker

For the message broker, stop the services in the following sequence (use the **Services** utility on Windows or the following commands on UNIX):

DirX Identity Message Broker *number*
UNIX (Linux): *install_path***/etc/dmmbrk-number stop**

Start the services in the following sequence:

DirX Identity Message Broker *number*
UNIX (Linux): *install_path***/etc/dmmbrk-number start**

Java-based Identity Server

For the Java-based Identity Server, stop the services in the following sequence (use the **Services** utility on Windows or the following commands on UNIX):

DirX Identity IdS-J-*domain***-S***n-version*
UNIX (Linux): *install_path***/etc/S99dmsvrj-***domain***-S***n* **stop**

Start the services in the following sequence:

DirX Identity IdS-J-*domain***-S***n version*
UNIX (Linux): *install_path***/etc/S99dmsvrj-***domain***-S***n* **start**

C++-based Identity Server

For the C++-based Identity server, stop the services in the following sequence (use the **Services** utility on Windows or the following commands on UNIX):

DirX Identity IdS-C *version*
UNIX (Linux): *install_path***/etc/S99dmsvr stop**

Start the services in the following sequence:

DirX Identity IdS-C *version*

UNIX (Linux): *install_path*/**etc/S99dmsvr start**

On Windows, check the Event Viewer to see if the startup is successful. On UNIX, check the log files to see if the startup is successful.

Startup is successful when six entries are displayed:

- DirX Identity IdS-C information: Service started: "C:\Program Files\Atos\Identity\bin\dxmmsssvr.exe" --- PID: *nnnn*
- Wed Jul 16 15:49:03 2008 dxmmsssvr.exe NOTICE_VERBOSE dxm stt dxmstatustracker.cpp 522-0 0x00001780
  STT6126: Status Tracker is running.
- Wed Jul 16 15:49:03 2008 dxmmsssvr.exe NOTICE_VERBOSE dxm sched dxmsdrimpl.cpp 2732-0 0x00001610
  SDR6570: Scheduling is "enabled".
- Wed Jul 16 15:49:03 2008 dxmmsssvr.exe NOTICE_VERBOSE dxm sched dxmsdrimpl.cpp 1412-0 0x00001610
  SDR6567: Scheduler is running.



Date and time information as well as process and thread IDs may differ in your case.

12.2. Warnings due to Missing Definitions within Object Descriptions

Indication:

You encounter the message:

```
STG617 Multiple values exist for single value property
'<attributename>' of DN=cn=...
```

Reason:

This message provides the hint that an attribute contains multiple values but is defined as single value property in object descriptions.The service layer transports only one of the values for further processing.This warning indicates that an attribute is not defined (or not correctly defined) in object descriptions.Note that not defined attributes have the default values **type="Java.lang.String"** and **multivalue="false"**.

Solution:

Specify the attribute in the appropriate object description correctly.



If you should find a system property that is not correctly defined, define this property in the Customer Extensions section but inform support about this

problem.Then this issue is corrected in the next product delivery.



For the attribute **dxrInheritedPrivilegeLink** the central **User.xml** is not up to date.Update the file in your domains.Use the delivered file from DVD:/**Installation/Patches/user-xml/User.xml** and import it with the DirX Identity Manager into the **User.xml** object (path: **Provisioning** → **Domain Configuration** → **Object Descriptions** → **User.xml**).

12.3. C++-based Identity Server Errors

This section describes C++-based Identity server errors that can occur and the steps to take to solve the problem.

12.3.1. Authentication Failed

Indication:

You receive the message 'Authentication to LDAP server failed ...' when you start the DirX Identity service.

Reason:

The bind parameters are not correct.

Solution:

Check for correct parameters in the **[metadir]** section of the **dxmmsssvr.ini** file.

12.3.2. Can’t Connect to LDAP Server

Indication:

You receive the message 'Can’t Connect to LDAP Server' when you start the DirX Identity service.

Reason:

The DirX Identity server is unable to connect to the LDAP server.

Solution:

1. Check to see whether the LDAP server of the directory service is running.
2. Check for the correct server name and port number in the **dxmmsssvr.ini** file.

12.3.3. Can’t Extract Serial Number from keyOwnerPSE

Indication:

DirX Identity Service does not start. Error message:

Can’t extract serial number from keyOwnerPSE=*string*

Reason:

You used a **dirxgenpse** tool from a previous version of DirX Identity / DirXmetahub to generate a new key in the environment of the current DirX Identity version. This generates a corrupted certificate.

Solution:

Use the **dirxgenpse** tool from the current version of the DirX Identity DVD to generate new certificates.

12.3.4. Can’t Retrieve ATS Server Data

Indication:

You receive the message 'Can’t retrieve ATS server data' when you start the DirX Identity service.

Reason:

The DirX Identity server did not find the correct object in the DirX Identity configuration database.

Solution:

Check to see if the **dnServerName** entry in the **dxmmsssvr.ini** file is correct.

12.3.5. Configuring a Maximum of More than 128 Threads Fails

Indication:

You have successfully configured more than 128 threads for the server, using the DirX Identity Manager. Nevertheless, the server fails in starting workflows when this thread limit is reached. The log file shows warnings like:

```
99 VThread-Info: ProcessSize=0, NoActiveThreads=128
         -- ERROR    util     cthread.cpp      268         52:08:511
"MSS2008: Failure to start a thread (internal error).".
```

Reason:

Regardless of your configuration, the environment variable DIRX_MAX_THREAD specifies the maximum number of threads as a hard limit for the server. The default value is 128. The maximum possible value is 512.

Solution:

Set the environment variable in your system environment. For Windows platforms, the

related computer must be rebooted.

For Unix platforms, a central profile or the file *install_path*/.dirxmetarc* is an appropriate location. The following lines should be present for defining a maximum of 512:

```
DIRX_MAX_THREADS=512
export DIRX_MAX_THREADS
```

12.3.6. CryptoException: Export keys expired!

Indication:

An agent reports the error:

WAR(STG700): com.siemens.dirxcommon.crypto.CryptoException: Export keys expired!

Reason:

The transfer of the crypto keys to the agent failed due to a timeout. The agent startup was too slow.

Solution:

The default timeout value is 10 seconds. Set a higher value at the corresponding C++-based server object in the connectivity configuration (tab AgtSvr Configuration).

12.3.7. DirX Identity Service Does Not Start Correctly

Indication:

DirX Identity Service does not start correctly.

Reason:

This problem could be caused by previous errors when the DirX Identity service could not be started successfully.

Solution:

Stop all services and then restart them.

12.3.8. Transferring File Fails (File Service)

Indication:

Transferring a file greater than 1 MB fails.

Reason:

Running the file service with ActiveMQ only supports files with a size up to 1 MB per default

configuration.

Solution:

Increase the configuration parameter **Buffer Size** for all related C++-based Server instances in the **Expert View** → **Configuration** → **DirX Identity Servers** → **C Servers**. The parameter can be found on the **AgtSvr Configuration** tab in the section **Fileservice**.



The file service is used to transfer files between C++-based Server instances as well as for viewing status files from another machine via the DirX Identity Manager.

12.3.9. High Load during Restart

Indication:

The C++-based Identity Server is under high load after a restart.

Reason:

When the watchdog restarts the server, heavy load of the computer can be the result. The reason is that all workflows where the deviation in the schedule is not yet over, start in parallel. If you handle large amounts of data, be sure that your computer has enough workspace to handle all the processes in parallel or set the deviations to avoid overlapping of workflow starts.

Solution:

For correct setup of non-conflicting schedules see also the section workflow rules in the documentation.

12.3.10. Internal Windows NT Error

Indication:

Message: Internal Windows NT error (during start of DirX Identity Service)

Reason:

May be caused by previous errors when the DirX Identity service could not be started successfully.

Solution:

Stop all services and then restart it.

12.3.11. Service does not start with encryption enabled

Indication:

DirX Identity Service does not start correctly when encryption is enabled.

Reason:

The encryption settings in the dxmmsssvr.ini file do not match with the settings in the configuration database.

Solution:

Check the settings in the Configuration object (configuration database) with the settings in the dxmmsssvr.ini file. Encryption must be set to 1 and the password must be either requested interactively or via an extra password file.

12.3.12. No Statistics with Compression Mode

Indication:

No statistics are displayed for a workflow

Reason 1:

If the compression mode setting for a workflow is different from "Detailed" in the workflow entry or "None" in the central configuration object (which is the default) the statistics info for the workflow status object is not available. It is only available for the activity status objects.

Reason 2:

The agent does not feed the statistics interface.

Solution 1:

Use another compression level.

Solution 2:

No solution available.

12.3.13. Service Not Available

Indication:

You cannot find the DirX Identity 7.xXxx service in your services list.

Reason:

You may have provided wrong values during the installation procedure.

Solution:

1. Perform **Start** → **Run...**
2. Enter **cmd** and click **OK**.
3. Enter **dxmsvr.exe -install**.

4. Perform **Start** → **Control Panel**.
5. Double-click **Services**.
6. Double-click **DirX Identity 7.xXxx**.
7. Set **This Account** and enter the values according to your requirements.
8. Select **Automatic** and then click **OK**.
9. If the service does not start immediately, perform **Start**.

12.3.14. Service Stops Directly after Start

Indication:

The DirX Identity service stops directly after start. This is indicated by the message:

MSS 2261: DirX Identity Server exited without errors.

Reason:

This happens mainly when you use the Windows 2000 service restart feature.

The stop message of the stop sequence is displayed after the new start messages of the server.

Solution:

No action necessary.

12.3.15. SHM9550

Indication:

Workflow executions fail. ProcessInfo and/or trace files show error message:
SHM9550: timeout to get semaphore (in dxmGetShm)

Reason:

The C++-based Identity Server is in operation with Encryption mode different from NONE, and heavy system load causes some workflows to fail with said error message.

Solution:

Increment the KeyGet timeout of the related C++-based server, using the DirX Identity Manager (expert view: Configuration/DirX Identity Servers/C++-based Servers/<server object>

12.3.16. The C++-based Server Does Not Run Properly Due to a Hostname Change

Indication:

If your C++-based Server does not run properly - for example, the status tracker does not run or the ATS Messaging Server is not found - it might be due to a change of the machine name where the main C++-based Server runs and a subsequent configuration of the C++-based Server with the Configuration Wizard.

Reason:

If your local machine name changed since the last run of the Configuration Wizard or if your DNS service, for example, calculates a fully-qualified local hostname instead of a short name previously, a new run of the Configuration Wizard with the C++-based Server step selected will result in the creation of a secondary C++-based Server object in the ConfDB regarded as the active one on this machine.

Solution:

Perform the following steps to make the server run properly again (provided that you want it to run as the main server):

1. Go to the **Expert View**, right-click the root object **Connectivity Configuration Data** and go to the **DataView**.
   On the **Attributes** tab click **edit** and change the dxmSpecificAttribute: delete the value **PrimaryServerC** or set it to an empty string.This will prevent the Configuration Wizard in the next run from interpreting the new hostname to become a secondary C++-based Server.If no value for an old primary server exists, it will regard the next configured one as the primary and will overwrite the main objects with the new hostname as display name and as dxmAddress in the system object.
2. Run the Configuration Wizard with only the C++-based Server option selected again and set the hostname in the System Step to your new machine name (or set **local.host=***your_new_machine_name* in *install_path***configuration.ini** before running the Configuration wizard to get suggested already the correct name in the system step).
3. After this process, the main C++-based Server with the correct hostname is the active one again.You can delete the secondary C++-based Server object and its linked service and system objects in the ConfDB if you want, but it will do no harm if you leave it.



This issue is solved in DirX Identity V8.0C.

12.4. Java-based Identity Server Errors

This section describes Java-based Identity server errors that can occur and the steps to take to solve the problem.

12.4.1. Cannot Claim Ownership Problem

Indication:

The warning message "Cannot claim ownership" occurs in rare cases on slow machines in the Java-based Server logs.

Reason:

Concurrent processes in the Java-based Server can compete for some time for a specific LDAP object. After some configurable time, a warning is logged and the thread stops working on this task.

Solution:

Set the workflowtaskreadtimeout value in your Java-based Server object to a higher value. Perform the following steps:

1. Start the Dirx Identity Manager and log into the **Connectivity** view group.
2. Navigate to **Configuration** → **DirX Identity Servers** → **Java Servers** → *your server*.
3. Enable the **Design mode** (one of the icons in the Manager's tool bar).
4. Click the **Configuration** tab and then **Edit**.
5. Search for the word **workflowtaskreadtimeout**.
6. Change the value from **60** to a higher value (use for example 300).
7. Save the configuration.
8. Check whether the problem is gone. If not, use a higher value.

12.4.2. Class Loading Problems

Indication:

Messages indicate that class loading fails.

Reason:

The file location has changed for

- Your own Java mappings, user hooks, connector filters for real-time workflows
- Custom connectors used in real-time workflows

Solution:

Perform the following steps to solve the problem:

- Stop the Java-based Server.
- Drop your custom extensions (jar files) in

  *install_path***ids-j-***domain*-S*n***confdb\jobs\framework\lib**

  and move the jar files to

  *install_path***ids-j-***domain*-S*n***confdb\common\lib**
- Start the Java-based Server.

12.4.3. Class Loading Problems due to Customer-specific Job Deployment

Indication:

Messages indicate that class loading fails.

Reason:

The deployment process for customer-specific jobs has been changed.

Solution:

Perform the following steps to solve the problem:

- Stop the Java-based Server.
- Check the *DirX Identity Migration Guide* for changes made to job deployment. Follow the instructions described in the section "Class Loading in the IdS-J Server (for Customer-specific Request Workflow Job Implementations)".
- Start the Java-based Server.

12.4.4. Corrupted Repository Files

Indication:

Messages indicate that repository files are corrupted.

Reason:

Repository files may be corrupted.

Solution:

Perform the following steps to solve the problem:

- Stop the Java-based Server.
- Back up the appropriate repository folder or all folders if you are unsure which folder is affected.
- Delete the appropriate repository folder or all folders.
- Start the Java-based Server.
- Inform DirX technical support about the problem and send a copy of the saved folders.

12.4.5. Escalation not entered for Request Workflows

Indication:

You have configured an escalation in your request workflow, but it is never entered. You find messages like the following ones in the IdS-J log or warning file:

04.08.2009 09:53:00.941 [Thread-3] [ ] *** WARNUNG ***
Called from
com.siemens.idm.server.config.nodes.IDMRequestActivityConfigImpl.insertEscalations()
insertEscalation in Activity Approval by Privilege Managers of workflow 4-Eye Approval :
escalations ignored: no node escalations found

Reason:

Some of the default request workflows (especially the 4-Eye Approval) are not aware of escalations. The XML-description stored in the dxmContent attribute has no node <escalations>. Therefore, the IdS-J server cannot load the escalations. If you have copied one of these workflows the problem also occurs in your copy.

Solution:

Select the definition of people activities within your workflow in the Identity Manager. Go to the Data View and save the dxmContent attribute to a file. Insert a line

<escalations/>

Take care that <escalations/> is a child of <activity>!

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<activity basicType="${DN4ID(THIS)@dxmType}"
name="${DN4ID(THIS)@dxmDisplayName}"
subType="${DN4ID(THIS)@dxmActivityType}">
<retrylimit>2</retrylimit>
<description>#{Request Workflows/Assignment... </description>
<condition>only-one-may-decide</condition>
<execution>parallel</execution>
<job/>
<findParticipants type="accessPolicy">
 ...
</findParticipants>
<escalations/>
<participantsFilter>
 <class>com.siemens.idm.requestworkflow.jobs.ParticipantFilterJob</cl
ass>
 <filterclass/>
</participantsFilter>
<participantsConstraint>
 ...
</participantsConstraint>
```

```
<startcondition></startcondition>
<timeout type="static">P0Y0M1DT0H0M0S</timeout>
<sign>true</sign>
<title>#{../_Nationalization.ApprovalByPrivilegeManagers_title}</titl
e>
</activity>
```

Import the corrected File via the Data View. Restart the Java-based Server or reload its configuration.

The warning should no longer appear in the warning / log files.

12.4.6. Java-based Server Does Not Stop on UNIX

Indication:

Stopping the Java-based Server using the command of the form **S99dmsvrj-* stop** fails with an error message of the form:

Exception in thread "main" java.rmi.RemoteException: HTTP transport error: java.io.IOException: HTTPS hostname wrong: should be *hostname*; nested exception is: HTTP transport error: java.io.IOException: HTTPS hostname wrong: should be *hostname*
...
ERROR: DirX Identity Java-based Server not deactivated (ex:1).

Reason:

The hostname which has been used during creation of the certificate does not match the expected host name according to the message above. The expected host name is hostname, as quoted in the error message. This inconsistency may also be a problem for SSL communication with other DirX Identity components.

Solution:

Stop all DirX Identity services. Create all the related key stores so that the host name is specified exactly as quoted in the exception (namely hostname) and try again.

12.4.7. Java-based Server Processing Stopped after Re-Boot

Indication:

After a re-boot, the Java-based Server stopped processing.

Reason:

After reboot, the Java-based Server tries to read the LDAP configuration in a loop. If this fails for the pre-configured time (about 5 minutes), the Java-based Server does not work because the configuration could not be loaded.

Solution:

Increase the timeout and repeat values in the file *install_path***/ids-j-***domain***-S***n***/bin/bindcredentials.xml** to values that are sufficient for your environment and restart the Java-based Server.

12.4.8. java.io.EOFException

Indication:

You encounter the following message in the Java-based Server logs:

IDSJADP007 The JMS messaging system of adaptor *one_of_the_configured_adaptors* reported the exception javax.jms.JMSException (root cause is java.io.EOFException)

Reason:

The C++-based server is not available.

Solution:

Check the availability of the C++-based server. If the C++-based server is running and the problem still persists, restart the C++-based server and the Java-based server.

12.4.9. No Automatic Restart of the Server after Crash on Linux

Indication:

After a crash on Linux, the Java-based Server is not re-started automatically.

Reason:

On Window platforms the Java-based Server is automatically re-started after a crash. The native mechanisms on these platforms are used to perform the restart. On Linux platforms, there is currently no native operating system support to perform this task.

Solution:

You need to build your own watchdog; for example, by using third-party tools.

12.4.10. Repository Exceptions for Duplicate Workflows

Indication:

The log files of the Java-based Server contain warnings of the following form:

```
-------------------------------------------------------------------
----------
06.02.2007 09:44:09 local-24 [  ] *** WARNING ***
Called from
-------------------------------------------------------------------
```

```
com.siemens.idm.server.adaptor.jms.AbstractJMSEventAdaptor.wakeup()
IDSJADP005 Adaptor 'PasswordChangeListener' attempted to delete msg
(identified by [segment=1, id=0, expired=false]) from the repository
but failed
------------------------------------------------------------------------
----------

com.siemens.idm.jini.persistence.EntryNotFoundException: Entry for
[segment=1, id=0, expired=false] doesn't exist in segment 1
...
```

The number of these warnings increases during productive operation of the server.

Reason:

Multiple password or real-time synchronization workflows are active for one topic. The server stores password change and real-time events in an internal repository. After processing such a message, each workflow tries to delete it from the repository. Only one of them succeeds in deleting the message while the others will fail.

Solution:

Check the configuration for active password synchronization workflows. Avoid configuring multiple active workflows for the same topic. Reload the workflow configuration of the Java-based Server using either the Identity Manager or the Identity Web Admin.

12.4.11. Running Java-based Agents with SSL to the LDAP Server

Indication:

You have set up the LDAP server for the provisioning store with SSL. Running Java-based DirX Identity agent (for example PolicyAgent) with SSL fails. The trace includes the following error message:

```
ERR(STG000): Unexpected exception: 'netscape.ldap.LDAPException:
JSSESocketFactory.makeSocket host:636,
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target (91); Connection to
LDAP-Server not possible'.
WAR(STG700): netscape.ldap.LDAPException:
JSSESocketFactory.makeSocket host:636,
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target (91); Connection to
```

```
LDAP-Server not possible
```

Reason:

The suitable LDAP server certificate is not in the trust store of the Java runtime environment of the Policy Agent (*dxi_java_home***/lib/security/cacerts**).

Solution:

Import the suitable certificate into the trust store of the Java runtime environment, as described in the *Connectivity Administration Guide*, chapter "Managing the Connectivity System", section "Setting Up the Java-based Server for SSL to the Connectivity Configuration", subsection "Managing Keys".

12.4.12. UNIX: Problem Stopping Java-based Identity Server Running with SSL

Indication:

Stopping DirX Identity using the command of the form **S99dmsvrj-* stop** fails with an error message of the form:

```
Exception in thread "main" java.rmi.RemoteException: HTTP transport
error: java.io.IOException: HTTPS hostname wrong: should be
<hostname>; nested exception is:
HTTP transport error: java.io.IOException: HTTPS hostname wrong:
should be <wlf-vm01-rhel.myCompany.de>
...
ERROR: DirX Identity Java-based Server not deactivated (ex:1).
```

Reason:

The hostname which has been used while creating the certificate does not match the expected host name according to the message above. The expected host name is *hostname*, as quoted in the error message. This inconsistency may also be a problem for SSL-communication with other DirX Identity components.

Solution:

Stop all DirX Identity services. Create all related key stores so that the host name is specified exactly as quoted in the exception message (namely *hostname*) and try again.

12.4.13. Unable to Run More than One Java-based Server per Domain

Indication:

You are unable to run more than one Java-based server per domain.

Reason:

Using multiple domains, each with one or more assigned Java-based servers, SSL and SSO are not yet supported by the Configuration Wizard and no documentation is provided yet.

Solution:

You can configure one Java-based server in a DirX Identity installation for SSL and you can use any of the provided SSO configurations for Web Center.

12.4.14. "No Ldap Entry available for" Messages During Startup

Indication:

You get messages like the following during startup of the Java-based server:

```
-------------------------------------------------------------
09.05.2016 00:22:31.428 [Thread-1] [  ] *** WARNING ***
Called from
com.siemens.idm.server.resource.ldap.ResVariable.getAttrVal()
 no Ldap Entry available for dxmSpecificAttributes(mapmailaddress)
-------------------------------------------------------------
09.05.2016 00:22:31.444 [Thread-1] [  ] *** WARNING ***
Called from
com.siemens.idm.server.resource.ldap.ResVariable.getAttrVal()
 no Ldap Entry available for dxmService-DN
```

Reason:

In the workflow definitions some fields cannot be filled. DN links or LDAP attributes are empty.

Solution:

Perform the following steps:

1. Stop Java-based server.
2. Backup the file **ids-j-*/confdb/server/local/logger.properties** and add a line **com.siemens.idm.server.resource.ldap=FINEST**
3. Start the Java-based server.
4. Check the file **server*00000.txt** for warnings like the one above. Search backwards from such a warning for the string "mapPath: in:". The workflow path follows this string. Check the workflow attributes and links.
5. Restore the **logger.properties** file.

12.4.15. ActiveMQ Warning about Missing Certificates

Indication:

The **wrapper.log** file of the Tanuki Java service wrapper contains a warning about a certificate issue:

```
STATUS | wrapper  | 2020/12/02 16:55:25 | --> Wrapper Started as
Service
STATUS | wrapper  | 2020/12/02 16:55:25 | Java Service Wrapper
Standard Edition 64-bit 3.5.37
STATUS | wrapper  | 2020/12/02 16:55:25 |   Copyright (C) 1999-2018
Tanuki Software, Ltd. All Rights Reserved.
STATUS | wrapper  | 2020/12/02 16:55:25 |
http://wrapper.tanukisoftware.com
STATUS | wrapper  | 2020/12/02 16:55:25 |   Licensed to Atos IT
Solutions and Services GmbH for DirX Identity
STATUS | wrapper  | 2020/12/02 16:55:25 |
WARN   | wrapper  | 2020/12/02 16:55:25 | A signature was found in
"D:\Programme\Atos\DirXIdentity\messagebroker\bin\service\windows_x86
_64\wrapper.exe", but checksum failed: (Errorcode: 0x800b010a) Eine
Zertifikatkette zu einer vertrauenswürdigen
Stammzertifizierungsstelle konnte nicht aufgebaut werden.
(0x800b010a)
WARN   | wrapper  | 2020/12/02 16:55:25 |   Signer Certificate:
WARN   | wrapper  | 2020/12/02 16:55:25 |     Serial Number:
WARN   | wrapper  | 2020/12/02 16:55:25 |       00 90 4d 8f d1 f3 86
8a ad 5f 17 e8 93 41 c3 08 f2
WARN   | wrapper  | 2020/12/02 16:55:25 |     Issuer Name: COMODO RSA
Code Signing CA
WARN   | wrapper  | 2020/12/02 16:55:25 |     Subject Name: Tanuki
Software Ltd.
WARN   | wrapper  | 2020/12/02 16:55:25 |   TimeStamp Certificate:
WARN   | wrapper  | 2020/12/02 16:55:25 |     Serial Number:
WARN   | wrapper  | 2020/12/02 16:55:25 |       16 88 f0 39 25 5e 63
8e 69 14 39 07 e6 33 0b
WARN   | wrapper  | 2020/12/02 16:55:25 |     Issuer Name: UTN-
USERFirst-Object
WARN   | wrapper  | 2020/12/02 16:55:25 |     Subject Name: COMODO
SHA-1 Time Stamping Signer
WARN   | wrapper  | 2020/12/02 16:55:25 |     Date of TimeStamp :
2018/12/14 13:30:25The error is not directly related to the Wrapper's
```

```
signature, therefore continue...
STATUS | wrapper  | 2020/12/02 16:55:26 | Launching a JVM...
```

Reason:

The executable **wrapper.exe** is signed but the verification of the signed executable cannot be done because root certificates for the Tanuki Software are missing.

Solution:

Install the necessary root certificates for the Tanuki Software.

Find information on the Tanuki Software website (https://wrapper.tanukisoftware.com/doc/english/troubleshooting.html#9):

Cross-signed certificates issued by AddTrust External CA Root (addtrustexternalcaroot.crt, comodorsaaddtrustca.crt and comodorsacodesigningca.crt) are suitable for all versions of the Wrapper, and more compatible with legacy devices using old versions of the Windows:

https://support.comodo.com/index.php?/Default/Knowledgebase/Article/View/985/108/code-signing-sha-2

Using 'mmc' or 'certmgr.msc', you can confirm the following installation:

"AddTrust External CA Root" should be installed under "Trusted Root Certificate Authorities\Certificates"

"COMODO RSA Certification Authority" and "COMODO RSA Code Signing" should be installed under "Intermediate Certification Authorities\Certificates". Make sure that their signature hash algorithms match the version of the Wrapper you are using (sh384 or sha-2 for Wrapper 3.5.28 and above, and sha-1 for earlier versions). This can be checked by double-clicking on the certificate and opening the 'Details' tab. You may also check that these certificates are enabled in the properties window accessible from the context menu.

"UTN-USERFirst-Object" should be installed under "Third Parties Root Certification Authorities\Certificates".

13. Problems with LDAP Servers

This section describes problems that relate to the LDAP meta directory servers used with DirX Identity and how to solve them.

13.1. "No Such Attribute or Value" Error

Indication:

While running DirX Identity services like privilege resolution or real-time workflows, the LDAP Server delivers the error message
"No such attribute or value" or "Attribut nicht vorhanden".

Reason:

This can happen during concurrent LDAP accesses.Two clients try to change the same attribute of the same entry at (almost) the same time.One service performs a delete value and add value operation where the delete <value> operation tries to delete a value that is no longer there (it was meanwhile deleted by another service).This causes the message.

Solution:

If the attribute has only a single value (regardless of its schema definition), you can switch the operation for this attribute to replace mode.Then the attribute is simply set regardless what the actual value is.Within DirX Identity, you can define this behavior per attribute in the relevant object description.

Find the relevant object description and add the definition incremental="false".This forces the services to work for this attribute in replace mode.Example:

<property name="employeeNumber" label="Employee Number" type="java.lang.String" incremental="false"/>



If this message occurs at an object that masters an attribute from another object, then the attribute definition of the mastered object is relevant and must be changed.If you master for example the attribute employeeNumber at an account object from the user object you must add **incremental="false"** to the **user.xml** to solve the problem.Be aware that there can be a hierarchy of XML definitions that inherit definitions!

13.2. Service does not Operate Properly after Directory Restore

Indication:

Operation of the whole system is not correct after a restore of the directory content.

Reason:

Exchanging directory content (restore operation for DirX Server) requires a restart of all services to guarantee correct operation.Due to various caches and local stores, the DirX Identity servers are no longer in sync.

Solution:

Perform the following steps:

- Restart the directory server.
- Restart all C++-based DirX Identity servers.
- Restart all Java-based DirX Identity servers (first the servers, then the workers).
- Restart all Tomcat servers (or restart the related applications).

13.3. "Unwilling to Perform" Error

Indication:

The LDAP server refuses a request with "Unwilling to perform".

Reason:

Unfortunately there are many reasons for this error message.A possible reason is that the client works with paging and the time between two page requests was too long.For DirX this time is set per default to 5 minutes.

The DirX Identity service agent (privilege resolution, consistency check) runs with pages of several hundred entries for optimized performance.Due to complex role structures, the resolution of hundreds of entries might reach the limit of 5 minutes.

Solution:

If this is the reason (check the LDAP auditing details), you can either

- increase the paging timeout of the LDAP server for example to 10 minutes, or you can
- decrease the number of entries per page in the service agent configuration.

If a Java-based application or server received the "Unwilling to Perform" error from a DirX LDAP server or reports are not sorted correctly see "Problems with LDAP Servers → DirX Directory → "Unwilling to Perfom" Error" for details.

13.4. DirX Directory

This section describes problems that relate to the DirX LDAP server.

13.4.1. Check-pointing does not Work

Indication:

Check-pointing does not work.

Reason:

There is no ordering matching rule for the sort attribute specified at the source side. Search request with attribute values greater or equal do not work. These search requests are essential for check-pointing operations.

Solution:

Ensure that an Ordering-Matching-Rule is specified for the sort attribute at the source side (see the input channel definition) for LDAP to LDAP workflows.

Here is an example of a schema file that extends the attribute dxmEXpersonalTitle:

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( 1.3.12.2.1107.1.3.102.4.3.87
 NAME 'dxmEXpersonalTitle'
 DESC 'This is a proprietary DirX Identity attribute.
 EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreOrderingMatch
 SUBSTR caseIgnoreSubstringsMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 SINGLE-VALUE )
```

The relevant line is **ORDERING caseIgnoreOrderingMatch**.

13.4.2. DIRX_LOGCFG_FILE Variable not allowed

Indication:

The system does not behave correctly.

Reason:

DirX Identity does not support the variable DIRX_LOGCFG_FILE.

Solution:

The use of this variable is not allowed.

13.4.3. Insufficient Size Limit for Searches

Indication:

The DirX Identity software operates incorrectly after Move Object or Copy Object operations or during provisioning operations.

Reason:

Running DirX Identity with a restricted size limit can result in incorrect behavior (for example, during rule-based searches over the whole database). This is especially a problem when the size limit is set by default to 2048.

Solution:

Workaround: Check that the search size limit for the configuration database is set to a high value (minimum is 10 000). We recommend setting an unlimited value.

If this is not possible, calculate the limit this way:

- For a Connectivity domain, evaluate the number of configuration objects.
- For a Provisioning domain, evaluate the maximum number of user entries in the DirX Identity domain, evaluate the maximum number of accounts for any of your target systems in the DirX Identity domain and evaluate the maximum number of privileges and other configuration objects in the DirX Identity Store.

Set the size limit (SL) to the highest number evaluated plus 20 %.

Note that the DirX Identity product will set user policies (with size limit 100000) for the subtree **dxmC=DIrXmetahub** (Connectivity data) and for all the Provisioning domains (Provisioning data). If you define your own limit, these limits are not changed during upgrade installation.

If you want to change the limits, use the following **dirxadm modify** operation to set a user policy:

For Connectivity Data:

modify / -addattr USP={USN={/dxmc=DirXmetahub},SL=-100000}

For the Provisioning Domain cn=My-Company:

modify / -addattr USP={USN={/CN=My-Company},SL=-100000}

13.4.4. Uninstallation and Installation

Indication:

DirX Manager, **dirxcp** and **dirxadm** no longer run after an uninstallation and subsequent installation of DirX.

Reason:

The uninstallation removed all the DirX Identity abbreviation files from the directory *dirx_install_path***client\conf**.

Solution:

Copy the required files from *dirxmetahub_install_path***client\conf** to the directory

*dirx_install_path***client\conf**.

13.4.5. "Unwilling to Perform" Error

Indication:

- Indication 1: A Java-based application or server gets an "Unwilling to perform" error message from a DirX LDAP server instance.
- Indication 2: Reports are not sorted correctly.

Reason:

Running a Java-based application that requires server-side sorting fails because the Java library runs by default with no de-referencing. This behavior results in 'Unwilling to perform' messages. You can also see a 'Database problem detected' message in the LDAP audit.

Because DirX Identity software is intelligent enough to overcome such problems, it tries without server-side sorting. This might work but the result of a report is, for example, not sorted.

This problem applies for all supported DirX LDAP servers.

Solution:

Set derefAlias overwrite mode for the LDAP server if the deployment uses no aliases (this is the case for DirX Identity installations). Perform the following steps:

- Use **dirxcp** to specify the value **X** for dontDerefAlias in the attribute ldapSearchSvcCtl of the ldapConfiguration subentry. The LDAP server now always uses "derefAlways" independent of the setting in the LDAP request, that is the DSA returns aliased objects (instead of the ALIAS itself) and hence avoids useless bit string computations in the DSA process.

Example dirxcp statements:

First, read the existing LDAP search service control values (ldapSearch ldapSearchSvcCtl) because you must specify all values in the modify operation:

```
dirxcp> search o=my-company -sub -filt ocl=ldapconfiguration -vtype
subentry -attr ldapSearchSvcCtl
```

dirxcp displays the following sample output:

```
{cn=ldapConfiguration,o=My-Company
ldapSearchSvcCtl=preferChaining=T:chainingProhibited=F:localScope=F:d
ontUseCopy=F:dontDereferenceAlias=F:subentries=F:copyShalldo=T:priori
ty=1:timelimit=0:sizelimit=0:scopeofreferral=0:attributesizelimit=0}
```

Then perform the following **dirxcp** (ldap binding) statement to set the derefAlias overwrite mode (**dontDereferenceAlias=X**):

```
dirxcp> modify cn=ldapConfiguration,o=My-Company -replaceattr
ldapSearchSvcCtl=preferChaining=T:chainingProhibited=F:localScope=F:d
ontUseCopy=F:dontDereferenceAlias=X:subentries=F:copyShalldo=T:priori
ty=1:timelimit=0:sizelimit=0:scopeofreferral=0:attributesizelimit=0
```

- Restart the LDAP server so that the modification of the LDAP configuration subentry becomes effective.
- If you want to verify your modification, check whether the LDAP server has logged the message "DerefAlias set to ALWAYS" in the file **TMP/fatalLDAP** of your DirX installation.

See also the general information about this error in "Problems with LDAP Servers → "Unwilling to Perform" Error".

13.4.6. Wrong use of DAP

Indication:

Error during schema installation (cannot connect to the directory server).

Reason:

During DirX Identity configuration, AccessControl subentries are created using **dirxcp** of the DirX installation via DAP.

Solution:

Make sure that **dirxcp** can connect to the directory server using DAP. Check the following items:

- The address given in the environment variable **DIRX_OWN_PSAP**.
- The server addresses listed in DIRX_INST_PATH**/client/conf/dirxcl.cfg**.

14. Problems with Utilities

This chapter describes utilities-related problems.

14.1. Locale Dependency Problem With logANT

Indication:

The logANT (log analyzer) for Java-based Server log files reports an error if the log files were produced with another locale.

Reason:

Locale specific information cannot be converted properly.

Solution:

Configure the correct locale in the start script, for example:

```
-Duser.language=en -Duser.country=US
```

14.1.1. Run Workflow Tool Hangs

Indication:

When called from another folder, the run workflow tool hangs.

Reason:

The run workflow tool (**runwf.bat** or **runwf.sh**, respectively) does not work if it is called from a folder which is different from the location of the tool.

Solution:

In order to avoid this problem, the calling batch file must perform a **cd** command to the folder where the run workflow tool is located.Here is a sample snippet for Windows:

```
set workdir=C:\Program Files\Atos\DirX
Identity\tools\utilities\workflowstarter
cd %workdir%
call runwf.bat
```

14.1.2. dirxgenpse Cannot Contact LDAP Server

Indication:

Using **dirxgenpse** with DirX Directory as the Connectivity store fails after some computation time.The output looks like this:

```
1. Search completed successfully.
2. Entries found: 1
3. Search references returned: 0
4. ldap_modify_ext_s: Can't contact LDAP server
Could not update directory entry - exiting
```

Reason:

The **dirxgenpse** utility reads the relevant LDAP directory entries before computing the PSE. Computing the relevant certificate and keys may be time-consuming especially when using a non-default length. The message indicates that the update of the relevant directory entry failed due to an LDAP connection timeout.

Solution:

Increase the LDAP connection idle time according to the DirX Directory documentation, restart the DirX Directory service, and retry the operation. After successful operation, set this idle time back to the value needed in your productive environment, then restart the DirX Directory service again. Here is a sample **dirxcp** command line for setting the idle time to 900 seconds:

```
modify cn=ldapConfiguration,o=My-Company -replaceattr
ldapConnectionIdleTime=900
```

15. Problems in Complex Scenarios

This section provides hints for complete complex scenarios like real-time workflows or password synchronization. Follow the steps in the following sections thoroughly to find the cause of your problem.

If these guides do not solve your problem, read other relevant parts of the DirX Identity documentation thoroughly or check for other troubleshooting hints in this document. If this investigation does not help, inform DirX Identity technical support about the problem.

The scenarios in this section include:

- Password synchronization

15.1. Password Synchronization

This section provides troubleshooting hints for a password synchronization environment.This description assumes a password management scenario that starts with a change of the password in a Windows domain.

See the section "Password Changes from a Windows Domain" in the *DirX Identity Connectivity Administration Guide* to understand this scenario.See also the section "Java-based Workflow Runtime Operation" in the *DirX Identity Connectivity Administration Guide* to understand how the Java-based Server works.

If your password synchronization does not work, perform the steps described in the following sections to locate and solve the problem.

15.1.1. Initial Checks

Perform the following initial checks to find out whether the servers are running correctly:

- Check that the **Windows Password Listener service** on the domain controller machine is running correctly. (On Windows, use the Services panel together with the Event Viewer.) If something is not correct, restart the Windows Password Listener service and check again. If this does not help, follow the instructions in section "Check Windows Password Listener" to solve the problem.
- Check that the **C++-based Server** is running correctly. (On Windows, use the Services panel together with the Event Viewer.) If something is not correct, restart the C++-based Server and check again. If this does not help, follow the instructions in section "Analyzing Message Server Operation" to solve the problem.
- Check that the **Java-based Server** is running correctly. (On Windows, use the Services panel together with the Event Viewer.) If something is not correct, restart the Java-based Server and check again. If this does not help, follow the instructions in section "Analyzing Java-based Server Operation" to solve the problem.

15.1.2. Identifying the Component

Perform the following steps to identify the component working incorrectly:

Windows Password Listener → Messaging:

- Check the current state of the queue *your_domain***.http://mc0x5ngc.ww930.my-it-solutions.net:8161/admin/browse.jsp;jsessionid=10jdd39ri3r5seytbedqscdg0?JMSDestination=my-company.dxm.event.pwd.changed[xm.event.pwd.changed]** with ActiveMQ Web Console (Queues page) and note the following values for later reference:
- The number of messages enqueued and dequeued. The number of consumers should be one. The number of pending messages should be low or zero.
- The number of sent requests and received responses the **PasswordChangeListener** in the Java-based Server has processed up to now.
- **Change a password** of an account in the Windows domain. This action should increase the number of enqueued and dequeued messages in Web Console.
- If the Messaging broker does not receive a message, check the Windows Password Listener for correct operation (go to the section "Checking Windows Password Listener".)
- If there is no subscriber queue, either the Java-based Server does not run or the PasswordChangeListener adaptor does not work (it has not subscribed). Proceed to the section "Analyzing Java-based Server Operation".
- If both conditions are fulfilled, the password change message was transferred correctly to the Java-based Server. If not, proceed with the section "Analyzing Message Server Operation".

Messaging → Java-based Server:

- Check that the number of sent requests of the **PasswordChangeListener** adaptor in the Java-based Server has increased by one. Check also that the number of received responses has increased by one.
- If this is not the case, proceed with section "Analyzing Java-based Server Operation".

Java-based Server → Event Manager:

- Check with Web Center that the password of the user was changed correctly. If not, proceed with section "Analyzing Event Manager Operation".
- Check in one of the connected target systems that are prepared for password synchronization that the password of the user was changed correctly.
- If this is not the case, proceed with section "Analyzing Target System Workflow Operation".
- If this is the case, your password change scenario works correctly.

15.1.3. Checking Windows Password Listener

Perform the following steps to check whether Windows Password Listener works correctly:

- Stop the Windows Password Listener service.
- Set **Trace=3** in the file **libdxmEventListenerAds.ini** in the folder conf.
- Change a password of a user. Check that there is a file in the **PasswordFiles** directory; for example, a file that look like 20080308131048823_1010. Also check the file dxmEventListener.log. It should clearly indicate that there was a password change for the user. This shows that the plug in the domain controller works correctly.
- Set **Trace=0** in the file **libdxmEventListenerAds.ini** in the folder conf.
- Start the Windows Password Listener service. Check that it runs correctly.
- Check whether the file in the **PasswordFiles** directory is gone. If yes, the Windows Password Listener service works correctly. (It could send the encrypted password to the Messaging Server.)
- If you still cannot see that a new message passed the dispatcher queue, check the parameters in the **options.ini** file to see if you're using the correct Messaging Service. If this is the problem, set the correct parameters and restart the Windows Password Listener service.
- If the Windows Password Listener does not work correctly (the password file is still in the PasswordFiles folder), check that a folder CertFile contains a certificate file (Certificate.txt).
- If there is no certificate file that is required for correct encryption of the password, check that the Windows Password Listener server can contact the Messaging Service (it tries to send a certificate request message to get the certificate from the LDAP directory).
- Check the messages in the Event Viewer and in the related log files whether there are errors. Eventually increase the trace level in the file **options.ini** file, restart the Windows Password Listener service and check for more detailed information. Check also that the Messaging Service is reachable from the domain controller machine. If not, change your firewall configuration.
- If there are any other messages in the Event Viewer, analyze them and solve them.
- If the Messaging Service is reachable, check that a certificate request message from the Windows Password Listener passed the subscriber queue with the topic **dxm.event.certificate.*.** If not, proceed with section "Analyzing Message Server Operation".
- If yes, check with the Web Admin, that the ConfigurationHandler adaptor of the Java-based Server processed the message and sent a certificate.
- If not, check that a correct certificate is available at the server_admin object (use the DataView of the DirX Identity Manager for this check) and if that is the case, proceed with the section "Analyzing Java-based Sever Operation".

15.1.4. Analyzing Java-based Server Operation

Perform the following steps to analyze Java-based Server operation:

- Check that the Java-based Server startup was successful. (On Windows, use the Event Viewer; on UNIX, use the log files in the folder *install_path***ids-j-***domain**-S*n***logs**.)
- Check that there are no warning or error messages in the **server-*** log files.

- If there are warning or error messages, analyze the reason of the message. (See the "Logging Messages" chapter for details.) Solve the related issue. If the reason is not obvious, use higher logging levels for the relevant component, restart the server and then check the messages again.

If there are problems, check the following issues:

- Check the parameters **host**, **port**, **user** and **domain** in the *install_path***ids-j-***domain*-**S***n***bindcredentials.xml** file are set correctly. Change it or perform a configuration again if not.
- Be sure that the **ldap** (password) and **pin** values in the file *install_path***ids-j-***domain*-**S***n***private\password.properties** are correct. If you are in doubt, set the values again in clear text. After the next start of the server, the values are encrypted again.
- If you use two certificates in parallel (during a certificate migration phase), be sure that the **previousPin** parameter in this file is set correctly. Set it again in clear text if you are in doubt.
- In Web Admin, click the **Java Server** node and then check the server state (which should be "The server is running"). If the server is suspended, click **Resume Server**.
- Open the **Adaptors** section and check for each adaptor the state (which should be "The adaptor is running"). If an adaptor is suspended, click **Resume Adaptor**.
- Check that the **PasswordChangeListener** is in the state "listening". If not, suspend and resume the adaptor. If this action does not help, check the error reason in the logging. Check also that the Messaging Service is reachable from this machine.
- Check the **DeadLetterQueue**. Click **Search Requests** and open the password change request that failed. Check for correct topic (cluster and resource are filled correctly).
- Open **Workflow Definitions → Real-time Workflows** and check that your **Event Manager** workflow is present. If not, proceed with the section "Analyzing Event Manager Operation".
- If the Event Manager workflow is visible, check that the **whenApplicable** section is set correctly. If not, proceed with the section "Analyzing Event Manager Operation"

If you still can’t solve the problem, look at the section "Problems with Identity Servers" for additional hints.

15.1.5. Analyzing Event Manager Operation

Perform the following steps to analyze Event Manager operation:

- In DirX Identity Manager’s Connectivity view, check that the **Event Manager** workflow is set to active.
- Check that the **Is applicable for** section is set correctly. You can restrict the type of application (see the password message for correct setting), and you can also restrict the Event Manager to specific **Cluster** and **Domain** values. Typically, all these values are set to ***.
- Perform **Load IdS-J Configuration** to upload the changed workflow to the Java-based Server.

- With Web Admin, check that the Event Manager is now present in the **Workflow Definitions → Real-time Workflows** section.
- If this is the case, but the workflow does not work correctly, check the other parameters of the workflow definition like **Connected Directory** and **Bind Profile**. Set the password of the bind profile again if you are not sure whether it is correct.
- Be sure that the **SMTP** part of the workflow definition is set correctly (**SMTP Server** and **Bind Profile**). Set the correct email addresses.
- If the **Event Manager** workflow works, check the status entry in the Monitoring view of the Connectivity view group (is located under the **Event based** node).
- If you encounter the message
  "Searching account; Exception: siemens.dxm.connector.DxmConnectorException: ERR(JOIN516): Account "*name*" doesn't exist for target system "*target system*""
  then the Windows account of the user could not be found. Verify that the account is not in the target system of DirX Identity. Check your target system synchronization and validation workflows for correct operation.
- If the account can be found, the status entry should indicate that the password of the user could be changed successfully. Note that DirX Identity checks whether the password was really changed. If not, the password change operation is aborted. No further action is performed.
- You should also find for each target system (besides the Windows target system that issued the modification) a message indicating that a synchronization action has been triggered. If this is not the case, check for all relevant target systems that **Prevent Password Synchronization** is not set.

15.1.6. Analyzing Target System Workflow Operation

Perform the following steps to analyze target system workflow operation:

If the Event Manager status entry indicates that synchronization messages could be sent, but the password synchronization to a target system does not work correctly, perform the following checks:

- In DirX Identity Manager's Connectivity View, check that the relevant **password synchronization workflow** is set to active.
- Check that the **Is applicable for** section is set correctly. The **Type**, **Cluster** and **Domain** fields must fit with the corresponding fields of your target system definition in the Provisioning view group.
- Perform **Load IdS-J Configuration** to upload the changed workflow to the Java-based Server.
- In Wed Admin, check that the password synchronization workflow is now present in the **Workflow Definitions → Real-time Workflows** section.
- If this is the case, but the workflow does not work correctly, check the other parameters of the workflow definition like **Connected Directory** and **Bind Profile**. Set the password of the bind profile again if you are not sure whether it is correct.
- Be sure that the **SMTP** part of the workflow definition is set correctly (**SMTP Server** and

Bind Profile). Set the correct email addresses.

- If the **password synchronization** workflow works, check the status entry in the Monitoring view of the Connectivity view group (is located under the **Event based** node).
- If you encounter error messages, analyze them and solve the problem.

If you still can't solve the problem, inform DirX Identity technical support about the problem.

15.1.7. Analyzing Account Password Manager Operation

If a password for a privileged account has been changed by Web Center or by the Account Password Manager Expiration or Reset workflow, the Web Center can only display the changed password correctly - assuming that the logged-in user has the appropriate rights - if the settings in the Connectivity Database regarding Encryption mode are set consistently:

- If encryption mode is set, which should always be the case in a productive environment, all components do a strong encryption and no problems arise with password display.
- If encryption mode is not set, the server_admin user object in the Connectivity Database should not have a userCertificate. Otherwise, the Account Password Manager encrypts passwords strongly and Web Center expects a scrambled password only because encryption mode is not set.

16. Logging Messages

The logging messages of the various components are listed in this chapter. For any logging message the user will find the following:

Message Number and Text - A unique message number together with a message text is presented.

Explanation - An explanation of the error, usually a description of what has happened in the system.

Action - The user action required for debugging.

The logging messages are divided into the subsequent groups:

C++-based Server - Messages of the C++-based Server components.

Common Java Components - Messages of common Java components.

Connectivity - Messages of agents and connectors.

High Avaibility - Messages for high avaibilty

Java-based Server - Messages of the Java-based Server components.

Manager - Messages of the DirX Identity Manager.

Services - Messages from DirX Identity Services.

16.1. Logging Messages for the C++-based Server Components

The C++-based server consists of two parts:

- **Workflow Server** - The component that controls the batch-type synchronization workflows.
- **Connector Server** - A set of components that control C/C++-based connectors.

ACL... Logging messages for the agent client interface.

AGC... Logging messages for the agent controller.It starts, stops and controls agent runs within the workflow server part.

AMQ... Logging messages for the the DirX Identity Active-MQ client.

CDB... Logging messages for the connectivity configuration database interface.

CMR... Logging messages for the configuration manager component. It loads and updates configuration data for all connector server components.

CSR... Logging messages for the connector server core component.

CUT... Logging messages for the connector server utilities.

CWR... Logging messages for the connector server wrapper.

DXC... Logging messages for the crypto library for C/C++-based components.

F1_... Logging messages for the sample filter extension. A skeleton for filter messages.

FSV... Logging messages for the file service component. Controls the exchange of files between different C++-based server instances via JMS messaging.

JDBF... Logging messages for the C++-based server delta filter.

MSS... Logging messages for the C++-based DirX Identity workflow server core component.

SDR... Logging messages for the scheduler of the workflows server.

SHM... Logging messages for the shared memory component of the workflow server used as interface between the agent wrapper and the agents.

STT... Logging messages for the status tracker of the workflow server part.

UTL... Logging messages for the utilities component.

WFE... Logging messages for the workflow engine of the workflow server part.

16.1.1. Logging Messages (ACL)

This topic describes the logging messages returned by the DirX Identity agent client interface.

ACL4002 (FTL): Exception occurred in *string*!

> **Explanation**: Internal Agent Client problem.
>
> **Action**: Inform DirX Identity technical support about the problem.

16.1.2. Logging Messages (AGC)

This topic describes the logging messages returned by the DirX Identity agent controller.

AGC4106 (ERR): Error in function *string* from ConfDB.

> **Explanation**: This message is returned if the configuration database API cannot read or write some of the configuration data. For example the configuration file (option.ini) cannot be created because a reference inside it cannot be resolved.
>
> **Action**: Use the DirX Identity Manager monitor view to examine the specific error message from the API.

AGC4107 (ERR): Cannot load library: *string*.

Explanation: This message is returned if the wrapper DLL cannot be loaded.

Action: Inform DirX Identity technical support about the problem.

AGC4109 (FTL): Exception occurred in *string* (Num: *number*)!

Explanation: Internal Agent Controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4110 (ERR): Cannot create file directory: *string*.

Explanation: This message is returned if a file directory of the specified status or work path cannot be created.

Action: Check your hardware or network configuration. If the error recurs, inform DirX Identity technical support about the problem.

AGC4113 (FTL): Insufficient memory in function *string*!

Explanation: The message is returned if the Agent Controller is unable to allocate the memory that is required to process the related workflow. This error can occur if there are too many processes running in parallel or as a consequence of too a heavy system load for DirX Identity.

Action: Reduce the system load and restart the DirX Identity server. If system load does not seem to be too heavy, inform DirX Identity technical support about the problem.

AGC4121 (INF): At least one *string* file could not be copied.

Explanation: In most cases, the reason for this message is that a file was specified in the job data-for example a report file-but was not created after the job finished. As a result, the copy operation of the specified file to the status area fails.

Action: None

AGC4122 (ERR): Function: *string* sets error *string*.

Explanation: This message is returned if a function stops with a severe error.

Action: Look at the error returned and try to solve the problem.

AGC4123 (ERR): No pin available for private key.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4124 (ERR): No private key in database. Job encryption mode is *string*. Job disableEncryption flag is FALSE.

Explanation: The private key is not in the data base.

Action: Check the database.

AGC4125 (INF): START Activity.
Activity Name: *string*

Workflow Name: *string*
Activity Start Time: *string*
Activity Exp. Time (s): *string*

Explanation: An activity has been started.

Action: None.

AGC4126 (INF): END Activity.
Activity Name: *string*
Workflow Name: *string*
Return code: *number*

Explanation: An activity has been terminated.

Action: If the return code is other than 0, then view the status area to identify the problem cause.

AGC4127 (WAR): Can't delete file '*string*': *string*

Explanation: Cannot delete file.

Action: Check detailed message and analyze preceding error messages.

AGC4128 (INF): Ignoring return code from Save Status Info (publish to status tracker error: *number*)

Explanation: Error code from saving status info is ignored.

Action: In dxmmsssvr.ini see flag IgnoreSaveStatusInfo.

AGC4129 (WAR): Checking file '*string*' to copy to status area - stat() call returned errno: *string*. Unable to copy file.

Explanation: Error code from stat() call to check if file to copy into status area exists.

Action: Check errno message why the file can not be copied/accessed.

AGC4503 (FTL): Insufficient memory.

Explanation: This message is returned if the agent controller is unable to allocate the memory that is required to process the related workflow. This error can occur if there are too many processes running in parallel or as a consequence of too heavy a system load for DirX Identity.

Action: Reduce the system load and restart the DirX Identity server. If system load does not seem to be too heavy, inform DirX Identity technical support about the problem.

AGC4506 (FTL): Exception occurred in *string*!

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4700 (FTL): Wrapper error: *string*.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4701 (ERR): Create process failed (Error code: *number*).

Explanation: This message is returned if the agent controller is unable to create the process. This error can occur if the settings in the authentication property page of the job are wrong.

Action: Check the settings in the authentication property page. Read also the help description of this property page.

AGC4702 (ERR): Process exit code isn't available (Error code: *number*).

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4704 (FTL): Exception occurred in *string*!

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4705 (ERR): Terminate process failed.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4706 (ERR): The function 'fork' failed (Error code: *number string*).

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4708 (ERR): The function 'waitpid' failed.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4709 (ERR): The job path is undefined.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4712 (FTL): Insufficient memory. Function: *string*.

Explanation: The message is returned if the Agent Controller is unable to allocate the memory that is required to process the related workflow. This error can occur if there are too many processes running in parallel or as a consequence of too heavy a system load for DirX Identity.

Action: Reduce the system load and restart the DirX Identity server. If system load does

not seem to be too heavy, inform DirX Identity technical support about the problem.

AGC4713 (ERR): Could not load the configuration file.

Explanation: This message is returned when the agent controller cannot load the configuration file.

Action: Put the Save Mode of the configuration file on Permanent and try the operation again. If the message recurs, check that the configuration file is in the specific work directory. If it is, inform DirX Identity technical support about the problem.

AGC4716 (ERR): The delta base data is wrong.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4717 (INF): The agent *string* was terminated (expiration time).

Explanation: This message indicates that a job has terminated.

Action: None

AGC4718 (INF): The agent *string* was terminated (abort).

Explanation: This message indicates that a job has terminated.

Action: None

AGC4719 (ERR): Cannot get the current working directory (Error code: *number* (*string*)).

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4720 (ERR): Cannot change the current working directory.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4721 (ERR): Cannot reset the current working directory.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4722 (ERR): Cannot find the corresponding USN number.

Explanation: This message indicates that delta handling has failed.

Action: Check the delta handling properties in the related job configuration object. It this information is correct, inform DirX Identity technical support about the problem.

AGC4723 (ERR): Cannot find the corresponding reference file.

Explanation: This message indicates that delta handling has failed.

Action: Check the delta handling properties in the related job configuration object. It this information is correct, inform DirX Identity technical support about the problem.

AGC4725 (ERR): Create pipe failed.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4726 (ERR): Redirecting STDOUT failed.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4727 (ERR): Redirecting STDERR failed.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4728 (ERR): DuplicateHandle failed.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4729 (ERR): Re-redirecting failed.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4730 (ERR): Closing handle failed.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4731 (ERR): The value of the delta reference file has an invalid syntax.

Explanation: This message indicates that delta handling has failed.

Action: Check the delta handling properties in the related job configuration object. If the properties are set correctly, inform DirX Identity technical support about the problem.

AGC4732 (ERR): The agent returns error code: *number*. Details see report or trace file.

Explanation: This message indicates that the agent has returned with error.

Action: Examine the report or trace file in the related status entry to get details about the error.

AGC4733 (INF): The agent *string* shall be terminated.

Explanation: This message indicates that a job shall be terminated.

Action: None

AGC4734 (ERR): Create process failed. A required privilege is not held by the client (Error code: *number*).

Explanation: This message indicates a privilege problem.

Action: Check the user rights in the user manager.

AGC4735 (ERR): Create process failed. Can't find the agent's executable (Error code: *number*).

Explanation: This message indicates that the agent's executable cannot be found.

Action: Check that the agents executable is available in the installation directory. Install the agent correctly if not.

AGC4736 (ERR): The agent cannot find a needed library. The user running the process might not have sufficient rights. (Error code: *number*).

Explanation: This message indicates that the agent controller cannot find a library that it requires.

Action: Consult the DirX Identity documentation regarding prerequisites.

AGC4737 (ERR): The agent was terminated abnormally (Signal: *number*).

Explanation: This message indicates that the agent was terminated abnormally.

Action: Check the logging or trace file of the agent.

AGC4738 (ERR): The agent controller cannot start process with a specific account in secure mode.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4739 (ERR): Error in the crypt library (Error code: *number*).

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4740 (ERR): Error in the shared memory library (Error text: *string*).

Explanation: This message indicates an agent controller problem. If you use encryption and an agent batch file has no executable bit this error can occur.

Action: Check the started agent if the batch file is a valid executable with set excutable bit otherwise inform DirX Identity technical support about the problem.

AGC4742 (ERR): Unscrambling the authentication password failed.

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4743 (ERR): Create process failed. Unknown user name or bad password (Error code: *number*).

Explanation: This message indicates that the user name is unknown or the user password is wrong.

Action: Check the authentication data of the job.

AGC4744 (ERR): Create process failed. The trust relationship between the primary domain and the trusted domain failed (Error code: *number*).

Explanation: The trust relationship between the primary domain and the trusted domain failed.

Action: Check the trust relationship.

AGC4745 (ERR): CreateFile() of *string* failed: *string* (retry: *number*).

Explanation: An internal or system error occurred in the DirX Identity Server.

Action: First check your file system otherwise inform DirX Identity technical support about the problem.

AGC4746 (ERR): ReadFile() in *string* failed: *string*.

Explanation: An internal or system error occurred in the DirX Identity Server.

Action: First check your file system otherwise inform DirX Identity technical support about the problem.

AGC4747 (ERR): CloseFile() of *string* failed: *string*.

Explanation: An internal or system error occurred in the DirX Identity Server.

Action: First check your file system otherwise inform DirX Identity technical support about the problem.

AGC4748 (WAR): CreateFile *string* retry *number*.

Explanation: To open the named file the given number of retries have been processed.

Action: This should happen only in rare situations. If this happens constantly inform technical support about the problem.

AGC4749 (INF): Killing process with PID *number* with signal: *number*

Explanation: This message indicates that a job with the given PID shall be terminated.

Action: None

AGC4750 (ERR): File 'notes.ini' not found.

Explanation: This message indicates that the agent cannot find the file notes.ini.

Action: Check the Notes Client installation and the file notes.ini.

AGC4751 (WAR): Function *string* to retrieve child process data of an agent failed: *string*

Explanation: This message indicates that the agent cannot find a child process of a started agent process.

Action: Check the given Windows error. Child process cannot be terminated.

AGC4752 (WAR): Function *string* on process failed with error *string* (Process name: *string*, PID *number*)

Explanation: This message indicates that the agent cannot terminate a child process of a started agent process.

Action: Check the given Windows error. Child process cannot be terminated.

AGC4753 (INF): The agent *string* with PID *number* was terminated (abort).

Explanation: This message indicates that a job has terminated.

Action: None

AGC4754 (INF): The agent child process *string* with PID *number* was terminated (abort).

Explanation: This message indicates that a job has terminated.

Action: None

AGC4755 (ERR): Create process failed (Error code: *number string*).

Explanation: This message is returned if the agent controller is unable to create the process. This error can occur if the settings in the authentication property page of the job are wrong.

Action: Check the settings in the authentication property page. Read also the help description of this property page.

AGC4756 (ERR): Create pipe failed (Error code: *number string*).

Explanation: This message indicates an internal agent controller problem.

Action: Inform DirX Identity technical support about the problem.

AGC4757 (ERR): The agent returns error code: *value string*. Details see report or trace file.

Explanation: This message indicates that the agent has returned with error.

Action: Examine the report or trace file in the related status entry to get details about the error.

AGC4758 (INF): Found for process *number* following child PIDs: *number* and *number*.

Explanation: This message indicates the PIDs of a started agent process.

Action: None.

16.1.3. Logging Messages (AMQ)

This topic describes the logging messages returned by the DirX Identity Active-MQ client.

AMQ9500 (INF): The client registered a new messaging configuration for *string* (host: *string*, port: *number*).

Explanation: The client obtained the described configuration for connection to ActiveMQ server.

Action: None.

AMQ9501 (WAR): Can not initialize the subscriber. The ActiveMQ connection thread did not respond on time (Error:).

Explanation: A JMS subscriber object can't be initialized. Initialization fails, if an internal subscriber thread can't be created or is not correctly established.

Action: Check the error logs and/or inform the DirX Identity technical support group.

AMQ9502 (WAR): Can't initialize the durable subscriber. The ActiveMQ connection thread did not respond on time (Error: *string*).

Explanation: A JMS subscriber object can't be initialized. Initialization fails, if an internal subscriber thread cannot be created or is not correctly established.

Action: Check the error logs and/or inform the DirX Identity technical support group.

AMQ9503 (WAR): Can't initialize the connection for the subscriber *string*. The ActiveMQ connection thread failed and will retry.

Explanation: A JMS subscriber object was initialized but without a valid connection to a ActiveMQ server. The ActiveMQ connection thread will retry to connect to a configured ActiveMQ server.

Action: Check the error logs and/or inform the DirX Identity technical support group.

AMQ9504 (ERR): Memory allocation failed.

Explanation: Memory allocation has failed.

Action: Reduce the system load. If the problem still exists, inform the DirX Identity technical support group.

AMQ9505 (ERR): The JMS client encountered an exception *string* in *string*, the cause is *string*.

Explanation: The JMS client library has thrown an exception.

Action: Check availability of the JMS broker, try to correct the cause if possible. If the problem persists inform DirX Identity technical support about the problem.

AMQ9506 (ERR): The JMS client could not get property *string* from incoming message.

Explanation: The JMS client library tried to read given property but failed.

Action: Check correctness of the given property on the JMS client side, try to correct the cause if possible. If the problem persists inform DirX Identity technical support about the problem.

AMQ9507 (ERR): Error while accessing ActiveMQ configuration information.

Explanation: An error has occurred while reading the directory information of the ActiveMQ server entry or not all of the information of the ActiveMQ server is available in the directory.

Action: Check the directory data of the ActiveMQ server which is part of the IDS-J server configuration.

AMQ9508 (ERR): Class *string*: Can't initialize object *string*.

Explanation: Initialization problem.

Action: Reduce the system load. If the problem still exists, inform the DirX Identity technical support group.

AMQ9509 (ERR): The client could not parse the configuration file *string*.

Explanation: The client could not parse described configuration file.

Action: Check the existence of the file and check if the client process has the read rights. Check the format of the given file.

AMQ9510 (ERR): The client could not parse the configuration file *string*. An invalid entry for *string* was found.

Explanation: The client could not parse part of the described configuration file.

Action: Check the format of the given file, all mandatory attributes must exists.

AMQ9511 (ERR): The client could not cast Destination object type to Topic object type in *string* for JMS subscriber *string*.

Explanation: The client obtained the improper object type.

Action: Check the error logs and/or inform the DirX Identity technical support group.

AMQ9512 (ERR): The client could not get handle to the internal message object in *string* while converting message to ActiveMQ format.

Explanation: The client could not convert internal message format to ActiveMQ format.

Action: Check the error logs and/or inform the DirX Identity technical support group.

AMQ9513 (ERR): The client could not get header of the internal message object in *string* while converting message to ActiveMQ format.

Explanation: The client could not convert internal message format to ActiveMQ format.

Action: Check the error logs and/or inform the DirX Identity technical support group.

AMQ9514 (ERR): The client could not cast Message object type to BytesMessage object type in *string*.

Explanation: The client obtained the improper object type.

Action: Check the error logs and/or inform the DirX Identity technical support group.

AMQ9515 (ERR): The client detected durable subscription for a non Topic object in *string* for destination *string*.

Explanation: The client obtained the improper object type.

Action: Check the error logs and/or inform the DirX Identity technical support group.

AMQ9516 (ERR): Subscriber thread for destination *string* was terminated due to ICOM problems.

Explanation: An internal subscriber thread was terminated due to an internal thread communication problems.

Action: Check the error logs and/or inform the DirX Identity technical support group.

AMQ9517 (ERR): Synchronization problems occurred while terminating a subscriber thread for destination *string*.

Explanation: An internal subscriber thread detected internal thread synchronization problems.

Action: Check the error logs and/or inform the DirX Identity technical support group.

AMQ9518 (ERR): Error while initializing thread synchronization mechanisms.

Explanation: An error has occurred while initializing the internal thread synchronization mechanisms.

Action: Check the error logs and/or inform the DirX Identity technical support group.

AMQ9519 (WAR): Publisher termination failed. (destination: *string*).

Explanation: Internal publisher thread communication (while terminating a publisher thread) failed.

Action: Inform the DirX Identity technical support group.

AMQ9520 (WAR): Initiating termination of publisher failed (destination: *string*); ICOMSEND error: *number*.

Explanation: Internal publisher thread communication (while terminating publisher thread) has failed.

Action: Inform the DirX Identity technical support group.

AMQ9522 (WAR): Initiating termination of subscriber failed (destination: *string*); ICOMSEND error: *number*.

Explanation: Internal subscriber thread communication (while terminating subscriber thread) has failed.

Action: Inform the DirX Identity technical support group.

AMQ9523 (INF): The client is using *string* as the connection URI for ActiveMQ server(s).

Explanation: The client created the described URI for connection to ActiveMQ server(s).

Action: None.

AMQ9524 (ERR): The client could not locate the file *string* containing a CA certificate which is necessary when using SSL with ActiveMQ.

Explanation: The SSL is enabled for the ActiveMQ client but the file with a CA certificate is missing. The client will be probably unable to connect to the ActiveMQ server.

Action: Check existence and accessibility of the CA certificate file. Disable SSL or create new certificates and keys when necessary.

AMQ9525 (ERR): The client could not locate the file *string* containing a client certificate which is necessary when using SSL with ActiveMQ.

Explanation: The SSL is enabled for the ActiveMQ client but the file with a client certificate is missing. The client will be probably unable to connect to the ActiveMQ server.

Action: Check existence and accessibility of the client certificate file. Disable SSL or create new certificates and keys when necessary.

AMQ9526 (WAR): The client detected that the connection to ActiveMQ server(s) was rebuilt, a message (destination: *string*) cannot be acknowledged.

Explanation: Connection to ActiveMQ server(s) was lost and newly rebuilt. A previously delivered message cannot be acknowledged and will be redelivered.

Action: Check availability of configured ActiveMQ server(s), check network connectivity. If the problem persists, inform the DirX Identity technical support group.

AMQ9527 (WAR): The client could not verify ID of the connection to ActiveMQ server(s), a message (destination: *string*) cannot be acknowledged.

Explanation: Connection to ActiveMQ server(s) was lost and newly rebuilt and no valid connection ID was found. A previously delivered message cannot be acknowledged and will be redelivered.

Action: Check availability of configured ActiveMQ server(s), check network connectivity. If the problem persists, inform the DirX Identity technical support group.

AMQ9528 (ERR): The client could read any valid ActiveMQ server configuration from the given file *string*.

Explanation: The given file does not contain valid information with connection parameters to ActiveMQ server(s).

Action: Check existence and accessibility of the configuration file, check validity of its content.

AMQ9529 (ERR): The client could not publish a message (destination: *string*), timeout was reached.

Explanation: Connection to ActiveMQ server(s) was lost and cannot be newly rebuilt, no message can be published.

Action: Check availability of configured ActiveMQ server(s), check network connectivity. If the problem persists, inform the DirX Identity technical support group.

AMQ9530 (ERR): The client could not create a publisher (destination: *string*), timeout was reached.

Explanation: Connection to ActiveMQ server(s) was lost and cannot be newly rebuilt, no message can be published.

Action: Check availability of configured ActiveMQ server(s), check network connectivity. If the problem persists, inform the DirX Identity technical support group.

AMQ9531 (ERR): The client could not get the password for access to the client key material (keystore) which is necessary when using SSL with ActiveMQ.

Explanation: The SSL is enabled for the ActiveMQ client but the file INST_PATH/ssl/password.properties (INST_PATH/conf/options.ini for Windows Password Listener resp.) does not contain the password for keystore access. The client will be probably unable to connect to the ActiveMQ server.

Action: Check existence of the item 'keystore' in the file INST_PATH/ssl/password.properties ('KeyStorePassword' in the file INST_PATH/conf/options.ini for Windows Password Listener resp.).

AMQ9534 (WAR): *string*

Explanation: A JMS subscriber object was initialized but without a valid connection to a ActiveMQ server. The ActiveMQ connection thread will retry to connect to a configured ActiveMQ server.

Action: Check the error logs and/or inform the DirX Identity technical support group.

AMQ9535 (WAR): The client could not open the configuration file *string* for writing.

Explanation: The client could not open the configuration file for writing (access?).

Action: Check if the client process has the write rights. Check the format of the given file.

AMQ9536 (WAR): The client could write into the configuration file *string*.

Explanation: The client could not write an encrypted value into the configuration file.

Action: Check if the client process has the write rights. Check the format of the given file.

AMQ9537 (ERR): Crypt library encryption failed: *number string*.

Explanation: A value cannot be encrypted.

Action: See the given error code for more information.

AMQ9538 (ERR): *string*.

Explanation: This is an ActiceMQ component error notice. See the text.

Action: See the error text for further information.

AMQ9539 (ERR): Crypt library decryption failed: *number string*.

Explanation: A value cannot be decrypted.

Action: See the given error code for more information.

16.1.4. Logging Messages (CDB)

This topic describes the logging messages returned by the DirX Identity configuration database interface.

CDB3001 (ERR): Cannot determine *string* for *string*: Value *string* is unknown.

Explanation: This message indicates that an attribute (fileFormat/deltaType) contains an illegal value.

Action: DirX Identity Manager normally maintains the correct values. Use the DirX Identity Manager to enter the value again.

CDB3002 (ERR): Search for file object failed: Cannot access object *string*.

Explanation: This message indicates that the DirX Identity configuration database has been corrupted: The job references a file object that does not exist.

Action: DirX Identity Manager normally maintains the integrity of the DirX Identity configuration database. Correct the reference or contact DirX Identity technical support.

CDB3003 (ERR): Cannot determine *string* from *string*: Wrong object class.

Explanation: This message indicates that the DirX Identity configuration database has been corrupted: A reference points to an object of the wrong object class.

Action: DirX Identity Manager normally maintains the integrity of the DirX Identity configuration database. Contact DirX Identity technical support.

CDB3006 (ERR): Cannot determine AgentType for *string*: DxmAgentType = *string* unknown.

Explanation: This message is obsolete in V6.0A10 but is intentionally kept in the message catalog.

Action: If the agent type is unknown, the ConfDB API assumes that the agent is a generic agent.

CDB3008 (ERR): Error on LDAP function *string*: *string*. LDAP resultcode=*number*.

Explanation: This message indicates that an error has occurred during LDAP access to the meta directory store.

Action: Contact DirX Identity technical support.

CDB3009 (ERR): Authentication to LDAP Server failed for user '*string*': Invalid credentials.

Explanation: This message indicates that connection to the LDAP server has failed due to invalid credentials.

Action: Check the bind profile.

CDB3010 (FTL): Memory error detected in *string*.

Explanation: This message indicates that program execution has been aborted because the program has run out of memory.

Action: Close all applications that are not required to be running. Enlarge your RAM storage/swap space.

CDB3013 (ERR): Registration of server *string* refused: Server *string* has already registered.

Explanation: This message indicates that a C++ based DirX Identity Server has already registered with the configuration database. Only one C++ based DirX Identity Server is allowed to register with one configuration database.

Action: Check whether you are already running a C++ based DirX Identity Server on this configuration database. If not, correct the host name in your C++ based DirX Identity Server configuration file and/or your C++ based DirX Identity Server configuration object in the DirX Identity configuration database.

CDB3015 (ERR): Cannot determine messaging port for server *string*: Reference chain is broken.

Explanation: This message indicates that the C++ based DirX Identity Server reference chain @dxmATSServer-DN@dxmService-DN@dxmDataPort cannot determine the messaging port.

Action: Check where the chain is broken and correct the references. Check if a 0 is entered and correct it by entering a valid port number.

CDB3016 (ERR): Cannot determine IP-address for messaging server *string*: Reference chain is broken.

Explanation: This message indicates that the C++ based DirX Identity Server reference chain @dxmATSServer-DN@dxmService-DN@dxmAddress of cannot determine the IP address of the messaging server.

Action: Check where the chain is broken and correct the references.

CDB3018 (ERR): Activity *string* of workflow *string* is not a start activity but has no predecessor.

Explanation: This message indicates that no value has been given for the attribute dxmPredecessor-DN of the current activity. Normally, DirX Identity Manager provides the value.

Action: If the activity is not the start activity, enter the DN of the predecessor activity. Otherwise, mark the activity as the start activity.

CDB3019 (ERR): Predecessor *string* of activity *string* does not exist for workflow *string*.

Explanation: This message is returned when an activity points to a predecessor activity that does not exist.

Action: Correct the dxmPredecessor-DN entry of the activity.

CDB3020 (FTL): Exception occurred in *string*.

Explanation: This message indicates that program execution has been aborted due to a severe internal error.

Action: Contact DirX Identity technical support.

CDB3022 (ERR): Wrong schema version: Expected version *string*, detected version *string*.

Explanation: This message indicates that the schema version of the DirX Identity configuration database is not compatible with the software version.

Action: Migrate the DirX Identity configuration database to the required schema version or install the correct software version.

CDB3023 (ERR): Initialization of SSL-connection failed: Can't access certdb *string*.

Explanation: This message indicates that the certificate database cannot be accessed, which means that the SSL connection cannot be initialized.

Action: Check your installation. Check the certdbpath entry of your initialization (*.ini) file.

CDB3024 (ERR): Failed to initialize SSL client from certdb *string*.

Explanation: This message indicates that the SSL client cannot be initialized from the certificate database.

Action: Check your certificate DB. It may not be compatible with the version of the LDAP SDK.

CDB3025 (ERR): Failed to initialize SSL connection for server *string*, SSL-port *string* from certdb *string*.

Explanation: This message indicates that the SSL connection cannot be established.

Action: Check the certificate DB, host name and SSL port. Check that the certificate DB fits with the private key installed at the server.

CDB3026 (ERR): Error: Can't process activity status data DN=*string*: Object does not exist.

Explanation: This message indicates that the activity status object referenced by a wfStatus object does not exist in the data store.

Action: Remove the files related to this activity status object by hand.

CDB3300 (ERR): Can't initialize LDAP SDK for server *string*:_number_.

Explanation: This message indicates that initialization of the LDAP access has failed.

Action: Contact DirX Identity technical support.

CDB3301 (ERR): Can't connect to LDAP Server *string*:_number_ for user *string*, resultcode=*number*.

Explanation: This message indicates that connection to the LDAP server has failed.

Action: Check the server address and port number. Check whether the LDAP server is running.

CDB3305 (ERR): Lost connection to LDAP server *string*:_number_. Rebind failed for user *string*, resultcode=*number*.

Explanation: This message indicates that the connection to the LDAP server has been lost, and attempts to rebind to the server have failed.

Action: Check your LDAP server and network.

CDB3310 (ERR): Ldap_search_ext_s: Search result not unique: *number* objects found that match the search criteria.

Explanation: This message indicates that an LDAP search has returned more than one object, although exactly one object is expected.

Action: Contact DirX Identity technical support.

CDB3322 (ERR): Ldap_explode_dn: Can't explode DN=*string*. Illegal syntax or out of memory.

Explanation: The ConfDB API is attempting to expand one distinguished name (DN) into its separate components. This action fails presumably because the syntax of the DN is illegal.

Action: Check the reference chains. Perhaps an attribute that does not contain a DN reference is being used to reference another object in the meta directory store.

CDB3323 (INF): Ldap_simple_bind_s failed with resultcode=*number* for server *string*, port *number*, user *string*. Retrying ..

Explanation: The bind to the LDAP server failed. This is seen as temporary failure and the bind will be retried after a defined interval.

Action: None.

CDB3324 (ERR): Lost connection to LDAP server *string*:_number_. Rebind failed, cannot initialize SD.

Explanation: This message indicates that the LDAP server connection has been lost, and attempts to rebind to the server have failed.

Action: Check your LDAP server and network.

CDB3327 (ERR): Can't connect to LDAP SSL-Server *string*:_number_ for user *string*, certdb *string*, resultcode=*number*.

Explanation: This message indicates that connection to the LDAP server has failed.

Action: Check the server address and port number. Check whether the LDAP server is

running. Check certificate DB, host name and SSL port. Check that the certificate DB fits with the private key installed at the server.

CDB3328 (ERR): Lost SSL-connection to LDAP server *string*:_number_. Rebind failed for user *string*, certdb *string*, resultcode=*number*.

Explanation: This message indicates that connection to the LDAP server has been lost, and rebind has failed.

Action: Check the server address and port number. Check whether the LDAP server is running. Check certificate DB, host name and SSL port. Check that the certificate DB fits with the private key installed at the server.

CDB3329 (WAR): Warning: Server is temporarily unavailable while executing LDAP function *string*: *string*. LDAP resultcode=*number*.

Explanation: The LDAP server is busy, unwilling or has shut down.

Action: Try again later.

CDB3330 (ERR): Internal error preparing paged control for first paged search, filter=*string*.

Explanation: An error occurred while initializing the cookie for the first paged search. Should never occur.

Action: Contact DirX Identity technical support.

CDB3331 (ERR): Internal error preparing paged control for next paged search, filter=*string*.

Explanation: An error occurred while initializing the cookie for another paged search. Should never occur.

Action: Contact DirX Identity technical support.

CDB3332 (ERR): Internal error preparing control for paged search, filter=*string*.

Explanation: An error occurred while initializing the cookie for a paged search. Should never occur.

Action: Please, contact DirX Identity technical support.

CDB3333 (WAR): Server is unwilling to perform simple paged search for filter=*string*. Falling back to normal search.

Explanation: The first attempt to read data using simple paged access was not successful. Maybe the limit for parallel simple paged processes is exceeded

Action: The data are retrieved by a normal search. Check your server configuration according to simple paged results, timeout and memory.

CDB3334 (ERR): Server is unwilling to perform retrieval of next page of result entries, filter=*string*. Increase the server's paging policy timeout.

Explanation: The server has returned some result pages and then is unwilling to return further results in simple paged mode.

Action: Increase the paging policy timeout at your server.

CDB3335 (ERR): Error in simple paged result for filter *string*: *string*.

Explanation: An unexpected error occurred during evaluation of the simple paged result controls returned by the server.

Action: Check detailed message. Contact DirX Identity technical support.

CDB3500 (ERR): Can't determine server name: Attribute *string* missing at LDAP object *string*.

Explanation: This message indicates that the LDAP server has no display name, so a check whether the server is registered cannot be performed.

Action: The DirX Identity Manager normally provides the display name. If this error occurs, enter the server name in the dxmDisplayName attribute of the object reported in the error message.

CDB3502 (ERR): Check of hostname failed: Can't retrieve server data for *string*.

Explanation: This message indicates that a check of the server registration has failed because the data cannot be read from the service configuration object in the DirX Identity configuration database.

Action: Check whether your LDAP server is running. If it is, check whether the dxmService-DN link of your C++ based DirX Identity Server configuration object points to the correct service configuration object.

CDB3506 (ERR): Cannot determine attribute *string* for *string*: Attribute is missing.

Explanation: This message indicates that a mandatory property is missing from a configuration object. DirX Identity Manager normally manages the case of missing mandatory properties.

Action: Open the configuration object for editing and enter the missing property.

CDB3507 (ERR): Error(s) occurred during evaluation of agent *string* for job *string*. Processing of AgentData aborted.

Explanation: This message is returned when errors have occurred during the evaluation of the job's agent data.

Action: Correct the agent data according to the preceding error messages.

CDB3508 (ERR): Error(s) occurred during evaluation of activity *string* for workflow *string*. Processing of Activity Data aborted.

Explanation: This message is returned when errors have occurred during the evaluation of the job's activity data.

Action: Correct the activity data according to the preceding error messages.

CDB3509 (ERR): Access to central config object failed.

Explanation: This message indicates that the central configuration object cannot be read.

Action: Check whether your LDAP server is running. Check whether exactly one object with objectClass=dxmCentralConfig exists in your configuration database.

CDB3510 (ERR): Cannot dereference item *string*: Attribute *string* does not exist.

Explanation: This message indicates that an attribute that does not exist at the object has been requested within a tag in a reference chain. The tag may contain an incorrect DN reference or the attribute is missing from the object.

Action: Correct the tag that contains the reference or add the missing attribute to the object.

CDB3511 (ERR): Cannot dereference item *string*: Search failed for *string*.

Explanation: This message indicates that an object that does not exist in the DirX Identity configuration database has been referenced within a tag in a reference chain.

Action: Correct the tag or the object reference.

CDB3512 (ERR): Evaluation of dxmCmdLine=*string* failed for Job *string*.

Explanation: This message indicates that the command line contains tags that cannot be evaluated.

Action: Correct the errors occurring during tag evaluation according to the preceding error messages.

CDB3513 (ERR): Processing of file item object *string* aborted with error(s).

Explanation: This message indicates that errors have occurred during evaluation of the job's file data.

Action: Correct the file data according to the preceding error messages.

CDB3515 (ERR): Processing of *string* data aborted with error(s) for job *string*.

Explanation: This message indicates that errors have occurred during evaluation of the job's file data.

Action: Correct the file data according to the preceding error messages.

CDB3521 (ERR): Can't open file *string* for writing.

Explanation: This message indicate that the ConfDB API cannot open a *.ini file, a Tcl script, or an attribute configuration file for writing.

Action: Check file protections and pathnames.

CDB3522 (ERR): Error(s) occurred during write of file *string*. Check disk space and network.

Explanation: This message indicates that an error has occurred during the write of a *.ini file, a Tcl script, or an attribute configuration file.

Action: Check your system resources: The file system could be full or the network could be unavailable.

CDB3528 (ERR): Error: Can't add DN=*string* of activity status object to workflow status object *string*.

Explanation: This message indicates that an activity status object has been created but its DN cannot be added to the workflow status object.

Action: Check your LDAP server. Delete the corrupted status objects by hand.

CDB3529 (ERR): Error(s) occurred during read of WFStatusData Object *string*.

Explanation: This message indicates that the DirX Identity service cannot read the WFStatus object from the DirX Identity configuration database.

Action: Check the LDAP server and the preceding LDAP error message.

CDB3530 (ERR): Can't create workflow status object in folder *string* for workflow *string*.

Explanation: This message indicates that the DirX Identity service cannot create the workflow status object.

Action: Check the LDAP server and the preceding LDAP error message.

CDB3531 (ERR): Error: Creation of workflow status container object with DN=*string* failed.

Explanation: This message indicates that the DirX Identity service cannot create the workflow status folder object.

Action: Check the LDAP server and the preceding LDAP error message.

CDB3532 (ERR): Error: Creation of activity status object *string* failed.

Explanation: This message indicates that the DirX Identity service cannot create the activity status folder object.

Action: Check the LDAP server and the preceding LDAP error message.

CDB3533 (ERR): Error during read of StatusData Object, DN=*string*. *string* failed.

Explanation: This message indicates that errors have occurred during the read of a status data object.

Action: Check the LDAP server and the preceding LDAP error message.

CDB3534 (ERR): Error during modification of StatusData Object *string*. *string* failed.

Explanation: This message indicates that errors have occurred during the modification of a status data object.

Action: Check the LDAP server and the preceding LDAP error message.

CDB3535 (ERR): Error in attribute *string* of StatusData Object *string*: Multiple occurrence of value *string*, 2nd occurrence ignored.

Explanation: This message indicates that a duplicate file entry has been detected. Since the files must be unique, this error should not occur.

Action: Contact DirX Identity technical support.

CDB3536 (ERR): Error *string* in line *number* at position *number*: End of tag is missing.

Explanation: This message indicates that a tag has been entered without an end-of-tag construct.

Action: Check the tag syntax and add the missing end-of-tag.

CDB3537 (ERR): Error *string* at position *number*: End of tag is missing.

Explanation: This message indicates that a tag has been entered without an end-of-tag construct.

Action: Check the tag syntax and add the missing end-of-tag.

CDB3538 (ERR): Error *string* in line *number* at position *number*: Unknown content type *string* in tag.

Explanation: This message indicates that a tag has been detected with unknown content.

Action: Check the contents of the tag for typing errors.

CDB3539 (ERR): Error *string* at position *number*: Unknown content type *string* in tag.

Explanation: This message indicates that a tag has been detected with unknown content.

Action: Check the contents of the tag for typing errors.

CDB3540 (ERR): Error *string* in line *number* in expression *string*, at position *number*: *string*.

Explanation: This message indicates that a syntax error has been detected during the scan of a tag. The error position is given relative to the start of the tag.

Action: Check the error and correct it according to the given hint.

CDB3541 (ERR): Error *string* in expression *string*, at position *number*: *string*.

Explanation: This message indicates that a syntax error has been detected during the scan of a tag. The error position is given relative to the start of the tag.

Action: Check the error and correct it according to the given hint.

CDB3542 (ERR): Error *string* in line *number* at position *number*: *string*.

Explanation: This message indicates that a syntax error has been detected during scan of a loop tag. The error position is given relative to the start of the line.

Action: Check the error and correct it according to the given hint.

CDB3543 (ERR): Error *string* in line *number* at position *number*: *string* is empty, *string* not allowed in this context.

Explanation: This message indicates that a tag references a loopvar or a file in the wrong context.

Action: Correct the tag. Loopvars are only allowed within loop statements.

CDB3544 (ERR): Error *string* at position *number*: *string* is empty, *string* not allowed in this context.

Explanation: This message indicates that a tag references a loopvar or a file in the wrong context.

Action: Correct the tag. Loopvars are only allowed within loop statements.

CDB3545 (ERR): Error *string* at position *number*: *string*.

Explanation: This message indicates that a syntax error has been detected during the scan of a loop tag.

Action: Check the error and correct it according to the given hint.

CDB3546 (ERR): Error *string* in line *number* at position *number*: *string* statement is missing, *string* unexpected.

Explanation: This message indicates that a syntax error has been detected during the scan of a loop tag.

Action: Check the error and correct it according to the given hint.

CDB3547 (ERR): Error *string* at position *number*: *string* statement is missing, *string* unexpected.

Explanation: This message indicates that a syntax error has been detected during the scan of a loop tag.

Action: Check the error and correct it according to the hint given.

CDB3548 (ERR): Error *string* while evaluating expression *string* in line *number* at position *number*: *string*.

Explanation: This message indicates that the tag syntax is correct, but the evaluation of tag data has failed (e.g., errors have occurred while accessing the DirX Identity configuration database).

Action: Correct the tag or the DirX Identity configuration database references. If the error recurs, contact DirX Identity technical support.

CDB3549 (ERR): Error *string* while evaluating expression *string* at position *number*: *string*.

Explanation: This message indicates that the tag syntax is correct, but the evaluation of tag data has failed (e.g., errors have occurred while accessing the DirX Identity configuration database).

Action: Correct tag or DirX Identity configuration database references. If the error recurs, contact DirX Identity technical support.

CDB3555 (ERR): Initialization of path handling failed, job *string*: Can’t determine *string* for

current activity *string*.

Explanation: This message indicates that the DirX Identity configuration database has become corrupted.

Action: DirX Identity Manager normally maintains configuration database integrity. Try to correct the missing attribute values and/or broken DN references. If this action does not work, contact DirX Identity technical support.

CDB3556 (ERR): Initialization of path handling failed, job *string*: Can't determine *string* for current workflow *string*.

Explanation: This message indicates that the DirX Identity configuration database has become corrupted.

Action: DirX Identity Manager normally maintains configuration database integrity. Try to correct the missing attribute values and/or broken DN references. If this action does not work, contact DirX Identity technical support.

CDB3557 (ERR): Processing of channel mapping *string* failed, job *string*: Can't determine *string* of creating activity.

Explanation: This message indicates that the DirX Identity configuration database has become corrupted.

Action: DirX Identity Manager normally maintains configuration database integrity. Try to correct the missing attribute values and/or broken DN references. If this action does not work, contact DirX Identity technical support.

CDB3558 (ERR): Processing of channel mapping *string* failed, job *string*: Can't determine filename from channel.

Explanation: This message indicates that the DirX Identity configuration database has become corrupted.

Action: DirX Identity Manager normally maintains configuration database integrity. Try to correct the missing attribute values and/or broken DN references. If this does not work, contact DirX Identity technical support.

CDB3560 (ERR): Can't retrieve messaging server data from path=*string*: Error(s) during search.

Explanation: This message indicates that the messaging server data cannot be retrieved. The DirX Identity configuration database could be corrupted or the LDAP server could be down.

Action: Check your LDAP server. Try to correct the missing attribute values and/or broken DN references. If this action does not work, contact DirX Identity technical support.

CDB3561 (ERR): Error while retrieving messaging server data for C++ based DirX Identity Server=*string*: Can't access display name of related server.

Explanation: This message indicates that the messaging server data cannot be

retrieved. The DirX Identity configuration database could be corrupted or the LDAP server could be down.

Action: Check your LDAP server. Try to correct the missing attribute values and/or broken DN references. If this does not work, contact DirX Identity technical support.

CDB3562 (ERR): Can't delete file '*string*': *string*

Explanation: Cannot delete file.

Action: Check detailed message and analyze preceding error messages.

CDB3563 (INF): Found *number* entries to delete in status area (filter: *string*)

Explanation: Searching for status data that should be deleted found the given number of entries.

Action: None.

CDB3564 (ERR): Error(s) occurred during access of activity *string* for workflow *string*. Processing of timeout for schedule *string* aborted.

Explanation: This message indicates that the data for an activity referenced by a workflow object cannot be read from the DirX Identity configuration database.

Action: Check your LDAP server. Try to correct the missing attribute values and/or broken DN references. If this action does not work, contact DirX Identity technical support.

CDB3565 (ERR): Error(s) occurred during evaluation of timeout of activity *string* for workflow *string*. Processing of timeout for schedule *string* aborted.

Explanation: This message indicates that the reference chain activity@dxmRunObject-DN@dxmTimeOut cannot be resolved. It is possible that dxmRunObject-DN is broken or that dxmTimeOut does not exist.

Action: Check your LDAP server. Try to correct the missing attribute values and/or broken DN references. If this action does not work, contact DirX Identity technical support.

CDB3566 (ERR): Conversion of attribute *string* = *string* of *string* failed. Format error, GENERALIZED_TIME expected.

Explanation: This message indicates that the conversion of an attribute from generalized time format to seconds has failed due to incorrect format.

Action: This error should not occur since the format is checked by the server. Contact DirX Identity technical support.

CDB3567 (ERR): Can't determine display name of related C++ based DirX Identity Server for schedule *string*.

Explanation: This message indicates that the reference chain mss_server@dxmWorkflow-DN@dxmMetahubSyncServer-DN@dxmDisplayName is broken.

Action: Check your LDAP server. Try to correct the missing attribute values and/or broken DN references. If this action does not work, contact DirX Identity technical support.

CDB3570 (ERR): Syntax error in dxmDeltaInfoItems *string* of job *string*: *string*.

Explanation: The delta info items are expected in format index;text. This message indicates that the semicolon (;) is missing, so that the index cannot be extracted.

Action: The DirX Identity Manager normally provides these items, so this error should not occur. Contact DirX Identity technical support.

CDB3571 (WAR): No matching entry found for dxmDeltaInfoSelected=*string* in dxmDeltaInfoItems of job *string*.

Explanation: dxmDeltaInfoSelected contains an index to select on entry in dxmDeltaInfoItems. This message indicates that this index does not exist in the items.

Action: DirX Identity Manager provides the items and dxmDeltaInfoSelected, so this error should not occur. Contact DirX Identity technical support.

CDB3572 (WAR): No matching entry found for dxmDeltaInfoSelected=*string* of job *string*: No dxmDeltaInfoItems exist.

Explanation: dxmDeltaInfoSelected contains an index to select on entry in dxmDeltaInfoItems. This message indicates that attribute dxmDeltaInfoItems is empty.

Action: DirX Identity Manager provides the items and dxmDeltaInfoSelected, so an error should not occur. Contact DirX Identity technical support.

CDB3573 (ERR): Error in dxmDeltaInfoItems *string* of job *string*: Can't extract filename from item *string*.

Explanation: dxmDeltaInfoItems must contain a file name in case the delta type is file. This message indicates that this file name is missing.

Action: DirX Identity Manager provides the items and dxmDeltaInfoSelected, so this error should not occur. Contact DirX Identity technical support.

CDB3576 (ERR): Can't access path *string*: *string*. Creation failed.

Explanation: This message indicates that the ConfDB API has attempted to create a path in the file system but creation has failed.

Action: Check protections and filenames.

CDB3578 (WAR): *string* failed for workflow status object, workflow *string*. No object found with wfInstId=*string*. Trying to create ..

Explanation: The ConfDB API has attempted to write data to a workflow status object that does not exist.

Action: ConfDB has created a workflow status object without an expiration date. You must delete it by hand.

CDB3579 (WAR): *string* failed for activity status object, DN=*string*. Object does not exist. Trying to create ..

Explanation: This message indicates that the ConfDB API has attempted to write data to an activity status object that does not exist.

Action: The ConfDB API has created an activity status object without an expiration date. You must delete it by hand.

CDB3580 (WAR): Creation of activity status object with DN=*string* failed, workflow status DN=*string* does not exist. Trying to create ..

Explanation: This message indicates that the ConfDB API has attempted to write data to an activity status object that does not exist. The attempt to create the object has failed, since the related workflow status object does not exist.

Action: The ConfDB API has created a workflow status object and an activity status object without expiration dates. You must delete them by hand.

CDB3583 (ERR): Processing of channel mapping failed for job *string*: Cannot read parent *string* of channel *string*.

Explanation: This message indicates that a parent object of an input channel object cannot be read from the meta directory store.

Action: Check server and network.

CDB3584 (ERR): Processing of channel mapping failed for channel *string* of job *string*: Reading of activities of workflow *string* failed.

Explanation: This message indicates that the activities of the current workflow cannot be read from the meta directory store.

Action: Check server and network.

CDB3585 (ERR): Processing of channel mapping failed for job *string*: Workflow *string* has multiple activities (*string* and *string*) referencing the same job *string*.

Explanation: This message indicates that a workflow has multiple activities that reference the same job. The workflow must not have more than one activity for the same job.

Action: Correct your workflow design.

CDB3586 (ERR): Cannot determine creating activity of InputChannel *string* of job *string*: Creating job *string* is not referenced by any activity of workflow *string*.

Explanation: The workflow must have a creating activity with a job as the parent of the input channel.

Action: Correct your workflow design.

CDB3587 (ERR): Processing of channel mapping failed for channel *string*, job *string*: Can't determine *string* of creating activity.

Explanation: This message indicates that the DirX Identity configuration database has become corrupted.

Action: DirX Identity Manager normally maintains configuration database integrity. Try to correct the missing attribute values/broken DN references. If this action does not work, contact DirX Identity technical support.

CDB3588 (ERR): Processing of channel mapping failed for channel *string*, job *string*: Can't read channel data.

Explanation: This message indicates that the DirX Identity configuration database has become corrupted.

Action: DirX Identity Manager normally maintains configuration database integrity. Try to correct the missing attribute values/broken DN references. If this action does not work, contact DirX Identity technical support.

CDB3589 (ERR): Processing of channel mapping failed for channel *string*, job *string*: Can't determine filename from channel.

Explanation: This message indicates that the DirX Identity configuration database has become corrupted.

Action: DirX Identity Manager normally maintains configuration database integrity. Try to correct the missing attribute values/broken DN references. If this action does not work, contact DirX Identity technical support.

CDB3590 (ERR): Processing of channel mapping failed for job *string*: Cannot read channel *string*.

Explanation: This message indicates that the DirX Identity configuration database has become corrupted.

Action: DirX Identity Manager normally maintains configuration database integrity. Try to correct the missing attribute values/broken DN references. If this action does not work, contact DirX Identity technical support.

CDB3591 (ERR): Processing of channel mapping failed for job *string*: Cannot determine type of connected directory for channel *string*.

Explanation: This message indicates that the DirX Identity configuration database has become corrupted.

Action: DirX Identity Manager normally maintains configuration database integrity. Try to correct the missing attribute values/broken DN references. If this action does not work, contact DirX Identity technical support.

CDB3592 (ERR): Error *string* in line *number* at position *number*: Can't determine DN of start tag for expression: *string*.

Explanation: This message indicates that a special function is referenced by a tag whose DN cannot be determined.

Action: This error should not occur since the Tcl code is created by the DirX Identity Manager. Contact DirX Identity technical support.

CDB3593 (ERR): Error *string* at position *number*: Can't determine DN of start tag for expression: *string*.

Explanation: This message indicates that a special function is referenced by a tag whose DN cannot be determined.

Action: This error should not occur since the Tcl code is created by the DirX Identity Manager. Contact DirX Identity technical support.

CDB3594 (ERR): Error: Duplicate display name *string* detected in mapping functions searched in tree *string*.

Explanation: This message indicates that duplicate display names were detected while searching mapping functions.

Action: Correct display names to get unique mapping function references.

CDB3595 (ERR): Error in job *string*: Search of mapping functions failed in tree *string*: *string*.

Explanation: This message indicates that a search for mapping functions was aborted with errors.

Action: Check server and network connections.

CDB3596 (ERR): Error in shared path entry of MSS *string* for creating server *string*: *string* is missing.

Explanation: This message indicates that the shared path entry on a C++ based DirX Identity Server is incomplete for the creating server.

Action: Correct the shared path definitions with DirX Identity Manager.

CDB3597 (WAR): Source path *string* on server *string* cannot be accessed by shared path *string* from server *string*. Conversion was done using dxmSharedPath=*string*.

Explanation: The path created by making use of the shared path entry is not accessible.

Action: Check for network connections. Correct the shared path definitions with the DirX Identity Manager.

CDB3599 (ERR): Initialization of path handling failed, job *string*: Illegal syntax for DN=*string* in attribute sharedPath in MSS server *string*.

Explanation: This message indicates that the shared path entry on a C++ based DirX Identity Server has a server entry in illegal DN syntax.

Action: Correct the shared path definitions with DirX Identity Manager.

CDB3605 (ERR): Can't create DN of WFStatusFolder for workflow *string*: Can't determine DisplayName of folder DN=*string*.

Explanation: This message indicates that the DirX Identity configuration database has become corrupted.

Action: DirX Identity Manager normally maintains configuration database integrity. Try to correct the missing attribute values/broken DN references. If this action does not

work, contact DirX Identity technical support.

CDB3606 (ERR): *string* failed for workflow status object, workflow *string*. Error while searching one unique object with wfInstId=*string*.

Explanation: This message indicates that an error occurred while searching workflow status object.

Action: Check the LDAP server and the preceding LDAP error message. It is possible that the wfInstId is not unique. If this is the case, delete one status data object by hand.

CDB3607 (ERR): *string* failed: No WFStatusData object found in folder *string* for workflow *string*.

Explanation: This message indicates that a search for WFStatus objects in the DirX Identity configuration database has failed.

Action: Check the LDAP server and preceding LDAP error messages.

CDB3608 (ERR): Read of account DN=*string* failed.

Explanation: This message indicates that the account object cannot be read. This error is unlikely since your bind was successful.

Action: Check whether your LDAP server is running.

CDB3609 (ERR): Can’t extract serial number from keyOwnerPSE=*string* of account DN=*string*.

Explanation: This message indicates that the keyOwnerPSE multivalue attribute contains an entry with wrong syntax. The serial number cannot be extracted or is not an integer.

Action: Correct the entry using the tool dirxgenpse.

CDB3610 (ERR): Reading of activity data failed for *string*.

Explanation: Can’t read the activity data of the current activity. GetJobData failed.

Action: Check whether your LDAP server is running.

CDB3611 (ERR): Reading of associated MSS data *string* failed, referenced from activity *string*.

Explanation: Can’t read the C++ based DirX Identity Server configuration data for the current job.

Action: Check whether your LDAP server is running. Check the dxmMetahubSyncServer-DN reference of the current activity.

CDB3612 (ERR): Reading of associated MSS data failed. Reference from activity *string* is missing.

Explanation: Can’t read the C++ based DirX Identity Server configuration data for the current job.

Action: Correct the dxmMetahubSyncServer-DN reference of the current activity.

CDB3613 (ERR): Format error in serial number *string* from keyOwnerPSE=*string* of account DN=*string*: Hexadecimal format [0-9a-f] expected.

Explanation: This message indicates that an incorrect character has been specified in the serial number of a pse. Only hexadecimal characters 0-9a-f are allowed in a serial number of a pse.

Action: Correct the keyOwnerPSE attribute of the account.

CDB3614 (ERR): Can't extract serial number from keyOwnerPSE=*string* of account DN=*string*: Dxc_initialize failed with resultcode=*number*.

Explanation: This message indicates that the dirxcrypt module cannot be initialized.

Action: Check the result code and the existence of files/directories for dirxcrypt.

CDB3615 (ERR): Cannot determine DN of parent workflow status object. Parent/child links are not set.

Explanation: The DN of the parent workflow status object cannot be determined.

Action: The navigation from parent to child and vice-versa will not work in the status view.

CDB3616 (ERR): Cannot determine DN of parent activity status object, rdn=*string*, from parent workflow status object *string*.

Explanation: The DN of the parent activity status object cannot be determined from dxmActivityStatusData-DN.

Action: The navigation from parent to child and vice-versa will not work in the status view.

CDB3617 (WAR): Cannot link WFStatusDataObject *string* path=*string* to its parent activity, parentWfName=*string*, parentActivityName=*string*.

Explanation: This message indicates that an error has occurred which prevents linking the WFStatusDataObject to its parent activity.

Action: The navigation from parent to child and vice-versa will not work in the status view.

CDB3618 (WAR): Cannot remove link to ChildWorkflowStatusData *string* from ActivityStatusObject *string*.

Explanation: This message indicates that an error has occurred while deleting a WFStatusObject, and so the reference from its parent activity cannot be removed.

Action: When trying to navigate from the parent to this child, a broken link will occur.

CDB3619 (WAR): Cannot remove link to ParentActivityStatusData *string* from WorkflowStatusObject *string*.

Explanation: This message indicates that an error has occurred while deleting an ActivityStatusObject, and so the reference from its child workflow cannot be removed.

Action: When trying to navigate from the child to this parent, a broken link will occur.

CDB3620 (ERR): Error in activity '*string*': Maximum recursion depth=20 exceeded. Check for loops in hierarchical workflow structure.

Explanation: This message indicates that a loop has been detected while processing the activities of a hierarchical workflow.

Action: Check workflow structure and correct runObject references.

CDB3621 (ERR): Error in activity '*string*': Can't read referenced job/workflow with RunObject='*string*'.

Explanation: This message indicates that the job or workflow referenced by the activity cannot be read.

Action: Check RunObject-DN reference and server connection.

CDB3622 (ERR): Error in activity '*string*': Object referenced by RunObject '*string*' is of wrong type. Has to be job or workflow.

Explanation: This message indicates that the object referenced by the activity is neither a job nor a workflow.

Action: Correct the activity's RunObject-DN reference.

CDB3623 (WAR): Activity *string* of workflow *string* is an end activity but is predecessor of activity *string*.

Explanation: This message indicates that an activity is marked as end activity but has a successor.

Action: Remove predecessor link or remove end activity flag.

CDB3624 (ERR): Activity *string* of workflow *string* is not an end activity but has no successor.

Explanation: This message indicates that an activity is not marked as an end activity but it does not have a successor.

Action: Set predecessor link of another activity to this activity or set end activity flag.

CDB3625 (ERR): Activity *string* of workflow *string* is a start activity but has a predecessor.

Explanation: This message indicates that an activity is marked as a start activity but has a predecessor link set.

Action: Remove predecessor link or remove start activity flag.

CDB3626 (ERR): Error(s) occurred during font conversion to latin1 of file *string*.

Explanation: A file of encoding iso8859-1 cannot be converted from UTF8 to latin1.

Action: As a workaround, set dxmEncoding of the file to utf-8.

CDB3627 (ERR): Error(s) during read of secure port for MSS DN=*string*: Reference chain is broken.

Explanation: The secure port for the MSS cannot be read. Thus, Java agents with password decryption cannot be run.

Action: Check reference @dxmService-DN from MSS to System. Enter value for secure port (e.g. 1818) for that system.

CDB3628 (ERR): Fatal error during scan of XML description *string* (line *number*, char *number*): *string*.

Explanation: The denoted XML description is erroneous.

Action: Correct the error or call Your support.

CDB3629 (ERR): Error during scan of XML description *string* (line *number*, char *number*): *string*.

Explanation: The denoted XML description is erroneous.

Action: Correct the error or call Your support.

CDB3630 (WAR): Error during scan of XML description *string* (line *number*, char *number*): *string*.

Explanation: The denoted XML description is erroneous.

Action: Correct the error or call Your support.

CDB3631 (ERR): Error during initialization of XML parser! Message: *string*.

Explanation: XERCES could not be initialized. Parsing of XML-files is not possible. (Should never occur).

Action: Call Your support.

CDB3632 (ERR): Missing *string* attribute in statistic entry tag of *string* (line *number*).

Explanation: A mandatory attribute is missing in the complete status description.

Action: Call Your support.

CDB3633 (ERR): Error during termination of XML parser! Message: *string*.

Explanation: XERCES could not be terminated. (Should never occur).

Action: Call Your support.

CDB3634 (ERR): Can’t create statistics for WFStatusData DN=*string*: No ActivityStatusData exist.

Explanation: There are no activity status data objects assigned via the dxmActivityStatusData link. Thus, no statistics can be collected from the activities.

Action: Check status region and calculate statistics manually in case activity status objects do exist.

CDB3635 (ERR): Can't create statistics for WFStatusData DN=*string*: Maximum recursion depths exceeded. Check for loops.

Explanation: The workflow statistics cannot be computed since a referenced activity status data object does not exist.

Action: Calculate the statistics manually from the statistics of the single activities.

CDB3636 (ERR): Can't create statistics for WFStatusData of activity status data DN=*string*: WFStatusData not found.

Explanation: The workflow statistics cannot be computed since an error occurred while reading a related activity status data object.

Action: Calculate the statistics manually from the statistics of the single activities.

CDB3637 (WAR): Can't evaluate disable statistics flag for ActivityStatusData DN=*string*: WFStatusData DN=*string* has no link to the original workflow. Including statistics.

Explanation: For reading the disable statistics flag, the assigned activity has to be determined. This failed, thus the statistics will be included.

Action: Re-calculate the statistics manually from the statistics of the single activities in case the inclusion is not intended.

CDB3638 (WAR): Can't evaluate disable statistics flag for ActivityStatusData DN=*string*: Workflow *string* referenced from WFStatusData DN=*string* does not exist. Including statistics.

Explanation: For reading the disable statistics flag, the assigned activity has to be determined. This failed, thus the statistics will be included.

Action: Re-calculate the statistics manually from the statistics of the single activities in case the inclusion is not intended.

CDB3639 (WAR): Can't evaluate disable statistics flag for ActivityStatusData DN=*string*: Workflow *string* referenced from WFStatusData DN=*string* cannot be read. Including statistics.

Explanation: For reading the disable statistics flag, the assigned activity has to be determined. This failed, thus the statistics will be included.

Action: Re-calculate the statistics manually from the statistics of the single activities in case the inclusion is not intended.

CDB3640 (WAR): Can't evaluate disable statistics flag for ActivityStatusData DN=*string*: Activity *string* of workflow *string* does not exist. Including statistics.

Explanation: For reading the disable statistics flag, the assigned activity has to be determined. This failed, thus the statistics will be included.

Action: Re-calculate the statistics manually from the statistics of the single activities in

case the inclusion is not intended.

CDB3641 (WAR): Can't evaluate disable statistics flag for ActivityStatusData DN=*string*: Activity *string* of workflow *string* cannot be read. Including statistics.

Explanation: For reading the disable statistics flag, the assigned activity has to be determined. This failed, thus the statistics will be included.

Action: Re-calculate the statistics manually from the statistics of the single activities in case the inclusion is not intended.

CDB3642 (WAR): Can't evaluate disable statistics flag for ActivityStatusData DN=*string*: Activity with display name *string* of workflow *string* does not exist. Including statistics.

Explanation: For reading the disable statistics flag, the assigned activity has to be determined. This failed, thus the statistics will be included.

Action: Re-calculate the statistics manually from the statistics of the single activities in case the inclusion is not intended.

CDB3643 (WAR): Can't evaluate disable statistics flag for ActivityStatusData DN=*string*: Workflow *string* has no activities. Including statistics.

Explanation: For reading the disable statistics flag, the assigned activity has to be determined. This failed, thus the statistics will be included.

Action: Re-calculate the statistics manually from the statistics of the single activities in case the inclusion is not intended.

CDB3644 (ERR): Error(s) occurred while writing end data / statistics to workflow status DN=*string*.

Explanation: The workflow statistics could not be written. Maybe the server connection is broken.

Action: Check messages of LDAP-API module. Calculate the statistics manually from the statistics of the single activities.

CDB3645 (ERR): Can't create statistics for parent activity DN=*string* of WFStatusData DN=*string*: Parent activity does not exist.

Explanation: The workflow statistics could not be written to the parent activity, since the dxmParentActivityStatusData-DN link is broken.

Action: Correct the statistics manually.

CDB3646 (ERR): Can't create statistics for parent activity DN=*string* of WFStatusData DN=*string*: Parent activity cannot be read.

Explanation: The workflow statistics could not be written to the parent activity, since the dxmParentActivityStatusData-DN cannot be read.

Action: Check messages of LDAP-API module. Correct the statistics manually.

CDB3647 (ERR): Error(s) occurred while writing statistics to parent activity status DN=*string*

of workflow status DN=*string*.

Explanation: The workflow statistics could not be written to the parent activity. Modification of the parent object failed.

Action: Check messages of LDAP-API module. Correct the statistics manually.

CDB3648 (ERR): Invalid range of schedule *string* (Invalid attribute dxmScheduleRange = *string*). Please correct the range of the schedule using the DirX Identity Manager.

Explanation: The range of the schedule is invalid. The related schedule will not be executed as long as the problem exists.

Action: This error should not occur except when modified with tools other than DirX Identity Manager. Correct the schedule as indicated in the message.

CDB3649 (ERR): Open of file *string* successful after *string* retries. Error message was *string*.

Explanation: An internal or system error occurred in the C++-based Server.

Action: First check your file system otherwise inform technical support about the problem.

CDB3650 (ERR): Open of file *string* failed: *string*. Retrying....

Explanation: To open the named file failed. The server retries to open the file again.

Action: This should happen only in rare situations. If this happens constantly inform technical support about the problem.

CDB3651 (WAR): Can't create search filter for expired WFStatus locations: *string*. Using default filter.

Explanation: Could not determine all MSS referencing the same message service for deletion of wfStatus entries. Using mss supplied with bind credentials.

Action: Check detailed message and analyze preceding ldap errors messages.

CDB3652 (ERR): Cannot determine SOAP port for server *string*: Reference chain is broken.

Explanation: This message indicates that the C++ based DirX Identity Server reference chain @dxmSoapService-Dn@dxmDataPort cannot determine the SOAP port.

Action: Check where the chain is broken and correct the references.

CDB3653 (ERR): Cannot determine SOAP SSL port for server *string*: Reference chain is broken.

Explanation: This message indicates that the C++ based DirX Identity Server reference chain @dxmSoapService-Dn@dxmSecurePort cannot determine the SOAP SSL port.

Action: Check where the chain is broken and correct the references.

CDB3654 (ERR): Cannot determine secure messaging port for server *string*.

Explanation: This message indicates that the C++ based DirX Identity Server reference

chain @dxmATSServer-DN@dxmService-DN@dxmSecurePort cannot determine the secure messaging port.

Action: Check where the chain is broken and correct the references. Check if the default value 0 is entered and correct it by a valid port number.

CDB3655 (ERR): Cannot determine port number for ActiveMQ server *string*.

Explanation: This message indicates that the JMS server data reference chain @dxmService-DN@dxmDataPort or @dxmService-DN@dxmSecurePort cannot determine the port number of the ActiveMQ server.

Action: Check if a service object is referenced by the configuration object and if a valid port number is configured.

CDB3656 (ERR): Cannot determine IP address for ActiveMQ server *string*.

Explanation: This message indicates that the JMS server data reference chain @dxmService-DN@dxmSystem-DN@dxmAddress cannot determine the IP address of the ActiveMQ server.

Action: Check if a service object is referenced by the configuration object and if a valid IP address is configured.

CDB3657 (ERR): Can’t retrieve ActiveMQ server data. Search did not result in an entry: Filter *string*.

Explanation: This message indicates that the ActiveMQ server data cannot be retrieved. The DirX Identity configuration database could be corrupted or the LDAP server could be down.

Action: Check your LDAP server. Try to correct the missing attribute values and/or broken DN references. If this does not work, contact DirX Identity technical support.

CDB3658 (ERR): Can’t retrieve container for ActiveMQ server data. Search did not result in an entry: DN *string*.

Explanation: This message indicates that the container for ActiveMQ server data cannot be retrieved. The DirX Identity configuration database could be corrupted or the LDAP server could be down.

Action: Check your LDAP server. If this does not work, contact DirX Identity technical support.

CDB3659 (WAR): The failover parameter *string* is invalid, it contains too many spaces or other forbidden characters. Ignoring it.

Explanation: This message indicates that the container for ActiveMQ server data contains an invalid value. The DirX Identity configuration database could be corrupted.

Action: Check the configured value for failover transport. Correct it if necessary. Check your LDAP server. If this does not work, contact DirX Identity technical support.

CDB3660 (ERR): Schedule *string* discarded due to data inconsistency problems with respect to the workflow to be scheduled.

Explanation: This message indicates that a schedule will not be processed due to data inconsistency problems with respect to the workflow to be scheduled.

Action: If the cause of the problem is not because the configuration database is unavailable, analyze error messages in event log or ASCII log files to check whether they indicate inconsistencies of the configuration database with respect to definitions regarding the workflow to be scheduled. If so, then fix the inconsistencies. Otherwise, inform DirX Identity technical support about the problem.

16.1.5. Logging Messages (CMR)

This topic describes the logging messages returned by the DirX Identity SPML connector service configuration.

CMR6200 (WAR): Not all mandatory key items are present in section '*string*' of connector: url-postfix, sharedlibrary, connector-ini, logging-no

Explanation: Not all mandatory key items are present in the mentioned section of connector.

Action: Update the configuration.

CMR6201 (WAR): Cannot find ini file for connector '*string*'; ini filename *string*, current working directory: *string*. Connector ignored.

Explanation: The ini file for the mentioned connector cannot be found.

Action: Update the configuration.

CMR6202 (WAR): No associated connector found for the SOAP URL '*string*' (reason: *string*). Connection ignored.

Explanation: The URL does not contain a configured connector.

Action: Check the configuration.

CMR6203 (WAR): Cannot bind to configuration database.

Explanation: Connector configuration cannot be read due to LDAP problems.

Action: Check the LDAP connection.

CMR6204 (WAR): Cannot find own entry in the configuration database.

Explanation: The DirX Identity server entry cannot be found in the configuration database.

Action: Check the configuration.

CMR6205 (WAR): Cannot find any configured connector in the configuration database.

Explanation: No connector entry found underneath the server entry.

Action: Check the configuration.

CMR6206 (WAR): Configuration for connector '*string*' incomplete. Connector ignored.

Explanation: Data is incomplete or no ini file found.

Action: Check the configuration.

CMR6207 (WAR): Evaluating of references failed. Configuration for connector '*string*' incomplete. Connector ignored.

Explanation: Evaluating references in the ini template failed.

Action: Check the configuration.

CMR6208 (WAR): Preprocessing of ini file template failed. Configuration for connector '*string*' incomplete. Connector ignored.

Explanation: Evaluating references in the ini template failed.

Action: Check the configuration.

CMR6209 (WAR): Processing of ini file template failed. Configuration for connector '*string*' incomplete. Connector ignored.

Explanation: Evaluating references in the ini template failed.

Action: Check the configuration.

CMR6210 (ERR): Read of account DN=*string* failed.

Explanation: The account object cannot be read. This is unlikely since your bind was successful.

Action: Check whether your LDAP server is running.

CMR6211 (ERR): Can’t extract serial number from keyOwnerPSE=*string* of account DN=*string*.

Explanation: The keyOwnerPSE multivalue attribute contains an entry with wrong syntax. The serial number cannot be extracted or is not an integer.

Action: Correct the entry using the tool dirxgenpse.

CMR6212 (ERR): Format error in serial number *string* from keyOwnerPSE=*string* of account DN=*string*: hexadecimal format [0-9a-f] expected.

Explanation: Only hexadecimal characters 0-9a-f are allowed in a serial number of a pse

Action: Correct the keyOwnerPSE attribute of the account.

CMR6213 (ERR): Can’t extract serial number from keyOwnerPSE=*string* of account DN=*string*: dxc_initialize failed with resultcode=*number*.

Explanation: The dirxcrypt module cannot be initialized.

Action: Check resultcode and the existence of files/directories for dirxcrypt.

CMR6214 (ERR): Read of object with filter '*string*' failed.

Explanation: The object cannot be read. This is unlikely since your bind was successful.

Action: Check whether your LDAP server is running.

CMR6215 (WAR): Duplicate configuration for connector '*string*' (URL path name: *string*). Connector ignored.

Explanation: A given connector cannot be configured twice.

Action: Change the configuration for the connector.

CMR6216 (WAR): No PIN and pse file provided. No encryption is possible for filters/connectors.

Explanation: If a connector is configured with checked PSE flag then a PIN and the PSE file must be given.

Action: Change the configuration for the connector.

CMR6217 (WAR): Error *string* while reading private key file '*string*'. No encryption is possible for filters/connectors.

Explanation: An error occurred while trying to read the PSE file.

Action: Depending on the error message provide a readable PSE file or contact customer support.

CMR6218 (ERR): Cannot find filter definition for filter '*string*'.

Explanation: A filter definition cannot be found.

Action: Change the filter configuration.

CMR6219 (ERR): Cannot find filter list definition for filter '*string*'.

Explanation: A filter list definition cannot be found.

Action: Change the filter definition.

CMR6220 (WAR): No connector found in the configuration database.

Explanation: No connector entry found underneath the server entry.

Action: Check the configuration.

CMR6221 (INF): Connector '*string*' with type '*string*' and path part of URL '*string*' is enabled.

Explanation: The named connector has been configured successfully.

Action: No action.

CMR6223 (ERR): Internal error in function '*string*' listname: '*string*'.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

CMR6224 (WAR): Filter '*string*' ignored: '*string*'.

Explanation: The named filter is ignored as indicated by error message.

Action: Correct the filter definition.

CMR6225 (WAR): Cannot find central configuration entry in the configuration database.

Explanation: The DirX Identity central configuration entry cannot be found in the configuration database.

Action: Check the configuration.

16.1.6. Logging Messages (CSR)

This topic describes the logging messages returned by the DirX Identity SPML connector service server.

CSR6600 (ERR): Exception occurred. Description: *string*.

Explanation: Internal error occurred.

Action: Inform DirX Identity technical support about the problem.

CSR6601 (ERR): Transmission protocol error. Description: *string*.

Explanation: Error transmitting data using network protocol. Details about the error can be found in the description.

Action: Probable cause of the error is in the configuration of the server environment, network problems or similar.

CSR6602 (WAR): SPML error. Description: *string*.

Explanation: Server received an SPML request that was invalid or not supported.

Action: None.

CSR6603 (WAR): Connector server cannot get current config data, request cannot be handled.

Explanation: Communication with the CMGR subcomponent failed. May be an internal error.

Action: Inspect other log messages and if you do not find any cause why there could be problems accessing configuration info for the connector server contact the technical support.

CSR6604 (WAR): Invalid URL or connector name in the request: *string*.

Explanation: SPML request contains the name of the destination connector. This name is encoded in the URL. Either the URL has not a valid form or there is no connector with the given name currently running.

Action: Make sure that the URL is correctly formed and that the connector is configured in the server.

CSR6605 (WAR): Connector *string* could not receive SPML request. Sending FAULT back.

Explanation: SPML requests are directed to connectors. It may happen that some connector cannot receive requests. It may be suspended or overwhelmed with requests or not responding (blocked). In that case the requests cannot be handled and FAULT responses are returned by the server.

Action: Inspect the operation of the connector, restart the server if needed. You may switch on or increase the logging level for the connector to see what happens.

CSR6606 (ERR): Connector name *string* is used more than one time in the configuration. Ignoring the second and other connectors with the same name.

Explanation: Two or more connectors are configured with the given name. This is an error since the second connector of the same name is ignored (is not started in the server).

Action: Check the connector server configuration and remove the duplicity.

CSR6608 (ERR): TermLibrary method could not be loaded from shared library *string*.

Explanation: Every shared component library contains exactly one component class. This component class must implement the interface for component classes (see the documentation for the connector developers). One of the method which must be implemented is the TermLibrary method. The server could not find this method in the given library.

Action: Put the correct library into the configuration.

CSR6609 (WAR): Warning: *string*

Explanation: A minor problem occurred, the server continues running.

Action: Look at the text of the message to find out more about the reason of the warning.

CSR6610 (INF): *string*

Explanation: Message describes main events during the normal connector server operation. See the message text for details.

Action: None.

CSR6611 (WAR): Connector server SPML warning. Description: *string*.

Explanation: Some unexpected or potentially erroneous state was detected in the connector server subsystem handling the SPML objects. In some cases this could point to an internal error.

Action: For developers only.

CSR6612 (WAR): SOAP RPC exception. Description: *string*.

Explanation: Some problem was detected when using SOAP RPC call between the components of the system. It may be caused by unavailability of a component, bad configuration of the communication interfaces or by an internal error.

Action: According to the text of the message check whether the problem is in your local configuration or inform DirX Identity technical support when it is an internal error.

CSR6613 (ERR): Connector name '*string*' is forbidden to use. Rename the connector or otherwise it will be ignored by the server.

Explanation: Some built-in server component use this name already. You must choose some other name for your connector in order to be able to use it.

Action: Change the name of the connector in your configuration data.

CSR6614 (ERR): Connector '*string*' is ignored because there are too many connectors configured.

Explanation: There is an internal server limit for the number of the connectors. If this limit is reached the server runs normally only the remaining connectors are ignored. The name of the ignored connector you can find in the text of this error message. There also can be such situation that there are more slots for connectors needed than they are configured. This is caused by the fact that after runtime server reconfiguration the old and the new connector exist concurrently for some short period of time (so called 'dynamic shortage of connector slots').

Action: Remove some connectors from the configuration of the server and restart or reconfigure the server. In case of a dynamic shortage (see explanation) you must restart the server.

CSR6615 (ERR): Connector table internal error.

Explanation: The server contains an internal structure named 'connector table'. An internal error in this structure occurred.

Action: Inform DirX Identity technical support about the problem.

CSR6616 (ERR): Connector '*string*' could not load one of its shared libraries. Connector will be stopped. Details: *string*

Explanation: C++ Connectors are components inserted into the C++ connector server. They are contained in shared libraries whose name you can find in the configuration, under the connector of the above name. The server encountered some error loading this library (see also other errors reported). Alternatively there could be an error while running the InitLibrary method inside the shared library. See the description of connectors. Alternatively the same could happen for a filter that is assigned to the above named connector. This error does not cause any problem to the server except that the connector is suspended and will not run until the server will be restarted or reconfigured. First the cause of the error must be found and removed, see the action description.

Action: Read the description about connectors and filters pluggable into the C++ connector server. Examine the connector and filter entries written in the server log to see the cause of the error while loading the library. Remove the cause (may be wrong configuration, missing shared library on the correct path or various other reasons). Finally, restart or reconfigure the C++ Identity server to make the connector available.

CSR6617 (ERR): Component library '*string*' threw an unknown exception. Connector '*string*' will be stopped.

Explanation: C++ connectors are components inserted into the C++ connector server. They are contained in shared libraries whose name you can find in the configuration, under the connector of the above name. They can have associated filters which are also contained in such shared libraries. Their names you can also find in the configuration. The DirX Identity Connector API specification defines what type of exceptions (special C++ events) can be thrown from within these libraries. This message means that the specified component library (associated with the specified connector) generated an unknown type of exception. The server cannot decode the type and meaning of the exception. This is considered internal error of the connector. The connector (and associated filters) will be suspended until the DirX Identity C++ server will be restarted or reconfigured in order to minimize the influence of a (possibly) critical state of this connector/filter library to the whole server's health. While the connector is suspended it refuses all requests coming to it.

Action: This error may be encountered when exceptional / unexpected operations or states occur. Look in the log files for other messages from the specified connector, try to find out which operation did the connector currently do in the time when the error occurred. If there is no connector specified in this message you can find the connector or filter by the name of the shared library or by the component which logged this message. In every case, however, there is (among possible other problems) an error inside the connector's code - it should be robust and check for every possible error situation and react in the way how the DirX Identity Connector API defines. In such a case this error would not occur. To remove this error from the shared library you must be able to analyze the library source code and change it. In case you are able to do it you should ensure that the exception handling of the library is in compliance with the API specification. Otherwise contact the developer of the above specified shared library.

CSR6618 (WAR): JMX error. Description: *string*.

Explanation: Server received a JMX request that was invalid or not supported.

Action: None.

16.1.7. Logging Messages (CUT)

This topic describes the logging messages returned by the DirX Identity SPML connector service utilities.

CUT6800 (FTL): Fatal exception occurred. Description: *string*.

Explanation: Fatal internal error occurred.

Action: Inform DirX Identity technical support about the problem.

CUT6801 (FTL): Insufficient memory or allocation error at *string*.

Explanation: Connector Server is unable to allocate new memory. This event can occur if there are too many processes running in parallel or if the system load is too heavy.

Action: Reduce the system load and restart the server. If the system load does not seem

to be too heavy, inform technical support about the problem.

CUT6802 (ERR): SOAP error occurred: *string*.

Explanation: Some error occurred in communication over SOAP protocol. The details of the error message show the cause of the error.

Action: See the message details to fix the cause. SOAP errors can occur because of an invalid configuration of the server (invalid port, SSL parameters, etc. ; e. g. 'Can't read key file' can also mean wrong password for keystore). Also various network errors can cause this error.

CUT6803 (WAR): SPML class warning: *string*.

Explanation: Exceptional (but not fatal) situation occurs in the connector server internal SPML classes.

Action: Most probably an internal error, inform the technical support.

CUT6804 (ERR): Exception occurred: *string*.

Explanation: An exception occurred during the connector server operation.

Action: See the message details to fix the cause of the problem.

CUT6805 (WAR): Thread framework discarded *string* messages from the ICOM queue.

Explanation: The C++ connector server serves the requests that are coming from the other DirX Identity components to carry out various provisioning or other operations (like reconfiguration or other server management operations). These requests are stored in the internal queues of the server called ICOM. In some exceptional cases (e.g. when the server is shutdown without long enough timeout) there could be a need to discards those internally stored requests without processing them or even notifying the client. In most cases this situation is harmless because the client will recognize that the operation was not acknowledged by the server and repeat the request again. This warning does not mean that anything is or gets corrupt but it means that there arose some exceptional situation which should not be happening on a regular basis.

Action: This warning may help in the diagnosis of a complex problem but does not require any specific action. It is exceptional and if happening regularly then there is a probability of some problem like badly set configuration parameters for the C++ server (timeouts etc.)

16.1.8. Logging Messages (CWR)

This topic describes the logging messages returned by the DirX Identity SPML connector service wrapper.

CWR6400 (FTL): Exception occurred in *string*. (ErrorMessage: *string*).

Explanation: An internal error has occurred in the connector wrapper.

Action: Inform DirX Identity technical support about the problem.

CWR6401 (FTL): Cannot initalize mutex '*string*' (Error code: *number*).

Explanation: An internal error has occurred in the connector wrapper.

Action: Inform DirX Identity technical support about the problem.

CWR6402 (FTL): Cannot destroy mutex '*string*' (Error code: *number*).

Explanation: An internal error has occurred in the connector wrapper.

Action: Inform DirX Identity technical support about the problem.

CWR6403 (FTL): Cannot lock mutex (Error code: *number*).

Explanation: An internal error has occurred in the connector wrapper.

Action: Inform DirX Identity technical support about the problem.

CWR6404 (FTL): Cannot unlock mutex (Error code: *number*).

Explanation: An internal error has occurred in the connector wrapper.

Action: Inform DirX Identity technical support about the problem.

CWR6406 (WAR): *string*.

Explanation: This is a warning message.

Action: See the text for further information.

16.1.9. Logging Messages (DXC)

This topic describes the error messages returned by the cryptographic library (Java code, for C code see SCE).

ERR(DXC001): DXC_ERROR

Explanation: This message indicates an internal library problem. This error should not occur in normal circumstances.

Action: Inform DirX Identity technical support about the problem.

ERR(DXC002): DXC_PARAMETER_ERROR

Explanation: This message indicates an internal library problem. This error should not occur in normal circumstances.

Action: Inform DirX Identity technical support about the problem.

ERR(DXC005): DXC_FILE_ERROR

Explanation: The random file ‘dxc_rand.dat’ does not exist or is not writable.

Action: Reinstall DirX Identity or change the access rights.

ERR(DXC010): DXC_LOGGING_ERROR

Explanation: This message indicates a general logging issue.

Action: Inform DirX Identity technical support about the problem.

ERR(DXC022): DXC_DATA_HEADER_FAILURE

Explanation: This message indicates that the library was not able to verify the header part of encrypted data. Maybe the data is not encrypted or contains escaped characters.

Action: Check and update your data or inform DirX Identity technical support about the problem.

ERR(DXC023): DXC_BASE64_FAILURE

Explanation: This message indicates that the library was not able to convert the data.

Action: Check and update your PSE environment or inform DirX Identity technical support about the problem.

ERR(DXC024): DXC_UNSUITABLE_KEY

Explanation: This message indicates that the library has not the right private keys to decrypt data. Maybe the data is encrypted with an outdated PSE.

Action: Check and update your PSE environment.

ERR(DXC043): DXC_EXPORTKEY_EXPIRED

Explanation: This message indicates that the library has not the right private keys to decrypt data. This is a timeout issue.

Action: Check and increment the KeyGetTimeout value of the DirX Identity server object.

16.1.10. Logging Messages (F1_)

This topic describes the logging messages returned by the DirX Identity C++-based Filter 1.

F1_2250 (INF): *string* (file: *string*, line: *number*)

Explanation: This is an informal notice.

Action: See the text for further information.

F1_2251 (INF): *string* (file: *string*, line: *number*)

Explanation: This is a verbose informal notice.

Action: See the text for further information.

F1_2252 (WAR): *string* (file: *string*, line: *number*)

Explanation: This is a warning notice.

Action: See the error text for further information.

F1_2253 (ERR): *string* (file: *string*, line: *number*)

Explanation: This is an error notice.

Action: See the error text for further information.

F1_2254 (FTL): *string* (file: *string*, line: *number*)

Explanation: This is a severe error notice.

Action: See the error text for further information.

16.1.11. Logging Messages (FSV)

This topic describes the logging messages returned by the DirX Identity file service interface.

FSV6700 (FTL): Exception occurred in *string*!

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6701 (ERR): No operation code in message.

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6702 (ERR): Wrong operation code in message.

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6703 (ERR): File Service aborted.

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6704 (ERR): GetTopic failed.

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6705 (ERR): No folder in message.

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6706 (ERR): No source file in message.

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6707 (ERR): No destination file in message.

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6708 (ERR): No data in message.

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6709 (ERR): Error on file or folder *string*. Function: *string*. Context: *string*.

Explanation: A file problem.

Action: Check your system resources: The file system could be full. Check file protections

FSV6710 (ERR): Cannot create message.

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6711 (ERR): Insufficient memory.

Explanation: This message is returned if the file service is unable to allocate the amount of memory that is required to process the file transfer. This error can occur if there are too many processes running in parallel or as a consequence of too heavy system load for DirX Identity.

Action: Reduce the system load and restart the DirX Identity C server. If the system load does not seem to be too heavy, then inform technical support about the problem.

FSV6712 (ERR): Function *string* not supported.

Explanation: This message is returned if file service attempts to process a request for an operation that is not supported.

Action: Inform support about the problem.

FSV6713 (ERR): Crypt error. Function: *string*, rtcode: *number*, error string: *string*

Explanation: This message is returned if the file service is unable to encrypt/decrypt a message.

Action: Check the returncode, the error string and the encryption log file.

FSV6714 (ERR): Decryption of message failed. Reason: Unsuitable key.

Explanation: This message is returned if different keys for file encryption/decryption have been used. This error occurs when the password of server_admin is changed while a workflow is running.

Action: Start the workflow again.

FSV6715 (ERR): opendir() of *string* failed: *string* (errno: *number*)

Explanation: A directory cannot be opened.

Action: Check your file system.

FSV6716 (WAR): The parameter max message length was too high, using max value *number*, consider usage of a lower value.

Explanation: The parameter max message length is used as a length for a single chunk of a file to be transferred. The higher value might cause errors caused by insufficient memory especially when working with encrypted files.

Action: Consider usage of values lower than 100 MB.

FSV6717 (ERR): Deletion of file/folder failed: folder *string* file *string*. Resulting path is too long.

Explanation: A directory/file cannot be deleted.

Action: Check your file system.

FSV6800 (FTL): Exception occurred in *string*!

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6801 (ERR): Insufficient memory.

Explanation: This message is returned if the file service is unable to allocate the amount of memory that is required to process the file transfer. This error can occur if there are too many processes running in parallel or as a consequence of too heavy a system load for DirX Identity.

Action: Reduce the system load and restart the DirX Identity C server. If the system load does not seem to be too heavy, then inform DirX Identity technical support about the problem.

FSV6802 (ERR): GetTopic failed.

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6803 (ERR): Cannot create message.

Explanation: Internal file service problem.

Action: Inform DirX Identity technical support about the problem.

FSV6804 (WAR): Abort for instance: *string*
DirX Identity server name: *string*
return value: *number* (*string*).

Explanation: Publishing of a message fails. The file service sends an abort message to

the specified DirX Identity C server.

Action: Inform DirX Identity technical support about the problem.

FSV6805 (WAR): EXIT Object *value*, Function *string*, return value is *number* (*string*).

Explanation: A file service function returns with an error.

Action: Check the return code.

16.1.12. Logging Messages (JDBF)

This topic describes the logging messages returned by the DirX Identity Delta Filter.

WAR(JDBF0): At *string1*: delta configuration warning: *string2*: *string3*

Explanation: There has been a minor error in configuration, for example duplicated operational attributes.

Action: Review and correct if desired.

FTL(JDBF1): At *string1*: delta configuration error: *string2*: *string3*

Explanation: There has been a fatal configuration error. A required operational attribute may be missing, or may have an unacceptable value. The report will carry more information about what is wrong.

Action: Correct configuration and re-try.

WAR(JDBF2): At *string1*: delta warning: *string2*: *string3*

Explanation: At present there is no use of this warning.

Action: None.

ERR(JDBF3): At *string1*: delta error: *string2*: *string3*

Explanation: A serious error has occurred in the processing of a delta filter operation. This may be due to a problem in creating or handling a synopsis file. Or it may be due to an unspecified data source (successor). Or it may be due to an unacceptable operational attribute supplied in the request, including the supply of an unknown synopsis identifier. Or it may be due to a failure in the data source (successor) that is more serious than an error return. Or it can be an unexpected system error. The report will carry more information about what has gone wrong.

Action: If appropriate, correct the request and re-try; it may be necessary to try the operation without a filter, or (in the event of synopsis file problems) to investigate possible permission or corruption problems.

FTL(JDBF4): At *string1*: delta system error: *string2*: *string3*

Explanation: There is a serious system problem, which is most likely to be in the access of synopsis files. It could also be due to a quite unexpected result from the data source. The report will carry more information about what has gone wrong.

Action: Investigate the problem in the supporting environment, or the source of data.

16.1.13. Logging Messages (MSS)

This topic describes the logging messages returned by the DirX Identity server.

MSS2003 (INF): Unknown message type.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an unknown message type in the message.

Action: Verify whether the versions of your client and server software are compatible.

MSS2004 (INF): Stub initialize failure (internal error: *string* (*number*)).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2005 (INF): Instance not found.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message because the addressed component no longer exists. This event can occur after the server is restarted.

Action: None.

MSS2006 (INF): Instance does not exist.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message because the addressed component no longer exists. This event can occur after the server is restarted.

Action: None.

MSS2007 (INF): Field DXMObjectType invalid.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to unexpected settings in the message.

Action: Verify whether the versions of your client and server software are compatible.

MSS2008 (INF): Failure to start a thread (internal error).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2010 (INF): Field JMSReplyTo not set.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to unexpected settings in the message.

Action: Verify whether the versions of your client and server software are compatible.

MSS2011 (INF): Field DXMObjectType not set.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to unexpected settings in the message.

Action: Verify whether the versions of your client and server software are compatible.

MSS2012 (INF): Field DXMOperation not set.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to unexpected settings in the message.

Action: Verify whether the versions of your client and server software are compatible.

MSS2013 (INF): Exception occurred in *string* (*string*).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2014 (INF): No version information found (field DXMVersion).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to unexpected settings in the message.

Action: Verify whether the versions of your client and server software are compatible.

MSS2015 (INF): Version of message incompatible.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an incompatible version string.

Action: Upgrade your server or your client (DirX Identity Manager).

MSS2016 (FTL): Unable to bind to meta directory (rtc=*string*).

Explanation: The DirX Identity server (C++-based) cannot bind to the meta directory.

Action: Check the bind arguments in the server configuration file.

MSS2017 (INF): The DirX Identity server (C++-based) is configured not to start this type of component

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to unexpected settings in the message.

Action: Verify whether the versions of your client and server software are compatible.

MSS2018 (INF): No instance Id found (field DXMInstId).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to unexpected settings in the message.

Action: Verify whether the versions of your client and server software are compatible.

MSS2023 (INF): Maximum number of threads for workflows and activities reached.

Explanation: The DirX Identity server (C++-based) cannot start an additional thread for processing the workflow. The limit has been reached.

Action: Increase the limit in the central configuration. If you think that some threads are not terminating properly, inform technical support about this issue.

MSS2029 (ERR): MSSsession::dispatch(): message with msg ID '*string*' with invalid fields: no InstID nor CorrID given.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2030 (ERR): Exception in ThreadProc_CmssWorker() (ulExit=*number* rtc=*string* step=*number*).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2032 (WAR): Attempt to start workflow/activity '*string*' a second time.

Explanation: A workflow should be started but the same workflow/activity is still running.

Action: Wait until the first workflow terminates before starting the same workflow/activity.

MSS2041 (FTL): Exception in MSSsession::initializeSvr() (rtc: *number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2050 (ERR): Cannot initialize the *string* subscriber (session id: *string*, topic: *string*).

Explanation: An error has occurred in the DirX Identity server (C++-based) during subscriber initialization.

Action: If you use WebSphere MQ messaging see the WebSphere MQ reason code for further information.

MSS2053 (FTL): Bind to secure port *number* failed.

Explanation: DirX Identity server (C++-based) can not bind to its secure port.

Action: Check if any other program is using the given port. Reason is given in previous error messages.

MSS2062 (INF): Unknown command line argument.

Explanation: An unknown command line argument was given to the server.

Action: Change the command line.

MSS2065 (FTL): Insufficient memory.

Explanation: DirX Identity server (C++-based) is unable to allocate memory that is required to process the workflow. This event can occur if there are too many processes running in parallel or if the system load is too heavy for DirX Identity.

Action: Reduce the system load and restart the DirX Identity server (C++-based). If the system load does not seem to be too heavy, then inform technical support about the problem.

MSS2066 (INF): Cast error (internal error).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2067 (FTL): Cannot change current directory to: *string* (*number*).

Explanation: The DirX Identity server (C++-based) cannot changed to the indicated directory.

Action: Change the attributes of the indicated directory.

MSS2068 (ERR): Unexpected exception in DirX Identity server (C++-based).

Explanation: An exception occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2069 (ERR): Internal error: registering of atexit() function failed.

Explanation: The DirX Identity server (C++-based) cannot register a function.

Action: Inform technical support about the problem.

MSS2070 (INF): Writing to stdout failed.

Explanation: The DirX Identity server (C++-based) cannot write to stdout.

Action: Check your file redirections of the server.

MSS2071 (INF): Synchronizing with NT service process failed.

Explanation: The DirX Identity server (C++-based) cannot synchronize with the service process due to a system error.

Action: Check your system to see if kernel event objects can no longer be properly used.

MSS2072 (INF): Field DXMNewSvrPwd not set.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to unexpected settings in the message.

Action: Verify whether the versions of your client and server software are compatible.

MSS2073 (INF): Change bind password failed with error code *number*.

Explanation: The DirX Identity server (C++-based) cannot change its bind password. See

the given error code for further information.

Action: See action of given error code.

MSS2074 (FTL): Fork failed (errno = *number*).

Explanation: The DirX Identity server (C++-based) cannot fork the process due to a system error.

Action: Check the errno field and if necessary, inform technical support about the problem.

MSS2080 (FTL): Data in local configuration file invalid (rtc = *string*).

Explanation: The DirX Identity server (C++-based) cannot start because the ini file settings are invalid.

Action: Check the server ini file settings.

MSS2085 (ERR): Unscrambling of bind password failed.

Explanation: The DirX Identity server (C++-based) cannot reset its bind password.

Action: Check the password.

MSS2088 (FTL): Failure reading local configuration file (errno = *number*).

Explanation: The DirX Identity server (C++-based) cannot start because the configuration file cannot be read.

Action: Check the errno field.

MSS2091 (FTL): WSAStartup() failed (rtc=*number*).

Explanation: A system error has occurred in the Windows socket library.

Action: See the system code for further information.

MSS2094 (ERR): CmssStub::setMsgDefaultProperties(): no msgid in message (msgtype: *string*).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2097 (ERR): CmssSvr::copyCreateMsg(): no msgid in message (msgtype: *string*).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2104 (ERR): Exception in CmssWorker::loadModule() (rtc=*string* step=*number*).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2106 (ERR): Exception in CmssWorker::unloadModule() (step=*number*).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2110 (FTL): gethostbyname() failed: *number*.

Explanation: The DirX Identity server (C++-based) cannot detect the IP address of the machine due to a system error.

Action: Check the indicated system error code.

MSS2111 (FTL): gethostname() failed: *number*.

Explanation: The DirX Identity server (C++-based) cannot detect the host name of the machine due to a system error.

Action: Check the indicated system error code.

MSS2140 (ERR): Internal error: attach to memory mapped file failed.

Explanation: The DirX Identity server (C++-based) cannot attach to the memory mapped file.

Action: Inform technical support about the problem.

MSS2141 (ERR): Internal error: attach to memory mapped file failed (extend function).

Explanation: The DirX Identity server (C++-based) cannot attach to the memory mapped file.

Action: Inform technical support about the problem.

MSS2142 (ERR): Internal error: reading of memory mapped file failed.

Explanation: The DirX Identity server (C++-based) cannot read the memory mapped file.

Action: Inform technical support about the problem.

MSS2144 (ERR): Encryption mode mismatch (locally no; configuration database not NONE).

Explanation: The DirX Identity server (C++-based) has been configured in its ini file to run not in encryption mode. The configuration database has been configured to run encryption mode other then NONE.

Action: Correct the configuration.

MSS2145 (ERR): Encryption mode mismatch (locally yes; configuration database NONE).

Explanation: The DirX Identity server (C++-based) has been configured in its ini file to run in encryption mode. The configuration database has been configured to run encryption mode NONE.

Action: Correct the configuration.

MSS2146 (ERR): Cannot save the new bind password in the configuration file.

Explanation: The DirX Identity server (C++-based) cannot save the new bind password. A restart of the server will fail.

Action: Check the mode of the ini file.

MSS2147 (ERR): No bind password is given.

Explanation: The DirX Identity server (C++-based) has not get any bind password via prompt dialog or in the ini file.

Action: Check the ini file and restart the server.

MSS2148 (ERR): Update of bind password failed. See the given error code for more information.

Explanation: The DirX Identity server (C++-based) cannot update its bind password due to a failure.

Action: See the given error code for more information.

MSS2202 (WAR): Unable to unbind from meta directory (rtc=*string*).

Explanation: The DirX Identity server (C++-based) cannot unbind from the meta directory.

Action: See the return code for further information.

MSS2203 (WAR): Cannot register DirX Identity server (C++-based) *string* (*string*).

Explanation: The DirX Identity server (C++-based) cannot register itself to the meta directory. This event can occur after a server crash.

Action: None.

MSS2204 (FTL): Cannot register DirX Identity server (C++-based) *string* (*string*).

Explanation: The DirX Identity server (C++-based) cannot register itself to the meta directory. This event occurs if two instances with the same server name are configured and one instance is already running or if the configuration in the meta directory and the server configuration file are addressing different machines.

Action: Check the configuration file and the settings in the meta directory.

MSS2205 (ERR): Cannot unregister DirX Identity server (C++-based) *string* (*string*).

Explanation: The DirX Identity server (C++-based) cannot unregister itself from the meta directory.

Action: See the return code for further information.

MSS2206 (ERR): Exception in MSSsession::waitForResponses() (IEvent: *value*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2207 (ERR): Failure in MSSsession::waitForResponses() (rtc: *number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2208 (ERR): Exception in MSSsession::findObject() (Id: *string*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2209 (FTL): Exception in MSSsession::initialize() (rtc: *number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2210 (ERR): Exception in MSSsession::dispatch() (Id: *string* rtc: *number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2211 (FTL): Cannot initialize mutex '*string*' (err: *number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2212 (FTL): Cannot destroy mutex '*string*' (err: *number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2213 (FTL): Cannot lock mutex (err: *number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2214 (FTL): Cannot unlock mutex (err: *number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2215 (ERR): Cannot start session with a *string* subscriber (session id: *string*, topic: *string*, err: *number*).

Explanation: An error has occurred in the DirX Identity server (C++-based) during subscriber initialization.

Action: If you use WebSphere MQ messaging see the WebSphere MQ reason code for further information.

MSS2216 (ERR): Cannot start session with a *string* subscriber (session id: *string*, err: *number*).

Explanation: An error has occurred in the DirX Identity server (C++-based) during subscriber initialization.

Action: If you use WebSphere MQ messaging see the WebSphere MQ reason code for further information.

MSS2217 (ERR): Dispatch of instId *string* failed (msgtype=*string*, rtc=*number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2218 (FTL): IComInit() failed (rtc=*number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2219 (ERR): IComAttach() failed (rtc=*number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2220 (INF): *string* subscriber initialized (session id: *string*, topic: *string*).

Explanation: An message subscriber has been initialized.

Action: None.

MSS2221 (INF): Caught signal *number*.

Explanation: A signal has been caught in the DirX Identity server (C++-based).

Action: None.

MSS2222 (INF): Terminating server...

Explanation: The DirX Identity server (C++-based) is terminating.

Action: None.

MSS2223 (INF): Starting subscriptions ...

Explanation: The DirX Identity server (C++-based) is on-line and can receive messages.

Action: None.

MSS2224 (INF): MSSsession::attach(): lost msg *string* expired!

Explanation: The DirX Identity server (C++-based) has discarded an expired message.

Action: None.

MSS2226 (WAR): MSSrequest::waitForResponse(0x*value*,0x*value*): no message received (state was 0x*value*, reset to 0x*value*).

Explanation: The DirX Identity server (C++-based) has not yet received a response for a message.

Action: None.

MSS2227 (WAR): MSSsession::dispatch(): no object in list for msg id '*string*' (InstID: *string*, CorrID: *string*, msgtype: *string*).

Explanation: The DirX Identity server (C++-based) cannot dispatch a received message to an instance.

Action: None.

MSS2229 (ERR): No request-obj for request with msg id *string* (InstID: *string*, CorrID: *string*, msgtype: *string*)

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2230 (ERR): Listener: Distorted message:
msgtype: *number*,
version: *string*,
replyto: *string*
message discarded.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an missing setting in the message.

Action: Inform technical support about the problem.

MSS2231 (INF): Listener: No instance ID given in message:
msgtype: *number*,
replyto: *string*
message discarded.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an missing setting in the message.

Action: None.

MSS2232 (INF): Listener error response message:
In inst ID: *string*,
In msg ID: *string*,
In msgtype: *string*,
Out return: *number*,
Out text: *string*.

Explanation: The DirX Identity server (C++-based) cannot properly process the incoming message. See 'Out text' for the reason.

Action: None.

MSS2233 (ERR): Listener: cannot reply to message:
msgtype: *string*,
instID: *string*,
return: *number*
message discarded.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an missing setting in the message.

Action: Inform technical support about the problem.

MSS2234 (ERR): Exception in CmssListener::onMessage() (Id: *string* msgtype: *string*).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2235 (INF): Monitor thread started (pagesize=*number*).

Explanation: The DirX Identity server (C++-based) has started the monitor component.

Action: None.

MSS2236 (INF): *string* started.

Explanation: The DirX Identity server (C++-based) has started the component.

Action: None.

MSS2237 (INF): Thread *string* terminated.

Explanation: The named thread terminated.

Action: None.

MSS2238 (INF): DirX Identity server (C++-based) is running.
PID=*number*, MaxNumberThreads=*number*,
Encryption mode=*string*, KeyTransferPort=*number*,
SSL to LDAP=*string*,
Connector SPML service=*string*, port=*number*, SSL port=*number*, SSL=*string*
Configuration for first ActiveMQ host=*string*, port=*number*, SSL=*string*

Explanation: The DirX Identity server (C++-based) is on line and ready to serve requests.

Action: None.

MSS2239 (INF): DirX Identity server (C++-based) is shutting down...

Explanation: The DirX Identity server (C++-based) is shutting down and will not service new incoming requests.

Action: None.

MSS2240 (ERR): *string* terminated with errors (exit code=*string*, rtc=*string*).

Explanation: The DirX Identity server (C++-based) terminated with the indicated error.

Action: See the error code for further actions.

MSS2242 (INF): Cleanup started.
ref time: *string*,
max. thread life time: *number*,
interval: *number*.

Explanation: The thread cleaner has started.

Action: None

MSS2243 (INF): Server main listener received message (flag=*number*):
MsgID: *string*,
InstID: *string*,
Method: *string*,
ReplyTo: *string*.

Explanation: The DirX Identity server (C++-based) received the indicated message.

Action: None.

MSS2244 (WAR): Problem: *string*.

Explanation: The DirX Identity server (C++-based) discovered the indicated internal issue.

Action: Inform technical support about the problem.

MSS2245 (ERR): InstID: *string* EXIT CmssStub::initialize(), rtc=*string* (*number*).

Explanation: The DirX Identity server (C++-based) has detected an internal error.

Action: Inform technical support about the problem.

MSS2246 (ERR): Inst ID: *string*
EXIT call, unknown exception thrown!

Explanation: An exception occurred during worker thread execution.

Action: Inform technical support about the problem.

MSS2248 (WAR): Setting #*number* not found.

Explanation: The DirX Identity server (C++-based) cannot find the setting in its configuration file.

Action: Ensure that the settings in the configuration file are correct.

MSS2249 (ERR): Exception in CmssStub::enqueue() (msgtype: *number* rtc: *number* type: *string*).

Explanation: An exception occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2250 (ERR): Exception in CmssWorker::Run() (Id: *string* rtc: *number*).

Explanation: An exception occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2251 (ERR): Failure in CmoduleList::find(): empty input name given.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2252 (ERR): Failure in CmoduleList::find(): empty name in module list (member=0x*value* name=*string* slot=*number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2253 (ERR): Wrong message received via ICom: op_code: *number* (worker: *string*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2254 (ERR): IComReceive() failure Id: *string* rtc: *number*.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2255 (ERR): IComSend() failure Id: *string* rtc: *number*.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2256 (ERR): *string* sent *string* to component *string* failed (rtc: *string*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2257 (ERR): CmssSvr::incrementWorkerCnt() failed (cnt: *number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2258 (ERR): Life saver thread failed for component *string* (rtc: *string*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2259 (ERR): Session termination failed (err: *number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2260 (INF): *string*.

Explanation: This is a DirX Identity server (C++-based) informal notice.

Action: See the error text for further information.

MSS2261 (INF): DirX Identity server (C++-based) exits without errors (PID: *number*).

Explanation: DirX Identity server (C++-based) has terminated.

Action: None.

MSS2262 (ERR): DirX Identity server (C++-based) has exited with errors (code: *string*, type: *string*, reason: *string*, PID: *number*).

Explanation: DirX Identity server (C++-based) has terminated with errors.

Action: See the error text for further information.

MSS2264 (INF): *string*.

Explanation: This is a DirX Identity server (C++-based) informal notice.

Action: See the text for further information.

MSS2265 (ERR): *string*.

Explanation: This is a DirX Identity server (C++-based) error notice.

Action: See the error text for further information.

MSS2266 (FTL): *string*.

Explanation: This is a DirX Identity server (C++-based) error notice.

Action: See the error text for further information.

MSS2267 (ERR): *string*.

Explanation: This is a DirX Identity server (C++-based) error notice.

Action: See the error text for further information.

MSS2270 (ERR): Life saver thread: received error response: component:_string_, rtc=*string* (response time=*number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2271 (ERR): CmssStub::initialize(): wait_for_sync from newly created thread failed with rtc=*number* (timeout=*number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2300 (ERR): Instance 'dxmmsssvr' is already running.

Explanation: A DirX Identity server (C++-based) instance is already running. Starting a second instance is not supported.

Action: Wait until the first instance terminates.

MSS2301 (ERR): Restarting instance 'dxmmsssvr'.

Explanation: A DirX Identity server (C++-based) instance has been restarted due to a previous failure.

Action: See the log files for further information.

MSS2302 (INF): dxmmsssvr started.

Explanation: The DirX Identity server (C++-based) has started.

Action: None.

MSS2303 (ERR): dxmmsssvr terminated with exit code *number* (*string*).

Explanation: The DirX Identity server (C++-based) has terminated with errors.

Action: See the exit code for further information.

MSS2304 (INF): dxmmsssvr terminated without errors (*string*).

Explanation: The DirX Identity server (C++-based) has terminated.

Action: None.

MSS2305 (ERR): Restart limit of *number* exceeded.

Explanation: The restart limit after DirX Identity server (C++-based) crashes has been reached.

Action: See the log files for further information.

MSS2306 (INF): dxmmsssvr received signal *number* (*string*).

Explanation: DirX Identity server (C++-based) has caught an unexpected signal.

Action: None.

MSS2307 (ERR): dxmmsssvr received SIGSTOP signal *number* (*string*).

Explanation: DirX Identity server (C++-based) has caught a STOP signal.

Action: None.

MSS2308 (INF): Sending dxmmsssvr instance (pid=*number*) signal *number*.

Explanation: The daemon process is sending a signal to stop the DirX Identity server

(C++-based).

Action: None.

MSS2309 (ERR): Signaling dxmmsssvr instance (pid=*number*) failed (errno=*number*).

Explanation: A system failure has occurred while sending a signal to stop the DirX Identity server (C++-based).

Action: See the errno code for further information.

MSS2310 (WAR): Distorted ReplyTo field in message: msg Id: *string*, InstID: *string*, msgtype: *string*, replyto: *string*.
Message discarded.

Explanation: The DirX Identity server (C++-based) has received a distorted message.

Action: Inform technical support about the problem.

MSS2311 (WAR): Distorted destination field in ACK message:
instID: *string*,
msgtype: *string*,
dest: *string*.

Explanation: The DirX Identity server (C++-based) has received a distorted message.

Action: Inform technical support about the problem.

MSS2312 (ERR): setsid() failure (errno=*number*).

Explanation: A system failure has occurred during a setsid() call.

Action: See the errno code for further information.

MSS2313 (ERR): Internal error: cannot access mssSvr instance to decrement worker count (thread=*string*, step=*number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2314 (ERR): CmssWorker::icomDetach() failure (rtc=*number*, worker=*string*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2315 (ERR): Exception in CmssCleaner::Run() (rtc=*number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2316 (ERR): CmssCleaner::Run(): No topic to publish to.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2317 (ERR): internal error: sending via ICom to cleaner thread failed (rtc=*number*).

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2318 (ERR): Cannot allocate condition variable '*string*'.

Explanation: An internal error occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2319 (ERR): Cannot initialize condition variable '*string*' (err: *number*).

Explanation: An internal error occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2320 (ERR): Cannot signal condition variable '*string*' (err: *number*).

Explanation: An internal error occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2321 (FTL): Distorted ReplyTo field: '*string*' (expected prefix: '*string*' msg type: *string* len: *number* ptr: 0x*value*).

Explanation: An internal error occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2322 (ERR): Function '*string*' not defined in '*string*'.

Explanation: An internal error occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2323 (ERR): CmssStub::replyAckMsg(): IComSend() for component *string* (msgtype *string*) failed with *number*.

Explanation: An internal error occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2324 (ERR): CmssStub::replyAckMsg(): IComSend() for component *string* (msgtype *string*) not possible.

Explanation: An internal error occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2325 (ERR): CmssSvr::initialize(): cannot start *string* thread.

Explanation: An internal error occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2326 (INF): Life saver thread started (time limit=*number*).

Explanation: The DirX Identity server (C++-based) has started the life saver component.

Action: None.

MSS2327 (INF): Life saver: component *string* must be restarted; reference time:_string_ (*number*), response time: *string* (*number*).

Explanation: The DirX Identity server (C++-based) has to restart a component again.

Action: None.

MSS2328 (INF): Life saver: component *string* must be restarted due to timeout; reference time:_string_ (*number*).

Explanation: The DirX Identity server (C++-based) has to restart a component again.

Action: None.

MSS2329 (INF): Life saver: cannot find component *string*. Will restart component.

Explanation: The DirX Identity server (C++-based) has to restart a component again.

Action: None.

MSS2330 (FTL): Service not started. Reason: Data in local configuration file invalid (item = *string*).

Explanation: The DirX Identity server (C++-based) cannot start because the ini file settings are invalid.

Action: Check the server ini file settings.

MSS2331 (ERR): dxmAttShm() failure: *string*.

Explanation: A system failure has occurred during a dxmAttShm() call.

Action: See the error message for further information.

MSS2332 (ERR): dxmPutShm() failure: *string*.

Explanation: A system failure has occurred during a dxmPutShm() call.

Action: See the error message for further information.

MSS2333 (ERR): dxmDetShm() failure: *string*.

Explanation: A system failure has occurred during a dxmDetShm() call.

Action: See the error message for further information.

MSS2334 (INF): Timeout for the DirX Identity login dialog. Server not started.

Explanation: No input received for password or PIN

Action: Start the DirX Identity server (C++-based) again

MSS2335 (ERR): thr_create() failure: return code = *number*.

Explanation: A system failure has occurred during a thr_create() call.

Action: See the returncode value for further information.

MSS2336 (ERR): Service not started. Reason: SVC initialization failed (status = *value*).

Explanation: Initialization of the serviceability failed.

Action: See the status value for further information.

MSS2337 (ERR): Service not started. Reason: Putenv for NSPR_NATIVE_THREADS_ONLY failed.

Explanation: Setting of the environment variable NSPR_NATIVE_THREADS_ONLY=1 failed

Action: Set the variable manually

MSS2338 (ERR): Initialization of crypt library failed: *number string*.

Explanation: The DirX Identity server (C++-based) cannot initialize the crypt library.

Action: See the given error code for more information.

MSS2339 (FTL): Service not started. Reason: insufficient memory.

Explanation: DirX Identity server (C++-based) is unable to allocate memory that is required. This event can occur if there are too many processes running in parallel or if the system load is too heavy for DirX Identity.

Action: Reduce the system load and restart the DirX Identity server (C++-based). If the system load does not seem to be too heavy, then inform technical support about the problem.

MSS2340 (ERR): Service not started. Reason: invalid value for 'server_restart' in the dxmmsssvr.ini file.

Explanation: The value of server_restart is invalid.

Action: Correct the value in the config file.

MSS2341 (ERR): Service not started. Reason: dxmPutShm() failure: *string*.

Explanation: A system failure has occurred during a dxmPutShm() call.

Action: See the error message for further information.

MSS2342 (ERR): Service not started. Reason: dxmAttShmEx() failure: *string*.

Explanation: A system failure has occurred during a dxmAttShmEx() call.

Action: See the error message for further information.

MSS2343 (ERR): Service not started. Reason: cannot read the configuration file *string*.

Explanation: The DirX Identity server (C++-based) cannot start because it cannot read its configuration file dxmmsssvr.ini.

Action: Check the location and the correct content of the dxmmsssvr.ini configuration file

MSS2344 (ERR): Service not started. Reason: file *string* not found.

Explanation: The specified file does not exist.

Action: Install the DirX Identity server (C++-based).

MSS2345 (ERR): Service not started. Reason: function CopyFile for file *string* failed - *string*.

Explanation: The specified file could not be copied.

Action: Maybe the DirX Identity server (C++-based) is not yet terminated. Wait a moment and start the service again.

MSS2348 (ERR): Exception in CmssSocket::Run() rtc=*number*.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2349 (ERR): CmssSocket::Run() create socket error: *number*: *string*.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2350 (ERR): CmssSocket::Run() bind socket error: *number*: *string*.

Explanation: Check other message that indicates the error of this issue.

Action: If you have serious problems with the key transfer socket then first change the port number in the configuration store and restart the service. If it still fails inform technical support.

MSS2351 (ERR): CmssSocket::Run() listen socket error: *number*: *string*.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2352 (ERR): CmssSocket::Run() accept socket error: *number*: *string*.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2353 (ERR): CmssSocket::Run() shutdown socket '*value*' error: *number*: *string*.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2354 (ERR): CmssSocket::Run() close socket '*value*' error: *number*: *string*.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2355 (WAR): CmssSocketWorker::Run() connected client '*string*' not local.

Explanation: Only local connections are allowed.

Action: Start the client on the machine where the DirX Identity server (C++-based) resides.

MSS2356 (FTL): Cannot create thread *string*.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2357 (INF): Invalid argument for PSE request.

Explanation: The DirX Identity server (C++-based) cannot process the PSE request.

Action: Inform technical support about the problem.

MSS2358 (WAR): Missing argument for PSE request.

Explanation: The DirX Identity server (C++-based) cannot process the PSE request.

Action: Inform technical support about the problem.

MSS2359 (WAR): Invalid operation request.

Explanation: The DirX Identity server (C++-based) cannot process the operation request.

Action: Inform technical support about the problem.

MSS2360 (WAR): Invalid version.

Explanation: The DirX Identity server (C++-based) cannot process the request.

Action: Inform technical support about the problem.

MSS2361 (WAR): Cannot find the requested information.

Explanation: The DirX Identity server (C++-based) cannot process the request. On Unix machines this could happen if the work path in the configuration database contains a linked directory.

Action: On Unix check the work path. Only a work path without linked directories is allowed.

MSS2362 (ERR): No port number for secure socket connections for DirX Identity agents is configured (although encryption mode is enabled).

Explanation: The DirX Identity server (C++-based) cannot start certain meta agents with encryption mode

Action: Configure a port number in the service object of the DirX Identity server (C++-based) object.

MSS2363 (ERR): CmssSocket::Run() select socket error: *number*: *string*.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2364 (ERR): CmssSocket::Run() select socket timed out after *number* seconds.

Explanation: The connected client did not respond after a certain time.

Action: Check the performance of the system. If the error still occurs inform technical support about the problem.

MSS2365 (WAR): MSSmethodMap::l2s invalid input parameter: *number*.

Explanation: An internal warning has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem if you find other error messages together with this one.

MSS2366 (INF): Enqueuing create message to the worker thread failed (internal error: *number*).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2367 (ERR): Enqueuing create message to the worker thread failed (internal error: *number*).

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to an internal error.

Action: Inform technical support about the problem.

MSS2368 (ERR): Message cannot be sent because field JMSReplyTo is missing.

Explanation: The DirX Identity server (C++-based) cannot respond to the incoming message due to an error.

Action: Inform technical support about the problem.

MSS2369 (ERR): Worker library cannot be loaded (InstId *string*).

Explanation: The DirX Identity server (C++-based) cannot load the worker library due to an internal error.

Action: Inform technical support about the problem.

MSS2370 (ERR): publish failed (error code *number*).

Explanation: The DirX Identity server (C++-based) cannot publish a response message

due to an internal error.

Action: Inform technical support about the problem.

MSS2371 (ERR): *string*.

Explanation: The check of the credentials failed.

Action: See the error text.

MSS2372 (ERR): Cannot destroy condition variable '*string*' (err: *number*).

Explanation: An internal error occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2373 (INF): Processing of this instance is delayed until instance with Id '*string*' has finished.

Explanation: A workflow or activity should be started but the same instance is still running.

Action: None.

MSS2374 (INF): Continue processing because predecessor has finished; abort: *string*.

Explanation: A delayed instance is continuing to process.

Action: None.

MSS2375 (INF): Instance has been aborted.

Explanation: The DirX Identity server (C++-based) cannot start an instance because the instance has been aborted.

Action: None.

MSS2376 (INF): Message statistics:
command msg: *number* comm int msg: *number*,
status msg: *number* filesvc msg: *number*,
workflow msg: *number* workf ok msg: *number*,
agent msg: *number* agent ok msg: *number*,
string.

Explanation: Statistics about processed messages.

Action: None.

MSS2377 (FTL): General exception in CConnectorManager::start().

Explanation: The C Connector framework could not be started.

Action: See messages of the framework for more details.

MSS2378 (FTL): General exception in CConnectorManager::abort().

Explanation: The C Connector framework could not be aborted.

Action: See messages of the framework for more details.

MSS2379 (FTL): Exception in CConnectorManager::abort(): *string*.

Explanation: The C Connector framework could not be aborted.

Action: See messages of the framework for more details.

MSS2380 (ERR): Initialization of thread in crypt library failed: *number string*.

Explanation: The DirX Identity server (C++-based) cannot initialize the crypt library.

Action: See the given error code for more information.

MSS2381 (ERR): crypt library function export_private_keys failed: *number string*.

Explanation: The DirX Identity server (C++-based) cannot initialize the crypt library.

Action: See the given error code for more information.

MSS2382 (ERR): crypt library function import_private_keys failed: *number string*.

Explanation: The DirX Identity server (C++-based) cannot initialize the crypt library.

Action: See the given error code for more information.

MSS2383 (ERR): crypt library decryption failed: *number string*.

Explanation: A value cannot be decrypted.

Action: See the given error code for more information.

MSS2384 (ERR): Encrypt/Decrypt error: *string string*.

Explanation: The DirX Identity server (C++-based) cannot encrypt/decrypt a password.

Action: See the given error code for more information.

MSS2385 (WAR): CmssSocket::Run() setsocketopt error: *number*: *string*.

Explanation: An internal warning has occurred in the DirX Identity server (C++-based).

Action: If you have serious problems with the key transfer socket inform technical support.

MSS2386 (FTL): cannot read local configuration file.

Explanation: The DirX Identity server (C++-based) cannot start because the configuration file cannot be read.

Action: Check the location and settings of the configuration file.

MSS2387 (ERR): Server could not be started and exited with code 127. Usually this means that a shared library is missing.

Explanation: Usually this means that a shared library is missing.

Action: Use the Unix command ldd for the executable dxmmsssvr to see which library is

missing. (after initializing the environment with .dirxmetarc)

MSS2388 (INF): The DirX Identity server (C++-based) is configured that the SPML Connector service is not running.

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to his configuration.

Action: Change the configuration.

MSS2389 (INF): The DirX Identity server (C++-based) cannot reread his configuration file (error = *number*)

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to unexpected results during the processing.

Action: Check other message that indicates the error of this issue.

MSS2390 (INF): The DirX Identity server (C++-based) cannot resynchronize the SPML Connector service (error = *number*)

Explanation: The DirX Identity server (C++-based) cannot process the incoming message due to unexpected results during the processing.

Action: Check other messages that indicate the error of this issue.

MSS2391 (ERR): Socket thread: Function gethostname() failed (*number*; *string*)

Explanation: The DirX Identity server (C++-based) cannot retrieve the hostname.

Action: Check other messages that indicate the error of this issue. If you have serious problems with the key transfer socket inform technical support.

MSS2392 (ERR): Socket thread: Function gethostbyname() failed (*number*; *string*)

Explanation: The DirX Identity server (C++-based) cannot retrieve the IP address of the hostname.

Action: Check other messages that indicate the error of this issue. If you have serious problems with the key transfer socket inform technical support.

MSS2393 (INF): Retrying to bind to the secure socket port (retry: *number*; wait interval: *number*; port: *number*)

Explanation: The DirX Identity server (C++-based) cannot bind to the secure socket and will retry it after a little while.

Action: Check other messages that indicate the error of this issue. If you have serious problems with the key transfer socket inform technical support.

MSS2394 (FTL): Socket thread: Shutting down of the server because secure port cannot be opened.

Explanation: The DirX Identity server (C++-based) cannot bind to the secure socket.

Action: Check other messages that indicate the error of this issue. First you can try to

change to another port number for the secure port in the configuration store. If you have still serious problems with the key transfer socket inform technical support.

MSS2395 (INF): Retrying to bind to the LDAP server (retry: *number*; wait interval: *number* sec)

Explanation: The DirX Identity service cannot bind to the LDAP server and will retry it after a little while.

Action: Check your LDAP server configuration (Is the port correct? etc.). If you have still serious problems inform technical support.

MSS2396 (INF): DirX Identity server (C++-based) exits during initialization (PID: *number*).

Explanation: DirX Identity server (C++-based) has terminated.

Action: None.

MSS2397 (WAR): The DirX Identity server (C++-based) is not released for the Windows version you are using (*string*).

Explanation: The DirX Identity server (C++-based) has detected a not supported Windows version.

Action: You can not use this product on this Windows platform for productive use.

MSS2398 (ERR): Format error in serial number *string* from keyOwnerPSE=*string* of account DN=*string*: hexadecimal format [0-9a-f] expected.

Explanation: Only hexadecimal characters 0-9a-f are allowed in a serial number of a pse

Action: Correct the keyOwnerPSE attribute of the account.

MSS2399 (ERR): Can’t extract serial number from keyOwnerPSE=*string* of account DN=*string*.

Explanation: The keyOwnerPSE multivalue attribute contains an entry with wrong syntax. The serial number cannot be extracted or is not an integer.

Action: Correct the entry using the tool dirxgenpse.

MSS2400 (ERR): Read of account DN=*string* failed.

Explanation: The account object cannot be read. This is unlikely since your bind was successful.

Action: Check whether your LDAP server is running.

MSS2401 (ERR): Encryption of bind password failed: *string*

Explanation: The bind password cannot be encrypted.

Action: See the given error code for more information.

MSS2402 (ERR): Cannot find any keyOwnerPSE attribute. Did you generate any key pair?

Explanation: Cannot find any keyOwnerPSE attribute.

Action: Did you generate any key pair via dirxgenpse?

MSS2403 (WAR): Terminating server immediately after interval of *number* seconds.

Explanation: The server is forcely stopped. Terminating gracefully failed.

Action: In rare cases the server can not shutdown gracefully. If this happens more often inform technical support about the problem.

MSS2404 (ERR): Internal error while Writing diagnostic information file.

Explanation: This message indicates that writing a diagnostic information file during startup failed for unknown reasons. This should not happen under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

MSS2405 (ERR): Writing diagnostic info file *string* failed.

Explanation: This message indicates that writing a diagnostic information file during startup failed.

Action: If the problem cause is obvious from the file name indicated in the message or from the log file of the C++-based server, then fix the problem, otherwise inform DirX Identity technical support about the problem.

MSS2407 (INF): Terminated subscriber (*string*).

Explanation: A message subscriber has been terminated.

Action: None.

MSS2408 (INF): Activating server...

Explanation: Command to activate server has been called.

Action: None.

MSS2409 (INF): Server activated: *number*

Explanation: Command to activate server has finished.

Action: None.

MSS2410 (INF): Deactivating server...

Explanation: Command to activate server has been called.

Action: None.

MSS2411 (INF): server deactivated: *number*

Explanation: Command to deactivate server has finished.

Action: See returncode.

MSS2412 (ERR): Activating messaging failed: *number*

Explanation: Command to activate server failed.

Action: None.

MSS2413 (ERR): Deactivating messaging failed: *number*

Explanation: Command to deactivate server failed.

Action: None.

MSS2414 (INF): Server is going to restart to reload configuration.

Explanation: Command to restart server in progress.

Action: None.

MSS2415 (INF): DirX Identity server (C++-based) exits to be restarted (PID: *number*).

Explanation: DirX Identity server (C++-based) has terminated. Service program will restart server.

Action: None.

MSS2416 (FTL): Thread start failed. Could not accept new create workflow requests. Server terminates with exit code 12 and will normally be restarted.

Explanation: The DirX Identity server (C++-based) cannot start new threads for incomming workflow requests. Server terminates and will be restarted.

Action: Analyze the previously detected errors. Check the number of active threads. Increase the limit for thread number if necessary. If the problem persists, inform DirX Identity technical support about the problem.

MSS2417 (ERR): CmssSocketWorker::readSocket() recv() error: *number*: *string*.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2418 (INF): connector service: *number* (1 is enabled) with ports *number* and *number* (SSL).

Explanation: Info about SPML connector service.

Action: None.

MSS2419 (ERR): Maximum thread count exceeded (maximum size=*number*).

Explanation: The DirX Identity server (C++-based) has reached the limit of internal running threads.

Action: Change the setting that controls the number of allowed worker threads (workflow engines and agents controllers).

MSS2420 (INF): Insufficient memory.

Explanation: The DirX Identity server (C++-based) is unable to allocate memory that is required to process the workflow. This event can occur if there are too many processes running in parallel or if the system load is too heavy for DirX Identity.

Action: Reduce the system load and restart the DirX Identity server (C++-based). If the system load does not seem to be too heavy, then inform technical support about the problem.

MSS2421 (ERR): Cast for msg (inst Id=*string*) failed.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

MSS2422 (FTL): WSAStartup() failed.

Explanation: The DirX Identity server (C++-based) cannot start due to an error during Windows socket initialization.

Action: Inform technical support about the problem.

MSS2423 (INF): Attempt to start workflow '*string*' a second time.

Explanation: The DirX Identity server (C++-based) cannot start a workflow instance because an instance of the same workflow is already running.

Action: Check whether the schedules are too tight or why the first workflow is still running.

MSS2424 (ERR): Crypt library encryption failed: *number string*.

Explanation: A value cannot be encrypted.

Action: See the given error code for more information.

16.1.14. Logging Messages (SDR)

This topic describes the logging messages returned by the DirX Identity scheduler.

SDR6500 (FTL): Exception occurred in *string*!

Explanation: An internal error has occurred in the scheduler.

Action: Inform DirX Identity technical support about the problem.

SDR6551 (ERR): Missed scheduled start.
Schedule name : *string*
Workflow DN : *string*
Workflow name : *string*
Missed start time : *string*
Current time : *string*

Explanation: This message is returned in event log or in ASCII log files if the scheduler has missed starting a workflow at a scheduled time. The scheduler will only start one workflow per schedule simultaneously because multiple simultaneous starts of the

same workflow are not supported. So if a workflow's execution is longer than deviation + schedule interval, then the scheduler will miss start times. This error also occurs if the polling time is greater than deviation + scheduler interval or if the workflow timed out.

Action: If the related schedule does not conform to section 'Rules for Schedules' of 'Workflow Design Rules' in the DirX Identity Connectivity Administration Guide, then customize schedule parameters according to the rules described there. If the cause of the problem is a workflow timeout, then analyze the problem as described for error WFE5015.

SDR6552 (ERR): Missed Scheduled Starts.
Schedule name : *string*
Workflow DN : *string*
Workflow name : *string*
Missed start times : *string* till *string*
Current time : *string*

Explanation: This message is returned in event log or ASCII log files if the scheduler has missed starting a workflow at multiple scheduled times in sequence (see SDR6551 for more information).

Action: See SDR6551.

SDR6553 (ERR): Publish createInstance ERROR *number* (*string*).
Workflow name: *string*
Workflow DN: *string*

Explanation: Before it can start workflows, the scheduler must create a workflow engine instance. This message is returned in event log or ASCII log files if the scheduler fails to create a workflow engine instance in the DirX Identity server.

Action: If the cause of this communication problem is neither heavy system load nor temporary unavailability of the message server, inform DirX Identity technical support about the problem.

SDR6554 (ERR): Publish execute ERROR *number* (*string*).
Workflow name: *string*
Workflow DN: *string*
Instance ID: *string*

Explanation: This message is returned in event log or ASCII log files if the scheduler fails to publish a request to execute a workflow to the DirX Identity server.

Action: If the cause of this communication problem is neither heavy system load nor temporary unavailability of the message server, inform DirX Identity technical support about the problem.

SDR6555 (ERR): Publish abort ERROR *number* (*string*).
Workflow name: *string*
Workflow DN: *string*
Instance ID: *string*

Explanation: This message is returned in event log or ASCII log files if the scheduler fails

to publish a request to abort a workflow to the DirX Identity server.

Action: If the cause of this communication problem is neither heavy system load nor temporary unavailability of the message server, inform DirX Identity technical support about the problem.

SDR6556 (ERR): *string* message acknowledged with return code *number* (*string*).
Message ID: *string*
Instance ID: *string*
Workflow name: *string*
Workflow DN: *string*

Explanation: This message is returned in event log or ASCII log files if the scheduler receives an error response for createInstance (workflow) or execute (workflow). The message contains information about the return code, the type of the acknowledged request (createInstance or execute), the message ID and information about the related workflow engine instance.

Action: Check the message to see if the cause of the problem is obvious and can be solved by administrative means, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

SDR6557 (ERR): Too late to start workflow. Latest possible start time exceeded by *number* seconds.
Schedule name : *string*
Workflow DN : *string*
Current time : *string*
Latest possible start time: *string*

Explanation: This message is returned in event log or ASCII log files if the scheduler detects that it is too late to start a workflow at a scheduled time. This situation can occur during very heavy system load (which may cause too much time to pass between scheduling the workflow and launching the workflow) or if the related schedule does not satisfy tuning recommendations.

Action: See SDR6651.

SDR6558 (ERR): Workflow timeout exceeded by *number* seconds - detected at *string*.
Schedule name : *string*
Workflow DN : *string*
Workflow name : *string*
Scheduled start time : *string*
Actual start time : *string*

Explanation: This message is returned in event log or ASCII log files if the scheduler detects that it must abort a workflow because it has exceeded its timeout value.

Action: See WFE5015

SDR6560 (ERR): Invalid value *number* for attribute *string*. Schedule will be discarded.
Schedule DN : *string*
Schedule name : *string*

Explanation: This message is returned when an integer attribute value for a schedule is invalid. For instance, the interval for a schedule must be greater than 0. The scheduler ignores the related schedule. The message displays the attribute name according to DirX Identity Manager's expert view and the related attribute value. A value of 0 means that the attribute may be missing from the configuration database.

Action: Supply a valid value for the related attribute to correct the schedule.

SDR6561 (ERR): Invalid value *number* for attribute *string* in central configuration object. Defaulted to *number* seconds.

Explanation: This message is returned when an integer value for a scheduler-relevant attribute in the central configuration object is invalid. For instance, the sync schedule interval must be greater than 0. The scheduler ignores the related attribute and assumes the suitable default value instead. The message displays the attribute name according to DirX Identity Manager's expert view and the related attribute value. A value of 0 means that the attribute may be missing from the configuration database.

Action: If the scheduler should run with a non-default value for the related attribute, then supply a valid attribute value to the central configuration object.

SDR6562 (INF): Schedule has expired - detected at *string*.
Schedule Name : *string*
Workflow DN : *string*
Expiration time: *string*

Explanation: This message is returned in event log or ASCII log files if a schedule expires.

Action: None.

SDR6563 (ERR): Sending cleanup request to status tracker failed with error *number* (*string*).

Explanation: This message is returned when the scheduler has failed to send a request to the status tracker to clean up the status area. The scheduler does not wait for a full cleanup interval in order to resend the request. Instead, it attempts to send the request again for as long as cleanup deviation permits. In this case, the minimum time interval between two attempts is at least the polling time and should not be much more under normal circumstances.

Action: If the retry mechanism has been successful; that is, if your status area has been cleaned up as scheduled, no further action is required. If the cause of this communication problem is neither heavy system load nor temporary unavailability of the message server, inform DirX Identity technical support about the problem. If the system load does not seem too heavy to you, then inform DirX Identity technical support as well.

SDR6564 (ERR): Synchronizing schedules failed with error *number* (*string*).
Suspending starting workflows until next synchronizing schedules is successful.

Explanation: This message is returned when the scheduler fails to read its schedules from the configuration database and thus cannot synchronize the schedules. As a result, the scheduler suspends starts of new workflows until it has reread its schedules successfully. Workflows that the scheduler has already started and that are still running

are not affected. The scheduler will not wait for a full time interval in order to re-attempt to send the request. Instead, it will re-attempt the operation more frequently. In this case, the minimum time interval between two attempts is at least the polling time and should not be much more under normal circumstances.

Action: If the cause of the problem is not because the configuration database is unavailable, analyze error messages in event log or ASCII log files to check whether they indicate inconsistencies of the configuration database with respect to workflow definitions. If so, then fix the inconsistencies. Otherwise, inform DirX Identity technical support about the problem.

SDR6565 (ERR): Received message (message ID = *string*) of unexpected type.
Instance ID: *string*,
Workflow DN: *string*

Explanation: This message is returned when the DirX Identity server folder forwards a message that does not represent a response that the scheduler expects.

Action: Inform DirX Identity technical support about the problem.

SDR6566 (ERR): Unexpected message (message ID = *string*) received - ignored.

Explanation: This message says that the DirX Identity server folder forwarded a message that does not represent a response being expected by the scheduler. The message does not even match with a workflow currently driven by the scheduler.

Action: Inform DirX Identity technical support about the problem.

SDR6567 (INF): Scheduler is running.

Explanation: This message indicates that the scheduler is running.

Action: None.

SDR6568 (INF): Scheduler is going down.

Explanation: This message indicates that the scheduler is terminating.

Action: None, if the message is returned because the administrator or system shutdown is terminating the DirX Identity server . Otherwise, inform DirX Identity technical support about the problem.

SDR6569 (INF): Scheduler stopped.

Explanation: This message indicates that the scheduler stopped.

Action: None, if the message is returned because the administrator or system shutdown is terminating the DirX Identity server. Otherwise, inform DirX Identity technical support about the problem.

SDR6570 (INF): Scheduling is *string*.

Explanation: This message will appear in event log or ASCII log files and indicates whether scheduling is enabled or disabled.

Action: None.

SDR6571 (INF): Schedule definition does not allow workflow execution before *string*.
Schedule Name : *string*
Workflow DN : *string*
Workflow Name : *string*

Explanation: This message indicates that the schedule definition does not allow workflow execution in a time horizon of 1 week. The name of the schedule and related workflow are included in the message. The time value (format YYYYMMDDhhmmssZ) specifies a lower bound for next schedule execution.

Action: If the related workflow should be executed more frequently, then adjust the start time with respect to schedule range and/or decrease the schedule interval.

SDR6572 (ERR): Attribute *string* is not correct. Schedule will be discarded.
Schedule DN : *string*
Schedule name : *string*

Explanation: This message is returned when an attribute for a schedule is invalid. The scheduler ignores the related schedule. The message displays the attribute name according to DirX Identity Manager's expert view.

Action: Correct the schedule.

SDR6601 (WAR): Object *value*, Function *string*: setting return value *number* (*string*) for acknowledge.

Explanation: An internal scheduler function stores said return value into a response message.

Action: None, if the return value indicates success. Otherwise, analyze error messages in event log or ASCII log files or in the status area. Check the messages to see if the cause of the problem is obvious and can be solved by administrative means, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

SDR6602 (WAR): Constructor for *string*: setting return value *number* (*string*) for acknowledge.

Explanation: A constructor for an internal scheduler class stores said return value into a response message.

Action: None, if the return value indicates success. Otherwise, analyze error messages in event log or ASCII log files or in the status area. Check the messages to see if the cause of the problem is obvious and can be solved by administrative means, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

SDR6603 (WAR): EXIT Object *value*, Function *string*, return value is *number* (*string*).

Explanation: An internal scheduler function has returned said return value.

Action: None, if the return value indicates success. Otherwise, analyze error messages in event log or ASCII log files or in the status area. Check the messages to see if the cause of the problem is obvious and can be solved by administrative means, then fix the

problem. Otherwise, inform DirX Identity technical support about the problem.

SDR6604 (WAR): EXIT Object *value*, Function *string*, return value is *number* (*string*).

Explanation: An internal scheduler function returns said return value.

Action: None, if the return value indicates success. Otherwise, analyze error messages in event log or ASCII log files or in the status area. Check the messages to see if the cause of the problem is obvious and can be solved by administrative means, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

SDR6605 (WAR): Internal Problem, error = *number*.

Explanation: An internal scheduler problem has occurred.

Action: Inform the DirX Identity technical support about the problem.

SDR6606 (INF): START Scheduler.
Scheduler instance ID: *string*

Explanation: The scheduler has been started.

Action: None.

SDR6607 (INF): END Scheduler.
Scheduler instance ID: *string*
Return code: *number*

Explanation: The scheduler has been terminated.

Action: None.

SDR6608 (INF): Cleanup request successfully sent to status tracker.

Explanation: The scheduler has triggered the status tracker to cleanup obsolete status entries.

Action: None.

SDR6609 (INF): New start time of status tracker cleanup computed.
Scheduled next at : *string*
Current Time : *string*

Explanation: The scheduler traces when next to trigger the status tracker to cleanup obsolete status entries. The computation of the related start time is based on the related internal schedule.

Action: None.

SDR6610 (INF): Synchronize schedules successful.

Explanation: The scheduler has re-read its schedules, either because having been triggered by the DirX Identity Manager, or automatically due to the related internal schedule.

Action: None.

SDR6611 (INF): New start time for synchronizing schedules computed.
Scheduled next at : *string*
Current Time : *string*

Explanation: The scheduler traces when next to re-read its schedules, based on the related internal schedule.

Action: None.

SDR6612 (INF): *string*

Explanation: This message is designed to log generic Data as notice.

Action: None.

SDR6613 (INF): Workflow timeout for schedule *string* has changed.
Workflow DN : *string*
Workflow name : *string*
Previous timeout : *number*
New timeout : *number*

Explanation: The scheduler has detected that said workflow timeout has been changed. It processes the workflow using the new timeout value.

Action: None.

SDR6651 (INF): Missed Start: *string*.

Explanation: See SDR6551. The string represents a missed start time.

Action: See SDR6551.

SDR6652 (INF): Missed Starts: *string* till *string*.

Explanation: See SDR6552. The strings represent an interval of missed start times.

Action: See SDR6552.

SDR6653 (INF): Publish createInstance ERROR *number* (*string*).

Explanation: See SDR6553. This message will appear in a status entry that has been created completely by the scheduler, because there is no workflow engine to perform this task in this error situation.

Action: See SDR6553.

SDR6654 (INF): Publish execute ERROR *number* (*string*).

Explanation: See SDR6554. This message will appear in a status entry that has been created completely by the scheduler, because there is no workflow engine to perform this task in this error situation.

Action: See SDR6554.

SDR6655 (INF): Publish abort ERROR *number* (*string*).

Explanation: See SDR6555.

Action: See SDR6555.

SDR6656 (INF): *string*-message acknowledged with return code *number* (*string*).

Explanation: See also SDR6556. The first *string* indicates the message type. If the message type has been createInstance, then the related status entry has been created completely by the scheduler, because there is no workflow engine to perform this task for this error situation.

Action: See SDR6556.

SDR6657 (INF): Too late to start workflow - latest possible start time was *string*.

Explanation: See also SDR6557. The related status entry has been created completely by the scheduler, because there is no workflow engine to perform this task for this error situation.

Action: See SDR6557.

SDR6658 (INF): Workflow time out exceeded by *number* seconds - detected at *string*.

Explanation: See SDR6558.

Action: See SDR6558.

SDR6659 (INF): Workflow will be aborted.

Explanation: As a consequence of a workflow timeout, the scheduler will abort the workflow.

Action: See SDR6558.

16.1.15. Logging Messages (SHM)

This topic describes the logging messages returned by the DirX Identity shared memory mapped file component.

SHM9501 (ERR): Insufficient memory

Explanation: A component is unable to allocate memory that is required to process the workflow. This event can occur if there are too many processes running in parallel or if the system load is too heavy for DirX Identity.

Action: Reduce the system load and restart the DirX Identity server (C++-based). If the system load does not seem to be too heavy, then inform technical support about the problem.

SHM9502 (ERR): Function not implemented yet.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9503 (ERR): Function called with an invalid parameter.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9504 (ERR): The shm log file cannot be reopened.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9505 (ERR): Opening the internal named file mapping object failed.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9506 (ERR): Creating the internal named file mapping object failed.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9507 (ERR): Getting a pointer to the file-mapped shared memory failed.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9508 (ERR): Opening or creating semaphore 'semafull' failed.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9509 (ERR): Opening or creating semaphore 'semaempty' failed.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9510 (ERR): Opening or creating semaphore 'semawriter' failed.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9511 (ERR): Destination PID in shared memory is invalid.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9512 (ERR): Invalid size given in shared memory.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9513 (ERR): Unexpected error on semaphore in function WaitForSingleObject().

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9514 (ERR): Waiting failed on semaphore in function WaitForSingleObject().

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9516 (ERR): Error on trying to access the shared memory exclusively.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9517 (ERR): Error on releasing a semaphore.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9518 (ERR): Invalid memory size given in shared memory.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9519 (ERR): Waiting failed on read event in function WaitForSingleObject().

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9520 (ERR): Unexpected error on read event in function WaitForSingleObject().

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9521 (ERR): Creating event failed.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9526 (ERR): Closing event handle failed.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9550 (WAR): Timeout occurred while waiting for an semaphore.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9552 (WAR): Semaphore is locked.

Explanation: An internal error has occurred in the DirX Identity server (C++-based).

Action: Inform technical support about the problem.

SHM9558 (WAR): A function to tidy up failed: *string* (in *string*;_string_)

Explanation: A warning has occurred during shutdown in the DirX Identity server (C++-based). Probably a file that should be deleted could not be found.

Action: Only if the server does not start-up anymore inform technical support about the problem.

16.1.16. Logging Messages (STT)

This topic describes the logging messages returned by the DirX Identity status tracker.

STT6000 (FTL): Exception occurred in *string*!

Explanation: This message indicates an internal status tracker client problem.

Action: Inform DirX Identity technical support about the problem.

STT6001 (ERR): Cannot create message.

Explanation: This message indicates an internal status tracker client problem.

Action: Inform DirX Identity technical support about the problem.

STT6002 (ERR): Error in function *string* of ConfDB.

Explanation: This message indicates an internal status tracker client problem.

Action: Inform DirX Identity technical support about the problem.

STT6003 (ERR): Parameter error.

Explanation: This message indicates an internal status tracker client problem.

Action: Inform DirX Identity technical support about the problem.

STT6004 (ERR): ActiveMQ session not available; cannot send message.

Explanation: This message indicates an internal status tracker client problem.

Action: Inform DirX Identity technical support about the problem.

STT6100 (FTL): Exception occurred in *string* (Num: *number*)!

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6101 (ERR): Error in function *string* of ConfDB (Return code: *number*).

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6102 (ERR): Error in function *string* of FileService.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6103 (ERR): No operation code in message.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6104 (ERR): Invalid operation code in message.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6105 (ERR): No remark in message.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6106 (ERR): No workflow name in message.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6107 (ERR): No workflow start time in message (OP-Code: *number*).

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6108 (ERR): No workflow end time in message.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6109 (ERR): No workflow status in message.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6110 (ERR): No activity name in message.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6111 (ERR): No activity start time in message (OP-Code: *number*).

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6112 (ERR): No activity end time in message.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6113 (ERR): No activity status in message.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6114 (ERR): No server name in message.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6115 (ERR): No status path in message.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6121 (ERR): No process data files in message.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6122 (INF): The StatusTracker was terminated (abort).

Explanation: This message indicates that the status tracker has terminated.

Action: None

STT6123 (FTL): Insufficient memory. Function: *string*.

Explanation: This message is returned if the agent controller is unable to allocate the memory that is required to process the related workflow. This error can occur if there are too many processes running in parallel or as a consequence of too heavy a system load for DirX Identity.

Action: Reduce the system load and restart the DirX Identity server. If the system load does not seem to be too heavy, inform DirX Identity technical support about the problem.

STT6124 (ERR): Cannot load the file: *string*.

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6125 (ERR): No original workflow DN in message (OP-Code: *number*).

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

STT6126 (INF): Status Tracker is running.

Explanation: This message indicates that the status tracker is running.

Action: None

STT6127 (INF): Status Tracker stopped.

Explanation: This message indicates that the status tracker has stopped.

Action: None

STT6128 (ERR): No workflow instance ID in message (OP-Code: *number*).

Explanation: This message indicates an internal status tracker problem.

Action: Inform DirX Identity technical support about the problem.

16.1.17. Logging Messages (UTL)

This topic describes the logging messages returned by the DirX Identity utilities.

UTL7001 (FTL): Function dlopen(*string*) failed.

Explanation: The DirX Identity server for C cannot open the indicated shared library.

Action: See if the library exists in the bin directory of the installation directory.

UTL7002 (WAR): Function dlfree(*string*) = *value*.

Explanation: The DirX Identity server for C cannot free the indicated shared library.

Action: None.

UTL7003 (ERR): Cannot initialize mutex '*string*' (err: *number*).

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7004 (ERR): Cannot destroy mutex '*string*' (err: *number*).

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7005 (ERR): Cannot lock mutex (err: *number*).

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7006 (ERR): Cannot unlock mutex (err: *number*).

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7051 (ERR): Thread *string* not started: *number*.

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7052 (ERR): Cannot allocate condition variable '*string*'.

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7053 (ERR): Cannot initialize condition variable '*string*' (err: *number*).

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7054 (ERR): Unexpected exception through thread function (address '*value*').

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7055 (ERR): Cannot shut down thread (err: *number*).

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7056 (ERR): Insufficient memory.

Explanation: DirX Identity server for C is unable to allocate memory that is required to process the workflow. This event can occur if there are too many processes running in parallel or if the system load is too heavy for DirX Identity.

Action: Reduce the system load and restart the DirX Identity server for C. If the system load does not seem to be too heavy, inform technical support about the problem.

UTL7057 (WAR): Problem: *string*

Explanation: The DirX Identity server for C detected the indicated issue.

Action: See the text for further actions.

UTL7058 (ERR): Unexpected exception occurred.

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7059 (ERR): Exception in CopyString(): str=*string*

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7060 (WAR): Suspicious alloc: bytes requested=0x*value*

Explanation: A high request for allocation memory occurred.

Action: Check other messages. If you don't recognize other issues then this might just happen in rare situations.

UTL7061 (ERR): Exception in DXMAlloc(): bytes requested=*number*

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7062 (ERR): Exception in DXMFree(): *value* (src: *string*, line *number*).

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7063 (FTL): Assertion *string* failed (*string*, *number*)!

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7064 (FTL): Calling DebugBreak() after assertion failed

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7065 (FTL): Open audit file '*string*' failed: errno = *number*

Explanation: The DirX Identity server for C cannot open the audit file.

Action: Check the file system and given errno number.

UTL7070 (WAR): Unknown profile entry type *number* (*string*).

Explanation: An unknown attribute was found in the file.

Action: Check the settings of the file.

UTL7080 (ERR): Unexpected exception occurred: *string*

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7082 (FTL): Internal fatal error occurred (terminating process)

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7083 (ERR): WriteFile() of string *string* failed: *string*

Explanation: An internal or system error occurred in the DirX Identity server for C.

Action: First check your file system otherwise inform technical support about the problem.

UTL7084 (ERR): CloseFile() of *string* failed: *string*

Explanation: An internal or system error occurred in the DirX Identity server for C.

Action: First check your file system otherwise inform technical support about the problem.

UTL7085 (ERR): ReadFile() in *string* failed: *string*

Explanation: An internal or system error occurred in the DirX Identity server for C.

Action: First check your file system otherwise inform technical support about the problem.

UTL7086 (ERR): ReadFile() buffer too small for file: *string*
bufferlength: *number*
buffer: *string*

Explanation: An internal or system error occurred in the DirX Identity server for C.

Action: First check your file system otherwise inform technical support about the problem.

UTL7087 (WAR): CreateFile *string* retry *number*

Explanation: To open the named file the given number of retries have been processed.

Action: This should happen only in rare situations. If this happens constantly inform technical support about the problem.

UTL7088 (ERR): CreateFile() of *string* failed: *string* (retry: *number*)

Explanation: An internal or system error occurred in the DirX Identity server for C.

Action: First check your file system otherwise inform technical support about the problem.

UTL7089 (ERR): *string*

Explanation: This is a DirX Identity server for C error notice.

Action: See the error text for further information.

UTL7090 (ERR): Cannot process file path as it is too long (>259 characters on Windows): in directory *string* the filename *string*

Explanation: A file / directory cannot be processed.

Action: Check your file system.

UTL7091 (ERR): *string* fct=*string* rtc=*string*

Explanation: An internal error occurred in the DirX Identity server for C.

Action: Inform technical support about the problem.

UTL7092 (WAR): Executing command failed: *string*.

Explanation: This message indicates that starting a command failed.

Action: None.

UTL7093 (ERR): opendir() of *string* failed: *string* (errno: *number*)

Explanation: A directory cannot be opened.

Action: Check your file system.

16.1.18. Logging Messages (WFE)

This topic describes the logging messages returned by the DirX Identity workflow engine.

WFE5000 (ERR): Internal workflow engine problem.

Explanation: This message indicates an internal workflow engine problem. This error should not occur under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

WFE5001 (ERR): Requested operation not implemented.

Explanation: This message is returned if the workflow engine processes a request for an operation that has not been implemented. This error should not occur under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

WFE5002 (ERR): Parameter error.

Explanation: This message is returned if the workflow engine attempts to process a request that contains an invalid parameter. This error should not occur under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

WFE5003 (ERR): Requested operation not supported.

Explanation: This message is returned if the workflow engine attempts to process a request for an operation that is not supported. This error should not occur under normal circumstances. This error message has been included for design reasons and is not relevant to the user.

Action: Inform DirX Identity technical support about the problem.

WFE5004 (ERR): Internal workflow engine problem (insufficient memory).

Explanation: This message is returned if the workflow engine is unable to allocate the amount of memory that is required to process the related workflow. This error can occur if there are too many processes running in parallel or as a consequence of too heavy system load for the DirX Identity system.

Action: Reduce the system load and restart the DirX Identity server. If the system load does not seem to be too heavy, inform DirX Identity technical support about the problem.

WFE5005 (ERR): Workflow execution terminated (at least one activity failed).

Explanation: This message is returned when one or more activities in the workflow have failed. When one activity fails, the workflow engine concludes that the entire workflow has failed.

Action: Analyze the related status entry for more detailed error messages. If the cause of the problem is obvious from these error messages, fix the problem. If detailed error messages are missing or unclear, examine the related event log for error information. If the cause of the problem is still unclear, inform DirX Identity technical support about the problem.

WFE5006 (ERR): Too busy to execute the request.

Explanation: This message is returned when the workflow engine instance has received more requests than it can process, e.g., to execute one workflow twice at the same time. This error should not occur under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

WFE5007 (ERR): Requested control operation (abort) not allowed in this state.

Explanation: This message exists for future enhancements. This error should not occur under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

WFE5009 (ERR): Cannot read workflow definition from configuration database.

Explanation: In order to process a workflow, the workflow engine must be able to read the related workflow definition from the configuration database, and this operation has failed. Possible reasons are missing connectivity of the configuration database and inconsistency of the workflow definition (e.g. missing end activity, missing start activity, no predecessor defined) due to manual development of workflow or an inconsistent configuration database.

Action: Check connectivity to the configuration database that the workflow engine is using and solve the problem if possible. Next, try to run the workflow again. If the same problem recurs (e.g., because incorrect credentials are being used internally), restart the server and then rerun the workflow.

WFE5010 (ERR): Cannot continue due to previous errors (createInstance failure).

Explanation: This message is returned when the workflow engine cannot continue because the related instance is invalid. This error should not occur under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

WFE5011 (ERR): Cannot connect to Status Tracker.

Explanation: During workflow execution, the workflow engine publishes status messages about the progress of the workflow. In order to perform this task, it must be able to connect to the status tracker, and it cannot perform this operation.

Action: Analyze event log/ASCII log files to determine the reason why the status tracker is not running. The problem may disappear after restarting the DirX Identity server. Inform DirX Identity technical support about the problem.

WFE5012 (ERR): Cannot continue (check activity results for details).

Explanation: This message is displayed if the workflow engine has not completed and does not expect any responses that could make it determine that the starting conditions may become true.

Action: Use the DirX Identity Manager expert view to check the consistency of your workflow definition. Perhaps an activity is marked as an end activity but has a successor. In this case, the workflow execution ends after the run of the first end activity.

WFE5013 (ERR): Inconsistent workflow definition (unknown predecessor for an activity).

Explanation: The workflow definition stored in the configuration database is inconsistent: an activity exists in the workflow that is not the start activity and that does not have a predecessor activity.

Action: Use the DirX Identity Manager expert view (and if necessary, the data view) to analyze the workflow definition. Correct the related data and rerun the workflow.

WFE5014 (ERR): Internal workflow engine problem (threads problem).

Explanation: Internal workflow engine problem that can occur because of a heavy system load. Since multiple tasks run in parallel in the DirX Identity server, there are multiple threads of execution.

Action: Try to reduce the system load and then restart the server. Inform DirX Identity technical support about the problem if the system load does not seem to be the problem or if reducing the system load seems to be too restrictive for product use.

WFE5015 (ERR): Workflow execution timed out.

Explanation: This message indicates that the workflow engine has stopped the workflow execution because its timeout value has been exceeded. See the (Central) Configuration help topics for information about the computation of a workflow timeout value.

Action: Analyze the status area to determine which activity is causing the delay. If helpful, increase the timeout values of the related jobs and/or the latency factor in the central configuration object. Run the workflow again.

WFE5016 (INF): Workflow execution aborted.

Explanation: This message indicates that the workflow engine has stopped workflow execution because it was requested to do so either by the DirX Identity Manager or the scheduler.

Action: If the scheduler or the DirX Identity Manager have not aborted the workflow, then a timeout condition is likely to be the reason for the error. Check the event log or the log files for timeout-related entries.

WFE5017 (INF): Activity aborted.

Explanation: This message indicates that the workflow engine has aborted the activity because it has aborted the workflow due to failure or timeout.

Action: If the scheduler or the DirX Identity Manager have not aborted the workflow, then a timeout condition is likely to be the reason for the error. Check the event log or the log files for timeout-related entries.

WFE5018 (ERR): Activity execution failed.

Explanation: This status entry message appears in an activity status entry to indicate that the execution of an activity has failed.

Action: Examine the entries in the status area for details.

WFE5019 (ERR): Activity execution (Creating Agent Instance) failed.

Explanation: This status entry message appears in an activity status entry to indicate that a preparation step (createInstance) for the execution of an activity has failed.

Action: If the cause of the problem is not obvious from the related status entry, check the event log or the log files to find the cause.

WFE5020 (INF): Activity completed with warning.

Explanation: This status entry message appears in an activity status entry to indicate that an activity has completed with a warning.

Action: Examine the entries in the status area for details.

WFE5021 (ERR): Unknown error *number* (*string*) received.

Explanation: This message indicates that the workflow engine has encountered an unknown problem during workflow execution, *number* indicates the unknown problem, *string* the related symbolic value.

Action: Analyze the event log or the ASCII log file in order to find and fix the problem. Inform DirX Identity technical support about the problem.

WFE5022 (INF): Partial workflow execution. Start activity: *string*.

Explanation: This status entry message appears in a workflow status entry to indicate partial execution of a workflow.

Action: None.

WFE5400 (FTL): Exception occurred in *string*!

Explanation: An internal workflow engine problem occurred. This should not happen under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

WFE5401 (WAR): Object *value*, Function *string*: setting return value *number* (*string*) for acknowledge.

Explanation: An internal workflow engine function is going to return said return value.

Action: None, if the return value indicates success. Otherwise, check the message to see if the cause of the problem is obvious and can be solved by administrative means, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

WFE5402 (WAR): Constructor for *string*: setting return value *number* (*string*) for acknowledge.

Explanation: A workflow engine instance function is going to return said return value.

Action: If the return value does not indicate success, the related workflow will not be executed. In this case, analyze related error messages event, ASCII log files or in the status area to identify the problem cause.

WFE5403 (WAR): EXIT Object *value*, Function *string*, return value is *number* (*string*).

Explanation: An internal workflow engine function has returned said return value.

Action: None, if the return value indicates success. Otherwise, analyze related error messages event log, ASCII log files or in the status area to identify the problem cause.

WFE5404 (WAR): Internal Problem, error = *number*.

Explanation: An internal non-fatal workflow engine problem occurred. This should not happen under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

WFE5405 (INF): START Workflow.
Workflow instance ID: *string*
Workflow DN: *string*
Workflow Name: *string*
Workflow Start Time: *string*

Explanation: A workflow has been started.

Action: None.

WFE5406 (INF): END Workflow.
Workflow instance ID: *string*
Workflow DN: *string*
Workflow Name: *string*
Workflow Start Time: *string*
Workflow State: *string*

Return code: *number*

Explanation: A workflow has been terminated.

Action: If the return code is other than 0, then view the status area to identify the problem cause.

WFE5407 (ERR): Publish createInstance ERROR *number* (*string*).
Workflow DN: *string*
Workflow Name: *string*
Activity Name: *string*
Run Object Name: *string*
Related MSS: *string*

Explanation: Before it can start activities, the scheduler must create an activity instance. This message is returned in event log or ASCII log files if the workflow engine fails to create an activity instance in the DirX Identity server.

Action: If the cause of this communication problem is neither heavy system load nor temporary unavailability of the message server, inform DirX Identity technical support about the problem.

WFE5408 (ERR): Publish execute ERROR *number* (*string*).
Workflow DN: *string*
Workflow Name: *string*
Activity Name: *string*
Run Object Name: *string*
Related MSS: *string*
Instance ID: *string*

Explanation: This message is returned in event log or ASCII log files if the workflow engine fails to publish a request to execute an activity to the DirX Identity server.

Action: If the cause of this communication problem is neither heavy system load nor temporary unavailability of the message server, inform DirX Identity technical support about the problem.

WFE5409 (ERR): Publish destroyInstance ERROR *number* (*string*).
Workflow DN: *string*
Workflow Name: *string*
Activity Name: *string*
Run Object Name: *string*
Related MSS: *string*
Instance ID: *string*

Explanation: After execution of an activity, the workflow engine has to destroy the related activity instance. So the agent controller can perform cleanup actions what includes also storage of delta information. This message is returned in event log or ASCII log files if the workflow engine fails to publish a request to destroy an activity instance to the DirX Identity server.

Action: If the cause of this communication problem is neither heavy system load nor temporary unavailability of the message server, inform DirX Identity technical support

about the problem.

WFE5410 (ERR): Publish abort ERROR *number* (*string*).
Workflow DN: *string*
Workflow Name: *string*
Activity Name: *string*
Run Object Name: *string*
Related MSS: *string*
Instance ID: *string*

Explanation: This message is returned in event log or ASCII log files if the workflow engine fails to publish a request to abort an activity instance to the DirX Identity server.

Action: If the cause of this communication problem is neither heavy system load nor temporary unavailability of the message server, inform DirX Identity technical support about the problem.

WFE5411 (ERR): *string*-message acknowledged with return code *number* (*string*).
Message ID: *string*,
Instance ID: *string*,
Workflow DN: *string*
Workflow Name: *string*
Activity Name: *string*
Run Object Name: *string*

Explanation: A workflow engine has received a response to a request regarding execution of an activity. The first *string* indicates the message type.

Action: If the return code is different from success, then analyze event log, ASCII log files and the status area in order to identity the problem cause.

WFE5412 (ERR): Received Message (Message ID = *string*) of unexpected type.
Instance ID: *string*,
Workflow DN: *string*
Workflow Name: *string*
Activity Name: *string*
Run Object Name: *string*

Explanation: This message is returned when the DirX Identity server folder forwards a message that does not represent a response that the workflow engine expects.

Action: Inform DirX Identity technical support about the problem.

WFE5413 (ERR): Unexpected Message (Message ID = *string*) received - ignored.
Workflow DN: *string*
Workflow Name: *string*

Explanation: This message says that the DirX Identity server folder forwarded a message that does not represent a response being expected by the workflow engine. The message does not even match with an activity currently driven by the workflow engine.

Action: Inform DirX Identity technical support about the problem.

WFE5900 (FTL): Exception occurred in *string*!

Explanation: An internal Problem occurred in the workflow engine (workflow client interface). This should not happen in normal circumstances.

Action: Inform DirX Identity technical support about the problem.

WFE5901 (WAR): EXIT Object *value*, Function *string*, return value is *number* (*string*).

Explanation: An internal workflow engine (workflow client interface) function has returned said return value.

Action: None, if the return value indicates success. Otherwise, analyze related error messages event log, ASCII log files or in the status area to identify the problem cause.

16.2. Logging Messages for the Common Java Components

CFG... Logging messages for the job framework.

JMSD... Logging messages for the Dynamic client for Messaging Provider.

JMXSOAP... Logging messages for the SOAP extension for JMX.

MSG... Logging messages for the DirX Identity Message Security (SSL).

PSF... Logging messages for the DirX Identity Provisioning Services Framework.

RPL... Logging messages for the file replacer.

STG... Logging messages for the DirX Identity storage component (the database layer used by almost all components).

SVCMGMT... Logging messages for the service management user hook for the DirX Identity connector framework.

16.2.1. Logging Messages (CFG)

This topic describes the logging messages returned by the DirX Identity Connector Framework: SPML classes, SOAP client, filters and test environment.

INF(CFG201): Program '*string1*', Version '*string2*', build '*string3*' started ***

Explanation: Display program start information

Action: None.

INF(CFG202): Program '*string1*' terminated.

Explanation: End of program reached.

Action: Check log file for warnings and errors.

INF(CFG203): Program '*string1*' terminated successfully.

Explanation: Check log file.

Action: None.

INF(CFG204): Program '*string1*' terminated with exit code *string2*..

Explanation: End of program reached, but errors or warnings occurred.

Action: Check log file for warnings and errors.

INF(CFG210): *string1* file *string2* successfully opened for reading.

Explanation: File has been opened for reading

Action: None.

INF(CFG211): Submitting *string1* request for ID=*string2*.

Explanation: The agent submits a request of the given type

Action: None.

INF(CFG212): *string1* file *string2* successfully opened for writing.

Explanation: File has been opened for writing

Action: None.

INF(CFG220): Search response contains *string1* entries.

Explanation: The number of entries contained in the search result are reported.

Action: None.

INF(CFG221): Processing entry: *string1*.

Explanation: The current entry's ID is shown for information.

Action: None.

INF(CFG222): Converting identifier of type '*string1*' to ID=*string2*.

Explanation: The LDIF file writer converted the SPML identifier to the given DN.

Action: None.

INF(CFG223): Creating identifier of type '*string1*' with ID=*string2*.

Explanation: The LDIF change reader created a SPML identifier for the DN found in the LDIF change record.

Action: None.

INF(CFG224): Decrypting value '*string1*' for '*string2*'

Explanation: The crypto transformer decrypted the given value for the given attribute

type.

Action: None.

INF(CFG225): *string1*

Explanation: A LOG message output from a Java script.

Action: None.

INF(CFG226): Statistics information:

Explanation: Prints statistics information.

Action: None.

INF(CFG227): Add requests: *string1*

Explanation: Number of add requests (successful or not).

Action: None.

INF(CFG228): Modify requests: *string1*

Explanation: Number of modify requests (successful or not).

Action: None.

INF(CFG229): Delete requests: *string1*

Explanation: Number of delete requests (successful or not).

Action: None.

INF(CFG230): Search requests: *string1*

Explanation: Number of search requests (successful or not).

Action: None.

INF(CFG231): AddEvent requests: *string1*

Explanation: Number of received add events.

Action: None.

INF(CFG232): ModifyEvent requests: *string1*

Explanation: Number of received modify events.

Action: None.

INF(CFG233): DeleteEvent requests: *string1*

Explanation: Number of received delete events.

Action: None.

INF(CFG234): Successful search operations: *string1*

Explanation: Number of successful search requests.

Action: None.

INF(CFG235): Successful add operations: *string1*

Explanation: Number of successful add requests.

Action: None.

INF(CFG236): Successful modify operations: *string1*

Explanation: Number of successful modify requests.

Action: None.

INF(CFG237): Successful delete operations: *string1*

Explanation: Number of successful delete requests.

Action: None.

INF(CFG238): Objects already up to date: *string1*

Explanation: The join engine didn't perform the modify request, because the target entry was already up to date.

Action: None.

INF(CFG239): Unsuccessful search operations: *string1*

Explanation: Number of search requests that returned a failure response.

Action: Check the error codes and texts in the corresponding log messages.

INF(CFG240): Unsuccessful add operations: *string1*

Explanation: Number of add requests that returned a failure response.

Action: Check the error codes and texts in the corresponding log messages.

INF(CFG241): Unsuccessful modify operations: *string1*.

Explanation: Number of modify requests that returned a failure response.

Action: Check the error codes and texts in the corresponding log messages.

INF(CFG242): Unsuccessful delete operations: *string1*.

Explanation: Number of delete requests that returned a failure response.

Action: Check the error codes and texts in the corresponding log messages.

INF(CFG243): Objects found: *string1*.

Explanation: The number of entries returned in search responses.

Action: None.

INF(CFG244): Requests ignored due to mapping problems: *string1*.

Explanation: The mapping result contained errors. The request was ignored.

Action: Check the mapping configuration and especially the mapping functions.

INF(CFG245): Requests ignored due to user defined filters: *string1*.

Explanation: The request matched filters given by the user.

Action: None.

INF(CFG246): Requests ignored due to missing permissions: *string1*.

Explanation: The component had not enough permissions to process the given request.

Action: Check the permissions.

INF(CFG247): Decrypting value '*string1*'.

Explanation: The crypt component decrypts the given value.

Action: None.

INF(CFG248): Add response does not contain any identifier.

Explanation: The add response from a target system connector does not contain an identifier.

Action: Make sure that the connector returns an identifier for an added entry or that this doesn't cause problems in further synchronization.

INF(CFG249): Modify response does not contain any operational attribute with contains an identifier.

Explanation: The ID List file writer needs an identifier in a modify response.

Action: Make sure that the connector returns an operational attribute named "Id".

INF(CFG250): Ignoring *string1*, requestid = *string2*.

Explanation: Display of configured checkpoint properties.

Action: None.

INF(CFG251): Checkpointing enabled: frequency = *string1*; requestId = *string2*; checkpoint file = *string3*.

Explanation: Checkpoint component logs its configuration options on start-up, if checkpointing is enabled.

Action: None.

INF(CFG252): Attribute *string1* recognized as member of the list of encrypted attributes.

Explanation: Display of list items.

Action: None.

INF(CFG253): Sending *string1* with id *string2* to url *string3*.

Explanation: The SPML Soap Connector is about to send the SPML request with the given request identifier to the given URL.

Action: None.

INF(CFG254): Read LDIF content record with DN = '*string1*'.

Explanation: The LDIF Connector has inserted the LDIF record as result entry into a SPML search response.

Action: None.

INF(CFG255): LdifConnector added entry: *string1* successfully.

Explanation: The current entry has been added successfully.

Action: None.

INF(CFG256): CsvConnector added entry: *string1* successfully.

Explanation: The current entry has been added successfully.

Action: None.

INF(CFG257): Read CSV content record with DN = '*string1*'.

Explanation: The CSV Connector has inserted the CSV record as result entry into a SPML search response.

Action: None.

INF(CFG258): Framework: Compiling script *string1* time= *string2*ms.

Explanation: JS compiling time.

Action: None.

WAR(CFG403): In record *string1* of LDIF change file: ATTRIBUTE_CONTENT not processed for DN=*string2*.

Explanation: The LDIF change file contains a content type record. The record is ignored.

Action: Check if record is correct.

WAR(CFG404): In record *string1* of LDIF change file: MODDN_CONTENT not processed for DN=*string2*.

Explanation: The LDIF change file contains a modify DN type record. The record is ignored.

Action: Check if record is correct.

WAR(CFG405): In record *string1* of LDIF change file: Unknown CONTENT for DN=*string2*.

Explanation: The LDIF change file contains a record with unknown content type. The record is ignored.

Action: Check if record is correct.

WAR(CFG406): Transformed request type not supported: *string1*.

Explanation: Transformation changed the original request to an unsupported type. The request is ignored.

Action: Correct the request transformer implementation / configuration.

WAR(CFG408): Transformed response type not supported: *string1*.

Explanation: Transformation changed the original response to an unsupported type. The response is ignored.

Action: Correct the response transformer implementation / configuration.

WAR(CFG409): Decryption of attribute value '*string1*' for attribute '*string2*' failed: *string3*

Explanation: Crypt Transformer received exception when decrypting.

Action: Encrypted input value wrong or wrong certificate?

WAR(CFG410): Writer *string1* does not support interface to write response *string2*.

Explanation: Transformation changed the original response to an extended response. The writer cannot process this response type. The response is ignored.

Action: Change transformation or use another writer, which is able to process SPML extended response.

WAR(CFG411): Request type not supported: *string1*.

Explanation: The SPML file reader encountered a request type that is not being handled. The request is ignored.

Action: Make sure that input data contain correct SPMLv1 requests.

WAR(CFG412): *string1*

Explanation: A WARNING message output from a Java script.

Action: Action depends on java script.

WAR(CFG420): IOException while parsing *string1* file: *string2*.

Explanation: An error occurred during read of LDIF change file.

Action: Make sure input LDIF change files are correct.

WAR(CFG421): Decryption of partially encrypted attribute value '*string1*' failed: *string2*.

Explanation: The filter could not get the occurrence of the encrypted token.

Action: Check the value of parameters "token delimiter" and "occurrence".

WAR(CFG422): Unsupported response type "*string1*" for this writer.

Explanation: This writer only supports add, modify, delete requests.

Action: Check the configuration.

WAR(CFG423): ClassDescriptorResolver.resolveByXMLName: element with unknown namespace: *string1*: *string2*.

Explanation: When unmarshalling an SPML request/response a XML element with unknown namespace was encountered.

Action: Check the XML document, if SPML compatible or add namespace declaration to document.

WAR(CFG424): Could not delete file *string1*.

Explanation: At the end of a successful job, CheckpointControl could not delete the checkpoint file.

Action: Most probably another process or component within this process has an open file handle! Try to delete the file manually or inform DirX Identity support!

WAR(CFG425): Wrong frequencyValue in checkpoint configuration: *string1*.

Explanation: The checkpoint configuration contains a wrong value for the last checkpoint; property: checkpoint_value.

Action: Check the delta values for the job in the configuration database: they seem to be corrupted!

WAR(CFG426): Wrong checkpoint type in checkpoint configuration: *string1*.

Explanation: The checkpoint configuration contains a wrong type for the last checkpoint value ; property: checkpoint_value.

Action: The required format is: "<nr>;<timestamp>=<ID or ATTRS>=...". Check the delta values for the job in the configuration database: they seem to be corrupted!

WAR(CFG427): value '*string1*' seems not to represent an encrypted value for attribute '*string2*'. Value passed on.

Explanation: The CryptTransformer assumes that the value has not the correct format syntax of an encrypted value.

Action: Check the value. It must start e. g. with a {'.

WAR(CFG428): Exception '*string1*' when closing LDIF reader for file '*string2*': *string3*.

Explanation: At the end of the job the LDIF connector tried to close the LDIF file. The error is ignored.

Action: Check the error message. Is the file used by another component?

WAR(CFG429): Record *string1* of LDIF file '*string2*' with DN = '*string3*' is not of type Attribute Content.

Explanation: LDIF connector expected attribute content record while performing a SearchRequest for this file. The record is ignored.

Action: Make sure that record is correct.

WAR(CFG430): Exception '*string1*' when closing CSV reader/writer for file '*string2*': *string3*.

Explanation: At the end of the job the CSV connector tried to close the CSV file. The error is ignored.

Action: Check the error message. Is the file used by another component?

ERR(CFG510): Usage: AgtSessionExe -c <confFilename> -p <program> -v <version> -d <buildDate> -Enc {ATTRIB_ADMIN_PW|ADMIN_PW|NONE} -Timeout <timeout> -Port <port> -CryptlogLevel <cryptloglevel> -AuditLevel <auditlevel>

Explanation: The Program has been called with wrong arguments

Action: Call the program as specified in the message.

FTL(CFG511): Initialization of controller stack failed: *string1*.

Explanation: The framework evaluated the job configuration and instantiated the configured classes. Should never occur with productive applications.

Action: Check the configuration for wrong class names.

FTL(CFG512): Open of controller failed: *string1*.

Explanation: The controller's open method ran onto an error.

Action: Check the controller configuration for invalid properties. Maybe a pathname does not exit.

ERR(CFG513): Closing the controller failed: *string1*.

Explanation: The controller's close method ran onto an error.

Action: Check the details, maybe it is incorrectly configured or a pathname does not exit.

FTL(CFG514): Exception *string1* when opening connectors: *string2*.

Explanation: The job controller was about to open the configured connectors, readers, writers and other components and received an exception.

Action: Check the controller configuration for invalid properties. Maybe a pathname does not exit.

FTL(CFG515): Open of connector *string1* failed: *string2*.

Explanation: The connector's open method ran onto an error.

Action: Check the details, maybe it is incorrectly configured or a pathname does not

exit.

ERR(CFG520): IOException while parsing *string1* file: *string2*.

Explanation: An error occurred during read of file.

Action: Check the detailed message. Maybe file doesn't exist or is locked by another application.

ERR(CFG552): Invalid response of *string1* operation detected in *string2*.

Explanation: An invalid response may occur due to programming errors in the connector or due to incompatible SPML versions.

Action: Inform DirX Identity technical support about the problem.

ERR(CFG553): Response of *string1* operation returns error: *string2*.

Explanation: An SPML response returns with an error.

Action: Check detailed error message.

ERR(CFG560): Unknown IdentifierType *string1* detected for ID=*string2*.

Explanation: The type of identifier returned cannot be processed correctly.

Action: Check processing of identifiers.

ERR(CFG561): Unsupported IdentifierType '*string1*' detected for ID=*string2*. Check processing of identifiers.

Explanation: LDIF File writer encountered a SPML identifier type it does not support.

Action: Either make sure the connector returns another type (not Liberty, SAML or PASSPORT) or deploy a writer with extended identifier handling.

ERR(CFG562): Unknown IdentifierType '*string1*' configured.

Explanation: The type of identifier returned cannot be processed correctly.

Action: Make sure the identifierType string complies exactly with SPMLv1.

ERR(CFG570): Exception in search request for requestID *string1*: *string2*.

Explanation: The connector returned an exception when executing the search request.

Action: Check the details and make sure the target system is accessible and the logon user has enough access rights.

ERR(CFG571): Exception in add request for requestID *string1*: *string2*.

Explanation: The connector returned an exception when executing the add request.

Action: Check the details and make sure the target system is accessible and the logon user has enough access rights.

ERR(CFG572): Exception in modify request for requestID *string1*: *string2*.

Explanation: The connector returned an exception when executing the modify request.

Action: Check the details and make sure the target system is accessible and the logon user has enough access rights.

ERR(CFG573): Exception in delete request for requestID *string1*: *string2*.

Explanation: The connector returned an exception when executing the delete request

Action: Check the details and make sure the target system is accessible and the logon user has enough access rights.

ERR(CFG574): Exception in extended request for requestID *string1*: *string2*.

Explanation: The connector returned an exception when executing the extended request.

Action: Make sure the connector supports an SPML extended request and the request is of correct format.

ERR(CFG575): Exception in batch request for requestID *string1*: *string2*.

Explanation: The connector returned an exception when executing the batch request.

Action: Check the details and make sure the connector supports a batch request.

ERR(CFG576): Exception while starting batchResponse: *string1*.

Explanation: The SPML writer caught an exception when writing the batch response start tag.

Action: Check the details. Especially make sure the file exists and is writable.

ERR(CFG577): Exception while opening basic writer: *string1*.

Explanation: The SPML writer could not be opened.

Action: Check the details. If configured for a file, especially make sure the file exists and is writable.

ERR(CFG578): Exception in processing response with ID *string1*: *string2*

Explanation: Either a response transformer or response writer returned an exception.

Action: Check the details. Especially check output files and transformer implementation (if any).

ERR(CFG579): Controller received null response for request with ID *string1*.

Explanation: The controller received a null response.

Action: Correct the connector implementation: it shall always return a response or throw an exception.

ERR(CFG580): Exception in transforming requestID *string1*: *string2*.

Explanation: The XSL-transformer returned an exception when transforming the

request.

Action: Check the XSL transformation document.

ERR(CFG581): Exception in transforming response to requestID *string1*: *string2*.

Explanation: The XSL-transformer returned an exception when transforming the response

Action: Check the XSL transformation document.

ERR(CFG583): Exception in transforming encrypted request with ID *string1*: *string2*.

Explanation: The crypt transformer was not able to decrypt an attribute of the input request.

Action: Make sure the input values are encrypted correctly and the transformer has the necessary certificates.

ERR(CFG584): in LdifFileWriter while writing searchResultEntry: response without id and no naming attribute configured.

Explanation: LDIFWriter could not generate DN of LDIF content entry due to missing ID and missing configured naming attribute.

Action: Make sure a naming attribute is configured.

ERR(CFG590): Exception *string1* in SpmlSoapProxy while sending request *string2* with id *string3*: *string4*

Explanation: An Exception occurred when sending a request to the SOAP service.

Action: Check SOAP Service: address, port, is it started?

ERR(CFG591): SpmlSoapProxy received SOAP Fault Exception. Code = *string1*, FaultString = *string2*, Detail: *string3*

Explanation: The requested Web Service is not available or returned a SOAP Fault.

Action: Check Soap Service: is it started and runs without errors?

ERR(CFG592): Exception *string1* in Spml Soap Connector while unmarshalling response for *string2* with id '*string3*': *string4*.

Explanation: An Exception occurred when sending a request to the SOAP server and reading its response.

Action: Check the details. Probably the response cannot be evaluated: is it well-formed? Does the responded Soap version match that of the request?

ERR(CFG593): No endpoint (URL) configured.

Explanation: A SOAP client requires address and port of the remote web service.

Action: Add URL or ssl, server, port and path to configuration.

ERR(CFG594): Exception while opening Soap Proxy connector: *string1*.

Explanation: An exception occurred while opening the SOAP proxy connector.

Action: Check the configuration of the Spml Soap Connector and availability of necessary libraries.

ERR(CFG602): Marshalling exception for response *string1* (id=*string1*): *string2*.

Explanation: The Spml writer encountered an exception from the Castor marshalling framework while marshalling an SPML response object.

Action: Check the detailed message and the response.

ERR(CFG603): Validation error while marshalling response *string1* (id=*string2*): *string3*.

Explanation: The Castor marshalling framework detected that the response does not comply with SPMLv1.

Action: Make sure the connector(s) return a valid SPMLv1 response.

ERR(CFG604): Exception while writing response *string1* (id=*string2*): *string3*.

Explanation: The Castor marshalling framework returned a general exception while marshalling a SPML response object.

Action: Check the message details. The problem should not be due to validation or well-formedness.

FTL(CFG605): Exception while unmarshalling requests: *string1*.

Explanation: The Castor unmarshalling framework returned a marshall exception while unmarshalling an SPML input file.

Action: Check the SPML input request.

FTL(CFG606): Validation exception while unmarshalling requests: *string1*.

Explanation: The input request does not comply with SPMLv1.

Action: Check the SPML input request.

ERR(CFG607): No request handler defined for reader *string1*.

Explanation: The SPML file reader does not have a controller to send received requests to.

Action: Inform DirX Identity Support: seems to be an error of the controller implementation.

ERR(CFG608): Exception when closing reader: *string1*.

Explanation: SPML File reader received an IO exception was caught when closing the reader.

Action: Check the input file: Does it exist?

FTL(CFG609): Validation exception while unmarshalling event: *string1*.

Explanation: Validation errors occurred while unmarshalling event.

Action: Make sure that the event contains valid SPML elements.

FTL(CFG610): Exception while unmarshalling event: *string1*.

Explanation: The Castor unmarshalling framework threw a Marshall Exception when trying to unmarshall a SPML request from an XML node.

Action: Check the message details. Make sure that the event contains valid SPML elements.

ERR(CFG611): *string1*.

Explanation: An ERROR message output from a Java script.

Action: Check the Java script for details.

ERR(CFG612): Close of *string1 string2* returned error: *string3*.

Explanation: At the end of the job the job controller tried to close all connectors / readers / writers and got an exception.

Action: Check detailed message for affected component. Check files or target systems.

ERR(CFG613): Error while executing Java Script '*string1*': *string2*.

Explanation: Execution of Java script failed.

Action: Check the Java script for details.

ERR(CFG614): Reading Java Script '*string1*' failed: *string2*.

Explanation: Reading of Java script failed.

Action: Check the configuration: reference, url, filename.

ERR(CFG615): Can't determine location of Java script file, no connection defined.

Explanation: The connection tag is missing in the controller configuration.

Action: Add the connection element with the script reference.

ERR(CFG616): Can't determine location of Java script file, neither url nor filename defined in connection.

Explanation: The Java script url / filename is missing in the controller configuration.

Action: Add the url or filename to the configuration.

ERR(CFG617): Axis Fault received in SpmlSoapProxy as response for request *string1*: *string2*.

Explanation: The remote Web Service returned an Axis Fault response.

Action: Check the message details and correct the Web Service.

ERR(CFG618): SOAP body missing in response for request *string1*.

Explanation: The remote Web Service returned a Soap response without body.

Action: Correct the Web Service.

ERR(CFG619): Can't extract message element from SOAP body in response for request *string1*.

Explanation: The remote Web Service returned a Soap response without message within body tag.

Action: Correct the Web Service.

ERR(CFG620): SpmlSoapProxy cannot extract DOM from message element in response to request *string1*. Message element: *string2*.

Explanation: The message element in the SOAP response body seems to be wrong.

Action: Check and correct the response format of the remote Web Service.

ERR(CFG630): Decryption of *string1*attribute *string2* failed for request ID=*string3*: *string4*.

Explanation: Could not decrypt an encrypted attribute. The reason is given in the details of the message.

Action: Probably the privateKey is not consistent to the certificate being used during encryption.

ERR(CFG631): Encryption of *string1*attribute *string2* failed for request ID=*string3*: *string4*. Clearing attribute value.

Explanation: Encryption failed for an attribute to be encrypted. Thus, the attribute value is cleared to prevent returning it as clear text.

Action: Check whether the certificate is valid and can be imported.

ERR(CFG632): Encryption of attributes *string1* failed: *string2*. Clearing attribute values.

Explanation: Encryption failed for some attributes to be encrypted. Thus, the attribute values are cleared to prevent returning it as clear text.

Action: Check whether the certificate is valid and can be imported.

ERR(CFG633): Encryption of attribute *string1* failed for *string2* result entry, ID=*string3* of SearchResponse ID=*string4*: *string5*. Clearing attribute value.

Explanation: Encryption failed for an attribute to be encrypted. The attribute values are cleared to prevent returning them as clear text.

Action: Check whether the certificate is valid and can be imported.

ERR(CFG634): Encryption of *string1*attribute *string2* failed for *string3* result entry of SearchResponse ID=*string4*: *string5*. Clearing attribute value.

Explanation: Encryption failed for an attribute to be encrypted. The attribute values are cleared to prevent returning them as clear text.

Action: Check whether the certificate is valid and can be imported.

ERR(CFG635): Encryption of attributes *string1* failed for entries with indices *string2* of SearchResponse ID=*string3*. Clearing attribute values.

Explanation: Encryption failed for some attributes of search response. The indicees of the erroneous entries are given. The erroneous attribute values are cleared to prevent returning them as clear text.

Action: Check whether the certificate is valid and can be imported.

ERR(CFG636): Cannot encrypt null response.

Explanation: The search response to be encrypted is null.

Action: Check your configuration or correct the connector.

ERR(CFG637): Cannot decrypt null request.

Explanation: The request to be encrypted is null.

Action: Check your configuration, the reader or the input request messages.

ERR(CFG638): Encryption of identifier attribute *string1* failed for *string2* result entry, ID=*string3* of SearchResponse ID=*string4*: *string5*. Clearing attribute value.

Explanation: Encryption failed for an identifier attribute to be encrypted. The attribute values are cleared to prevent returning it as clear text.

Action: Check whether the certificate is valid and can be imported.

ERR(CFG639): Connector *string1* does not support interface to handle SPMLv1 *string2*.

Explanation: Connector needs to implement DxmRequestor interface with methods addRequest(), modifyRequest(), etc.

Action: Take a connector class that implements this SPMLv1 request!

ERR(CFG640): Controller *string1* does not support interface to handle SPMLv1 requests.

Explanation: Controller *string1* needs to handle SPMLv1 requests: add, modify, delete, search.

Action: Configure a controller class that handles SPMLv1 requests!

ERR(CFG641): Connector *string1* does not exist.

Explanation: The given connector name has not been found in the configuration file.

Action: Check the connector names in the job configuration.

ERR(CFG642): Writer *string1* does not support interface to handle SPMLv1 *string2*.

Explanation: Writer needs to implement DxmResponseWriter interface with methods add(), modify(), etc.

Action: Configure a writer class that implements DxmResponseWriter!

ERR(CFG643): Exception *string1* when opening checkpoint file *string2*: *string3*.

Explanation: CheckpointControl could not open the checkpoint file for writing.

Action: Check access rights for file or open state!

ERR(CFG644): Invalid *string1*; RequestId = *string2*.

Explanation: CheckpointControl received an invalid Add-/Modify- or Delete Request without identifier and attributes.

Action: Check the input file!

ERR(CFG645): Exception *string1* while writing checkpoint to file: *string2*.

Explanation: CheckpointControl caught an exception while setting a new checkpoint.

Action: Check the checkpoint file! Does it exist? Have other processes a write handle?

ERR(CFG646): Basic Controller caught DxmConnectorTempException while opening connector '*string1*': *string2*.

Explanation: The connector has problems with the configuration or in binding to the target system.

Action: Check the connector configuration and make sure the target system is online.

FTL(CFG647): Connector '*string1*' failed to open LDIF file '*string2*': *string3*.

Explanation: The file cannot be opened.

Action: Check protections and existing of paths.

ERR(CFG648): *string1* when loading a *string2*: *string3*.

Explanation: When instantiating the components of the job the referred class could not be instantiated. Job loading fails.

Action: Check the jar files and the classpath.

ERR(CFG649): Cannot open *string1* file: *string2*.

Explanation: The file cannot be opened.

Action: Check protections and existing of paths.

ERR(CFG650): Logging request id=*string1* failed: *string2*.

Explanation: The LDIF change log output of the request could not be created. The request will be missing in the log.

Action: Check the message details.

ERR(CFG651): Logging response id=*string1* failed: *string2*.

Explanation: The LDIF change log output of the response could not be created. The response will be missing in the log.

Action: Check the message details.

ERR(CFG652): *string1* when instantiating class with name '*string2*': *string3*.

Explanation: When instantiating the components of the job the referred class could not be instantiated. Job loading fails.

Action: Check the jar files and the classpath.

ERR(CFG653): *string1* when loading class with name '*string2*': *string3*.

Explanation: When starting the components of the job the referred class could not be loaded from one of the class loaders. Job loading fails.

Action: Check the jar files and the classpath.

FTL(CFG654): Connector '*string1*' failed to open CSV file '*string2*': *string3*.

Explanation: The file cannot be opened.

Action: Check protections and existing of paths.

FTL(CFG655): Start of controller failed: *string1*.

Explanation: The controller's start method ran onto an error.

Action: Check the controller configuration for invalid properties. Maybe a pathname does not exit.

FTL(CFG656): Exception while unmarshalling responses: *string1*.

Explanation: The Castor unmarshalling framework returned a marshall exception while unmarshalling an SPML response.

Action: Check the SPML response.

ERR(CFG657): Marshalling exception for response: *string1*.

Explanation: The Spml writer encountered an exception from the Castor marshalling framework while marshalling an SPML response object.

Action: Check the detailed message and the response.

ERR(CFG666): *string1*.

Explanation: Generic error message.

Action: Inform DirX Identity support.

FTL(CFG900): No logging configuration found for the controller in the configuration file.

Explanation: Add a <logging .../> tag to the controller definition.

Action: Correct the configuration as described in explanation.

INF(CFG901): No log level defined in the controllers logging tag in the configuration file. Using default 5.

Explanation: Enter a level="<level>" property to the logging tag with <level> ranges from 0 to 9.

Action: Correct the configuration as described in explanation, if you want another level.

INF(CFG902): No filename defined in the controllers logging tag in the configuration file. Writing Logging to console.

Explanation: Enter a filename="<filename>" property to the logging tag with <filename> containing the absolute pathname of the trace file.

Action: Correct the configuration as described in explanation, if you want to write the log to a file.

FTL(CFG903): Cannot open trace file *string1* for writing.

Explanation: Enter a filename="<filename>" property to the logging tag with <filename> containing the absolute pathname of the trace file.

Action: Correct the configuration as described in explanation.

FTL(CFG904): Cannot determine filename of *string1* file, no connection defined for the reader.

Explanation: Add a <connection> tag to the definition of the reader.

Action: Correct the configuration as described in explanation.

FTL(CFG905): Cannot determine filename of *string1* file, not defined in connection.

Explanation: Add a filename="<filename>" property to the connection tag with <filename> containing the absolute pathname of the file to be read.

Action: Correct the configuration as described in explanation.

FTL(CFG906): Cannot open *string1* file: *string2*.

Explanation: The file cannot be opened.

Action: Check protections and existing of paths.

FTL(CFG907): No port configured to get PSE.

Explanation: No port configured to get PSE.

Action: Configure a port for the PSE.

FTL(CFG912): Cannot decrypt parameter '*string1*' with value '*string2*': *string3*.

Explanation: Password used in the job configuration cannot be decrypted.

Action: Check your passwords and the certificates or the key- and trust-stores!

FTL(CFG913): Exception occurred while processing configuration of *string1*.

Explanation: Password used in the given job configuration cannot be decrypted.

Action: See other messages.

FTL(CFG914): No stylesheet defined for XSL transformer in the configuration.

Explanation: XSL transformation needs a XSL stylesheet.

Action: Enter a filename="<filename>" attribute to the connection element containing the absolute pathname of the stylesheet file.

FTL(CFG915): Can't determine filename of *string1* file, no connection defined for the reader.

Explanation: The file writer needs a <connection> tag with a file name attribute.

Action: Add a <connection> tag to the definition of the writer.

FTL(CFG916): Can't determine filename of *string1* file, not defined in connection.

Explanation: The file writer needs a <connection> tag with a file name attribute.

Action: Add a filename="<filename>" attribute to the connection tag with <filename> containing the absolute pathname of the file to be written.

FTL(CFG917): Connection tag missing in configuration of connector.

Explanation: Each connector requires a <connection> tag in its configuration.

Action: Add a <connection> tag to the definition of the connector.

FTL(CFG918): Error while parsing *string1* file *string2*: *string3*.

Explanation: The Event reader tried to read an XML file and received a parsing exception.

Action: Check the XML document.

FTL(CFG919): Open of reader '*string1*' failed: Can't open *string2* file: *string3*.

Explanation: The reader cannot open the input file.

Action: Check protections and existing of paths.

FTL(CFG920): Error in job configuration: Can't build stack: *string1*.

Explanation: Building of job connectors failed. Maybe a classname is wrong in the configuration.

Action: Check class names in configuration and required jar files.

FTL(CFG921): Can't load configuration: *string1*.

Explanation: Can't load job configuration. The configuration file or mapping file cannot be read.

Action: Check file names in command line for job.

FTL(CFG922): Can't map configuration: *string1*.

Explanation: Errors while mapping configuration classes. The mapping file is inconsistent to the software version or to the configuration.

Action: Check the configuration format with the configuration XML schema.

FTL(CFG923): Syntax error in configuration: *string1*.

Explanation: The Castor unmarshalling framework threw a MarshalException while reading the job configuration.

Action: Check the configuration format with the configuration XML schema.

FTL(CFG924): Validation error in configuration: *string1*.

Explanation: The job configuration does not comply with the configuration XML schema.

Action: Check the configuration format with the configuration XML schema.

FTL(CFG925): Aborted. Reason: *string1*.

Explanation: A fatal error has occurred while running the framework job. Program execution has been stopped.

Action: Check the previous error messages. Probably the error originated in some connector or the job controller.

FTL(CFG926): Import of private keys via socket failed: *string1*.

Explanation: The agent has been called with encryption options.

Action: Check private key provider (MSS) and socket number.

FTL(CFG927): Execution aborted due to missing of class: *string1*.

Explanation: The agent has aborted due to a missing class.

Action: Check installation: classpath in the run script and existence of required jar files.

FTL(CFG928): Out of memory error detected. Execution aborted.

Explanation: The agent ran out of memory.

Action: Increase the -XmxMMMm option of the agent's run script.

FTL(CFG929): Framework does not run in encryption mode but value is strong encrypted:'*string1*'.

Explanation: A strong encrypted value cannot be processed because the framework is running in encryption mode NONE.

Action: Check the configuration. If this is a stand-alone executable check global encryption mode and attribute 'disable encryption' of associated bind profile.

FTL(CFG930): Evaluating the JSON response failed: attribute name *string1*, value to evaluate: '*string2*'.

Explanation: The JSON response from the service couldn't be correctly evaluated.

Action: Check the JSON response from the service.

16.2.2. Logging Messages (JMSD)

This topic describes the logging messages returned by the Dynamic client for Messaging Provider.

ERR(JMSD100): Could not get valid JMS configuration.

Explanation: The JMS system was unable to read information about active message server.

Action: If the issue continues check if your LDAP server for connectivity configuration is running and if e. g. no firewall blocks any connection.

ERR(JMSD101): Missing JMS configuration data.

Explanation: The JMS system was unable to read valid information about active message server (host, port, SSL).

Action: If the issue continues check if your JMS configuration is correctly configured, check activated service, host, port and SSL property of the JMS service.

ERR(JMSD102): No configuration object with search base *string1* and filter *string2* was found.

Explanation: The JMS system was unable to read described information about active message server.

Action: If the issue continues check if your JMS configuration is correctly configured, check activated service, host, port and SSL property of the JMS service.

ERR(JMSD103): Could not find a valid value for entry *string1* and attribute *string2*.

Explanation: The JMS system was unable to read described information about active message server.

Action: If the issue continues check if your JMS configuration is correctly configured, check activated service, host, port and SSL property of the JMS service.

ERR(JMSD104): More configuration objects with search base *string1* and filter *string2* was found.

Explanation: The JMS system was unable to read unique information about active message server.

Action: If the issue continues check if your JMS configuration is correctly configured, check activated service, host, port and SSL property of the JMS service.

ERR(JMSD105): No valid configuration object with search base *string1* and filter *string2* was found.

Explanation: The JMS system was unable to read described information about active

message server. No or multiple entries was found.

Action: If the issue continues check if your JMS configuration is correctly configured, check activated service, host, port and SSL property of the JMS service.

ERR(JMSD106): Could not read object with search base *string1* and filter *string2* due to error *string3*.

Explanation: The JMS system was unable to read described information about active message server due to an error.

Action: If the issue continues check if your LDAP server for connectivity configuration is running and if e. g. no firewall blocks any connection.

ERR(JMSD107): Cannot connect to the LDAP server *string1*:_string2_ as user *string3*, where the DirX Identity connectivity DB is installed due to an error *string4*.

Explanation: The JMS system was unable to read described information about active message server due to an error.

Action: If the issue continues check if your LDAP server for connectivity configuration is running and if e. g. no firewall blocks any connection.

ERR(JMSD108): Cannot get connection to the LDAP server *string1*:_string2_ as user *string3*, where the DirX Identity connectivity DB is installed.

Explanation: The JMS system was unable to read described information about active message server due to an error.

Action: If the issue continues check if your LDAP server for connectivity configuration is running and if e. g. no firewall blocks any connection.

ERR(JMSD110): JMS problem when creating a JMS message; body: *string1*; error: *string2*.

Explanation: There is a problem with JMS when creating a JMS message.

Action: Check the JMS problem.

ERR(JMSD111): JMS problem when creating the queue with name '*string1*'; error: *string2*.

Explanation: There is a problem with JMS when creating an AciveMQ queue.

Action: Check the JMS problem.

ERR(JMSD112): JMS problem when creating the JMS message producer object; error: *string1*.

Explanation: There is a problem with JMS when creating an ActiveMQ producer.

Action: Check the JMS problem.

ERR(JMSD113): JMS problem when creating the JMS message consumer object; error: *string1*.

Explanation: There is a problem with JMS when creating an ActiveMQ consumer.

Action: Check the JMS problem.

ERR(JMSD114): JMS problem when creating the JMS connection object with client id '*string1*'; error: *string2*.

Explanation: There is a problem with JMS when creating an ActiveMQ connection.

Action: Check the JMS problem.

ERR(JMSD115): JMS problem when creating the JMS topic connection object with client id '*string1*'; error: *string2*.

Explanation: There is a problem with JMS when creating an ActiveMQ topic connection.

Action: Check the JMS problem.

ERR(JMSD116): JMS problem when creating the JMS session object; error: *string1*.

Explanation: There is a problem with JMS when creating an ActiveMQ session.

Action: Check the JMS problem.

ERR(JMSD117): JMS problem when creating the JMS topic session object.

Explanation: There is a problem with JMS when creating an ActiveMQ topic session.

Action: Check the JMS problem.

ERR(JMSD118): JMS problem when creating the topic with name '*string1*'; error: *string2*.

Explanation: There is a problem with JMS when creating a JMS topic.

Action: Check the JMS problem.

ERR(JMSD119): Problem while getting SSL flag from JMS broker configuration; error: *string2*.

Explanation: There is a problem when extracting the SSL flag from the current broker configuration

Action: Check the JMS problem.

ERR(JMSD120): *string1* exception when sending message; error: *string2*; trying to reestablish connection/session.

Explanation: Sending of a JMS messages failed.

Action: Check the JMS problem.

ERR(JMSD121): *string1* exception when reconnecting; error: *string2*.

Explanation: Reconnecting to JMS messaging system failed.

Action: Check the JMS problem.

ERR(JMSD122): *string1* exception when publishing message; error: *string2*; trying to reestablish connection/session.

Explanation: Publishing of a JMS messages failed.

Action: Check the JMS problem.

ERR(JMSD123): *string1* The environment property DIRXIDENTITY_INST_PATH is missing, cannot get access to the keys required for a JMS connection secured by SSL/TLS.

Explanation: The component is probably running on a host without definition of the DIRXIDENTITY_INST_PATH environment property which is necessary to get cryptographic keys for SSL/TLS connections to a configured ActiveMQ broker.

Action: Define the environment property DIRXIDENTITY_INST_PATH as an existing path within the file system and create a folder DIRXIDENTITY_INST_PATH/ssl. Create or copy the key material to this folder (identity-keystore, identity-truststore, password.properties), see the documentation for more detail. Restart the erroneous component.

ERR(JMSD124): Closing the JMS connection '*string1*' failed; error: *string2*. Instance: *string3*, Current number of connections: *string4*.

Explanation: JMS Connection could not be closed successfully.

Action: Check the error message.

WAR(JMSD300): Retrying (*string1*.) to read object with search base *string2* and filter *string3*, last exception was *string4*.

Explanation: The JMS system was unable to read described information about active message server. Retrying.

Action: If the issue continues check if your LDAP server for connectivity configuration is running and if e. g. no firewall blocks any connection.

WAR(JMSD301): Cannot connect to the LDAP server *string1*:_string2_ as user *string3*, where the DirX Identity connectivity DB is installed due to a temporary error *string4*, *string5*. retry will be performed.

Explanation: The JMS system was unable to read described information about active message server due to an error.

Action: If the issue continues check if your LDAP server for connectivity configuration is running and if e. g. no firewall blocks any connection.

WAR(JMSD302): Cannot close *string1*.

Explanation: An object could not be closed.

Action: See other messages why the session or connection could not be closed.

INF(JMSD500): JMS configuration successfully read; host: *string1*, port: *string2*, SSL: *string3*, URL (for clients): *string4*.

Explanation: The JMS messaging information is shown.

Action: None

INF(JMSD501): Successfully created new JMS connection '*string1*'. Instance: *string2*, Current number of connections: *string3*.

Explanation: Connection has been created.

Action: None.

INF(JMSD502): Successfully closed JMS connection '*string1*'. Instance: *string2*, Current number of connections: *string3*.

Explanation: Connection has been closed.

Action: None.

INF(JMSD503): Successfully closed *string1* JMS connections. Failed to close *string2* JMS connections.

Explanation: Connections have been closed.

Action: None.

16.2.3. Logging Messages (JMXSOAP)

This topic describes the logging messages returned by the SOAP extension for JMX.

ERR(JMXSOAP001): Could not parse configured URL '*string1*' for JMX SOAP requests.

Explanation: The WebAdmin could not use the configured or default URL for sending JMX requests via SOAP.

Action: Check the configuration of the JMX service for the C++-based server.

ERR(JMXSOAP002): Could not marshal the generated JMX request.

Explanation: An internal error occurred while marshalling the JMX request.

Action: Inform DirX Identity technical support about the problem.

ERR(JMXSOAP003): Could not unmarshal the received JMX response.

Explanation: An internal error occurred while unmarshalling the JMX request.

Action: Inform DirX Identity technical support about the problem.

WAR(JMXSOAP101): An error occurred while sending the JMX request via the SOAP connection to '*string1*'.

Explanation: An error occurred and the JMX request could not be correctly processed.

Action: Check the error, check also the availability of the C++-based server.

WAR(JMXSOAP102): An error occurred while extracting the JMX response from the SOAP response.

Explanation: An error occurred and the JMX request could not be correctly processed.

Action: Check the error, check also the availability of the C++-based server.

WAR(JMXSOAP103): The JMX response contained an invalid object name '*string1*'.

Explanation: The C++ server returned an invalid object name.

Action: Inform DirX Identity technical support about the problem.

WAR(JMXSOAP104): A connection error occurred while sending the JMX request via the SOAP connection to '*string1*': *string2*

Explanation: An error occurred and the JMX request could not be correctly processed.

Action: Check the error, check also the availability of the C++-based server.

16.2.4. Logging Messages (MSG)

This topic describes the logging messages returned by the DirX Identity Message Security.

ERR(MSG500): Exception while initializing crypt support: *string1*.

Explanation: Errors occurred while initializing crypt support.

Action: Check the detailed message. Call DirX Identity support.

ERR(MSG501): Exception while decrypting password: *string1*.

Explanation: Errors occurred while decrypting a password.

Action: Check the detailed message. Check the file password.properties. Call DirX Identity support.

ERR(MSG502): Path '*string1*' not found in the file security.ini.

Explanation: Path not found or incorrect.

Action: Check the file security.ini.

ERR(MSG503): Exception while encrypting password: *string1*.

Explanation: Errors occurred while encrypting a password.

Action: Check the detailed message. Check the file password.properties. Call DirX Identity support.

ERR(MSG504): File '*string1*' not found.

Explanation: File not found.

Action: Check the file and the path in the file security.ini.

16.2.5. Logging Messages (PSF)

This topic describes the logging messages returned by the DirX Identity Provisioning Services Framework: request / response classes, SOAP client and test environment.

ERR(PSFAUTH001): Missing mandatory configuration parameter for *string1*.

Explanation: The configured authentication class could not read mandatory configuration parameters.

Action: Correct configuration of the authentication class, see documentation for more details.

ERR(PSFAUTH002): The keystore *string1* for *string2* was not found or is not accessible.

Explanation: The configured authentication class could not read the configured keystore.

Action: Correct configuration of the authentication class, check access rights.

ERR(PSFAUTH003): An exception has occurred while initializing the keystore *string1* for *string2*.

Explanation: The configured authentication class could not initialize the configured keystore.

Action: Correct configuration of the authentication class, check access rights, inspect the exception message.

ERR(PSFAUTH004): An exception has occurred while initializing the cryptographic tools for *string1*.

Explanation: The configured authentication class could not initialize the cryptographic tools.

Action: Correct configuration of the authentication class, check access rights, inspect the exception message.

ERR(PSFAUTH005): No private key with the alias *string1* was found in the configured keystore for *string2*.

Explanation: The configured authentication class could not find private key with the configured alias name.

Action: Correct configuration of the authentication class, verify that the key alias is correct.

ERR(PSFAUTH006): An exception has occurred while encrypting data for *string1*.

Explanation: The configured authentication class could not encrypt data.

Action: Correct configuration of the authentication class, inspect the exception message.

ERR(PSFAUTH007): An exception has occurred while initializing the class *string1*.

Explanation: The configured authentication class could not be initialized.

Action: Correct configuration of the authentication class, inspect the exception message.

ERR(PSFAUTH008): An exception has occurred while processing the outgoing message in the class *string1*.

Explanation: The configured authentication class could not process outgoing message.

Action: Correct configuration of the authentication class, inspect the exception message.

ERR(PSFAUTH009): An exception has occurred while processing the incoming message in the class *string1*.

Explanation: The configured authentication class could not process incoming message.

Action: Correct configuration of the authentication class, inspect the exception message.

FTL(PSFBXFR001): Cannot determine filename of *string1* file, no connection defined for the reader.

Explanation: The basic XML File Reader fails in determination of input file name because of said reason. The related file type is indicated by *string1*.

Action: Add a <connection> tag to the definition of the reader.

FTL(PSFBXFR002): Cannot determine filename of *string1* file, not defined in connection.

Explanation: The basic XML File Reader cannot determine the input file name because of the reason indicated in the message. The related file type is indicated by *string1*.

Action: Add a filename="<filename>" property to the connection tag with <filename> containing the pathname of the file to be read.

FTL(PSFBXFR003): Open of reader '*string1*' failed: Cannot open *string2* file: *string3*

Explanation: The basic XML File Reader cannot open the file *string3*.

Action: Check permissions and existence of pathnames.

INF(PSFBXFR004): *string1* file *string2* successfully opened for reading.

Explanation: The basic XML File Reader has opened the file *string2* for reading.

Action: None.

ERR(PSFBXFR005): Exception when closing reader: *string1*

Explanation: An exception was caught when closing the basic XML File reader, *string1* represents the details.

Action: Check the message to see if the cause of the problem is obvious and can be solved by administrative means and/or appropriate configuration of your application, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

ERR(PSFBXFR006): No request handler defined for reader *string1*.

Explanation: No request handler is defined in the configuration for the reader indicated in the message.

Action: Define request handler and try again.

FTL(PSFBXFR007): Exception while unmarshalling requests: *string1*

Explanation: The basic XML File Reader found errors while unmarshalling XML input file.

Action: Check the message to see if the cause of the problem is obvious and can be solved by administrative means and/or appropriate configuration of your application, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

FTL(PSFBXFR008): Validation exception while unmarshalling requests: *string1*

Explanation: The basic XML File Reader found validation errors while unmarshalling SPML input file.

Action: Check the message to see if the cause of the problem is obvious, then correct the SPML input file to fix the problem. Otherwise, inform DirX Identity technical support about the problem.

ERR(PSFCDR100): ClassDescriptorResolver.resolveByXMLName: Element with unknown namespace: '*string1*':'*string2*'.

Explanation: Read XML element with unexpected name in namespace urn:siemens:dxm:provisioning:

Action: Read XML element with unexpected name in namespace urn:siemens:dxm:provisioning:

FTL(PSFPFR001): Controller *string1* does not support interface DxmProvControllerProvisioning to handle Provisioning requests.

Explanation: Controller *string1* needs to implement interface DxmProvControllerProvisioning method "processProvRequest()".

Action: Configure an appropriate controller class.

INF(PSFPSOP001): Sending Provisioning request *string1* with id = *string2* to *string3*.

Explanation: Provisioning Soap Client sends request to URL *string3*.

Action: None.

ERR(PSFPSOP002): *string1*: SOAP fault detected: *string2*

Explanation: SOAP client caught SOAPFaultException.

Action: Check URL or peer!

ERR(PSFPSOP003): *string1*: Caught Exception *string3* when sending Soap Request: *string2*

Explanation: SOAP client caught Exception.

Action: Check data, URL or peer!

ERR(PSFPSOP004): ProvSoapClient received null response for request *string1* with ID *string2*.

Explanation: The remote soap service did not return a response for provisioning request.

Action: If the problem cause is not temporary unavailability of the remote soap service, inform DirX Identity technical support about the problem.

ERR(PSFPSOP005): ProvSoapClient caught AxisFault while reading SOAP body from response for request *string1*: *string2*

Explanation: SOAP client caught AxisFault, may be the response is corrupted.

Action: If the problem cause is not temporary unavailability of the remote soap service, inform DirX Identity technical support about the problem.

ERR(PSFPSOP006): ProvSoapClient received Soap response with missing Body for request *string1* with ID *string2*: *string3*

Explanation: Missing Body in Soap Response.

Action: Check Soap Service.

ERR(PSFPSOP007): ProvSoapClient received Soap response with missing Element for request *string1* with ID *string2*: *string3*

Explanation: Missing Element in Soap Response Body.

Action: Check Soap Service.

ERR(PSFPSOP008): ProvSoapClient cannot extract DOM from Soap Response element for request *string1* with ID *string2*: *string3*

Explanation: The Soap client fails in extracting the contents of a soap response. Possible problem cause is a corrupted Soap Response Body.

Action: Check Soap Response Body.

ERR(PSFPSOP009): ProvSoapClient caught exception while unmarshalling Provisioning response from Soap Response for request *string1* with ID *string2*: *string3*

Explanation: Soap Response Body: invalid Provisioning Response?

Action: Inform DirX Identity technical support about the problem.

ERR(PSFPSOP010): No endpoint (URL) configured.

Explanation: Configuration does not specify destination URL.

Action: Add URL to configuration.

ERR(PSFPSOP011): Exception while opening Soap Client: *string1*.

Explanation: An exception occurred while opening the SOAP client.

Action: Check if configuration matches your environment: correct URL, Soap Service started, etc.

FTL(PSFPTPC001): First connector *string1* cannot handle Provisioning Requests.

Explanation: The class configured for connector *string1* cannot process provisioning requests.

Action: Configure a connector class that can process provisioning requests.

FTL(PSFPTPC002): Exception *string1* in provisioning request *string2* for requestID *string3*: *string4*

Explanation: Connector threw Exception when processing provisioning request.

Action: Check the message to see if the cause of the problem is obvious and can be solved by correction of the request, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

WAR(PSFPTPC003): Controller creates default error response upon connector exception for request *string1* with requestID *string2*.

Explanation: Connector threw DxmConnectorException when processing provisioning request; processing continues with error response.

Action: None.

ERR(PSFPTPC004): Controller received null response for request *string1* with ID *string2*.

Explanation: The connector returned no response for provisioning request; controller generates error response.

Action: Check the message to see if the cause of the problem is obvious and can be solved by administrative means, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

ERR(PSFPTPC005): Exception in processing response *string1* with ID *string2*: *string3*

Explanation: Some Writer threw exception when processing response; continue with next request.

Action: Inform DirX Identity technical support about the problem.

ERR(PSFPTPC006): Response writer *string1* does not support writing processing response *string2* with ID *string3*.

Explanation: Do not write response to this channel; continue with next request.

Action: Inform DirX Identity technical support about the problem.

ERR(PSFPTPC007): Exception writing BatchResponse Start-Tag for requestid *string1*: *string2*

Explanation: Do not write response to this channel; probably not well-formed XML.

Action: Check the message to see if the cause of the problem is obvious and can be solved by correction of the request file, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

INF(PSFPTPC008): Passing Request *string1* to connector *string2*.

Explanation: Passing provisioning request to connector for processing.

Action: None.

INF(PSFPTPC009): Receiving Response *string1* from connector.

Explanation: Connector returned provisioning response.

Action: None.

ERR(PSFPTPC010): Exception writing BatchResponse End-Tag: *string2*

Explanation: Cannot write batchResponse; probably not well-formed XML.

Action: Check the message to see if the cause of the problem is obvious and can be solved by correction of the request file, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

ERR(PSFTCON002): Connection tag missing in configuration of connector.

Explanation: Test Connector execution fails due to the problem indicated in the message.

Action: Add connection tag to connector configuration.

ERR(PSFTCON003): Open of test connector *string1* failed: *string2*

Explanation: Test Connector execution fails due to the problem indicated in the message.

Action: Correct filenames and connector configuration to fix the problem.

FTL(PSFURPC001): Cannot instantiate response for request *string1*: *string2*

Explanation: ResponseCreator caught exception when guessing response class name based on request. Possible problem cause is a different name schema.

Action: Inform DirX Identity technical support about the problem.

FTL(PSFURPC002): Cannot instantiate response *string1*: *string2*

Explanation: ResponseCreator caught exception when instantiating response class based on class name. Possible problem cause is a different name schema.

Action: Inform DirX Identity technical support about the problem.

ERR(PSFUSER001): Exception when serializing provisioning request/response: *string1*

Explanation: Object cannot be marshalled to string, may be it is not valid.

Action: Inform DirX Identity technical support about the problem.

16.2.6. Logging Messages (RPL)

This topic describes the logging messages returned by the File replacer. This tool is used during configuration, migration and patch procedures.

INF(RPL200): Program '*string1*', Version '*string2*' of '*string3*' started ***.

> **Explanation**: Start message.
>
> **Action**: None.

INF(RPL201): Program '*string1*' ended with *string2* fatalError(s), *string3* error(s), and *string4* warning(s).

> **Explanation**: Informs the user that the program ended correctly and displays error statistics.
>
> **Action**: None.

INF(RPL202): The following files are used as source files for replacement:

> **Explanation**: Informs on the source files being processed.
>
> **Action**: None.

INF(RPL203): *string1*.

> **Explanation**: A filename of a source file.
>
> **Action**: None.

INF(RPL204): The following files are replaced by the source files:

> **Explanation**: Informs on the target files being replaced.
>
> **Action**: None.

INF(RPL205): The target directory *string1* contains patches. The patches are NOT overwritten.

> **Explanation**: Informs on patches in the target tree that are not being overwritten.
>
> **Action**: None.

INF(RPL206): File *string1* has been successfully replaced.

> **Explanation**: Information that the file has been replaced successfully.
>
> **Action**: None.

INF(RPL207): File *string1* already exists. Content will be checked.

> **Explanation**: The file has been replaced successfully after checking the file content in detail.
>
> **Action**: None.

INF(RPL208): Content of existing file *string1* is OK.

Explanation: The content of the file is the same.

Action: None.

INF(RPL209): Existing files will be stored as backup copy in the folder *string1*.

Explanation: A backup copy of the file is created in the backup folder before the file is replaced.

Action: None.

INF(RPL210): File '*string1*' successfully updated.

Explanation: Update of install_history.txt file successfully.

Action: None.

INF(RPL211): Skipping the folder '*string1*' in the target directory *string2*.

Explanation: Informs on the given folder in the target tree that is not considered further.

Action: None.

INF(RPL212): Skipping folder ending with '.V8.n' (where n is a number and/or 'nSPn') in the target directory *string1*.

Explanation: Informs on '*.V8.n' folder in the target tree that is not considered further.

Action: None.

WAR(RPL400): The source folder contains '*string1*' that is not a file. Ignoring '*string1*'.

Explanation: A directory is part of the source directory.

Action: The directory and its contents are ignored.

WAR(RPL401): Running in test mode. Files are NOT copied!

Explanation: ReplaceIt runs in test mode. Files are not copied. Remove -test option to run in process mode.

Action: None.

WAR(RPL402): Could not set modify time stamp on copied file *string1*.

Explanation: The file has been successfully copied but its time stamp could not be modified.

Action: Check if a program using this file is running (for example IdS-J, Tomcat). Stop all running programs and repeat ReplaceIt.

ERR(RPL500): The target directory *string1* is not readable.

Explanation: This directory and its subtree cannot be processed because it is not

readable.

Action: Check and correct the access rights.

ERR(RPL501): Source and target files *string1* are identical, the file was not updated.

Explanation: The source file and the target file are identical. This may occur if the source path is contained in the target definitions.

Action: Check your -src / -tgt options.

ERR(RPL502): Cannot replace file *string1*: *string2*.

Explanation: Copying the file failed. Maybe the target file is corrupted now.

Action: Check and correct the access rights. Repeat run of ReplaceIt until it runs without errors or copy the file manually.

ERR(RPL504): Verification failed for file *string1*.

Explanation: The file has been copied but verification of the copied file failed. The copied file is corrupt.

Action: Repeat run of ReplaceIt until it runs without errors or copy the file manually.

ERR(RPL505): Cannot verify file *string1*: *string2*.

Explanation: An error occurred while verifying the copied file.

Action: Check details and correct the error. Repeat the run of ReplaceIt until it runs without errors or copy the file manually.

ERR(RPL506): Content of existing file *string1* is different; check with source file *string2*.

Explanation: The content of the file is different, but should be the same.

Action: Check the files.

ERR(RPL507): Verification failed for file '*string1*'; error: *string2*.

Explanation: The file has been copied but verification of the copied file failed. The copied file is corrupt.

Action: Rerun of ReplaceIt until it runs without errors or copy the file manually.

ERR(RPL508): Creating folder '*string1*' as backup folder failed.

Explanation: Creating the backup directory failed.

Action: Create the backup folder manually.

ERR(RPL509): Could not find file '*string1*'.

Explanation: The 'readme.txt' file could not be found in the current working directory. The update of the install_history.txt file could not be done.

Action: Check for the file. Update the install_history.txt file manually.

ERR(RPL510): Could not find file '*string1*'.

Explanation: The 'install_history.txt' file could not be found in the installation directory. The update of the install_history.txt file could not be done.

Action: Check for the file. Update the install_history.txt file manually.

ERR(RPL511): The environment variable '*string1*' is not set.

Explanation: The variable is mandatory. The update of the install_history.txt file could not be done.

Action: Check and change the environment setting. Update the install_history.txt file manually.

ERR(RPL512): Could not find or read the file '*string1*'.

Explanation: The readme.txt file could not be found or was not accessible in the current working directory. The update of the install_history.txt file could not be done.

Action: Check for the file. Update the install_history.txt file manually.

ERR(RPL513): The line '*string1*' does not match. Date '*string2*', version'*string3*', Account '*string4*', message text '*string5*'.

Explanation: The line which is looked up in the readme.txt file to put in into the install_history.txt file does not match. The update of the install_history.txt file could not be done.

Action: Update the install_history.txt file manually.

ERR(RPL514): The line '*string1*' does not match.

Explanation: The line which is looked up in the readme.txt file to put in into the install_history.txt file does not match. The update of the install_history.txt file could not be done.

Action: Update the install_history.txt file manually.

ERR(RPL515): Could not read the file '*string1*'; error: *string2*.

Explanation: The install_history.txt file could not be read.

Action: Check the error message. This situation is not a serious problem, but may result in the fact that the hotfix entry is not listed on a new line.

FTL(RPL900): Usage: ReplaceFiles [-test] -src <Source Dir> -tgt <TargetDir> [<TargetDir> <TargetDir> ... <TargetDir>] [-saveDir <backupDir>] [-instHist].

Explanation: The program was called with incorrect options.

Action: Call program according to the usage above.

FTL(RPL903): Cannott open log file '*string1*' for writing: *string2*.

Explanation: The log file could not be opened.

Action: Check whether the file already exists and is in use.

FTL(RPL904): The source directory *string1* does not exist.

Explanation: The option -src <Source Dir> must point to the directory containing the source files for replacements.

Action: Correct the source dir option.

FTL(RPL905): The source directory *string1* is not a directory.

Explanation: The directory provided with -src <Source Dir> is not a directory.

Action: Correct the pathname.

FTL(RPL906): The source directory *string1* is not readable.

Explanation: The directory provided with -src <Source Dir> is not readable.

Action: Give read access to the source dir.

FTL(RPL908): The target directory *string1* is not a directory.

Explanation: The directory provided with -tgt <Target Dir> is not a directory.

Action: Correct the target dir option.

FTL(RPL909): The file *string1* does not exist.

Explanation: The file provided with -excl option does not exist.

Action: Correct the excl option.

FTL(RPL910): Error while reading from file *string1*. Error: *string2*.

Explanation: Reading from file failed.

Action: Check and correct the error.

16.2.7. Logging Messages (STG)

This topic describes the logging messages returned by the DirX Identity LDAP-based Storage.

ERR(STG000): Unexpected exception: '*string1*'.

Explanation: An exception occurred. Check the message text for details.

Action: None.

WAR(STG002): *string1*

Explanation: Encountered a recoverable error situation.

Action: Check the message text for details.

ERR(STG003): *string1*

Explanation: A severe error occurred. Check the message text for details.

Action: Inform DirX Identity technical support about the problem.

FTL(STG004): *string1*

Explanation: A fatal error occurred, the system won't work properly. Check the message text for details.

Action: Inform DirX Identity technical support about the problem.

ERR(STG050): Garbage Collector freed modified/transacted/new object: *string1*

Explanation: Internal error, all modifications to the referred object are lost.

Action: Inform DirX Identity technical support about the problem.

ERR(STG051): Couldn't refresh '*string1*' (cause: *string2*).

Explanation: Inconsistent in-memory representation of an LDAP entry.

Action: Check the reason described in the detailed message.

ERR(STG052): Failed to read children of *string1* (cause: *string2*).

Explanation: An LDAP error occurred during reading subordinated entries.

Action: Check the reason described in the detailed message. Check server and network connections.

WAR(STG053): getProperty('*string1*') failed (cause: *string2*).

Explanation: Unable to retrieve requested property.

Action: Check the reason described in the detailed message.

WAR(STG054): setProperty(*string1*) failed (cause: *string2*).

Explanation: Unable to retrieve requested property.

Action: Check the reason described in the detailed message.

ERR(STG055): Type-conversion of property '*string1*' failed (cause: *string2*).

Explanation: Configuration error, failed to perform type conversion.

Action: Check the configuration i.e. the Object Descriptions

ERR(STG056): Failed to add new object '*string1*' to the LDAP store (cause: *string2*).

Explanation: Severe error, couldn't add a new entry to the LDAP store.

Action: Check the reason described in the detailed message. Check server and network connections.

INF(STG057): Naming attribute of '*string1*' changed (old: *string2*, new: *string3*).

Explanation: Prior to modifying the object, the storage will perform a rename operation.

Action: None.

INF(STG058): Created superior object '*string1*'.

Explanation: In order to add the object to the LDAP store, a superior object had to be created.

Action: None.

WAR(STG059): Display name '*string1*' of '*string2*' isn't unique!

Explanation: The storage enforces per-level unique display names. It detected a violation of this regulation.

Action: Choose a unique display name.

ERR(STG060): Failed to check unique display name of object '*string1*' (cause: *string2*).

Explanation: The storage enforces per-level unique display names. Failed to check this.

Action: Check the reason described in the detailed message.

ERR(STG061): Detected a problem while converting object '*string1*' to XML (cause: *string2*).

Explanation: Internal error, object inconsistent.

Action: Check the reason described in the detailed message, Inform DirX Identity technical support about the problem.

WAR(STG062): getPropertyFromPath: content of '*string1*' doesn't reference an LDAP-entry!

Explanation: Wrong reference, the referred object doesn't exist.

Action: In case the error occurred while processing a report, check the report configuration.

INF(STG063): *string1* while running the script '*string2*'; error: *string3*.

Explanation: An error occurred when running a JAVA script.

Action: Check the error.

WAR(STG101): Object Descriptor '*string1*' referred an undefined parent '*string2*'.

Explanation: Configuration error, the parent Object Descriptor isn't defined.

Action: Check the configuration.

WAR(STG102): No <mapping> section in Object Descriptor *string1*, class=*string2*

Explanation: Mappings link LDAP entries with their associated Object Descriptors. If this information is missing, LDAP entries won't be presented properly.

Action: Check the configuration.

ERR(STG103): Detected problems while reading XML-formatted Object Descriptors (cause: *string1*).

Explanation: An IO-error occurred, the configuration wasn't completely read.

Action: Check the configuration, check network connections.

ERR(STG104): Detected problems while parsing XML-formatted content: '*string1*' (cause: *string2*).

Explanation: Incorrect configuration (e.g. not well-formed XML).

Action: Check the configuration.

ERR(STG105): Detected problems while parsing XML-formatted content '*string1*': *string2*.

Explanation: Incorrect configuration (e.g. not well-formed XML).

Action: Check the configuration.

ERR(STG106): Alias: Must be inside <globalproperties> or <object>.

Explanation: Alias out of scope, should be enclosed in a 'globalproperties' or 'object' tag.

Action: Check the configuration i.e. the Object Descriptors.

ERR(STG107): Alias: invalid format. Should be: <alias abbreviation="..." value="...".

Explanation: Detected a syntax error in the alias instruction.

Action: Check the configuration i.e. the Object Descriptors.

ERR(STG108): Listconfig: Must be inside <object>.

Explanation: Listconfig out of scope, should be enclosed in a 'object' tag.

Action: Check the configuration i.e. the Object Descriptors.

ERR(STG109): Join: No ObjectDescriptor '*string1*'

Explanation: Configuration problem: Missing Object Descriptor.

Action: Check the configuration i.e. the Object Descriptors.

WAR(STG110): Property page '*string1*': insertAfter ignored, page replaced!

Explanation: Property page already exists and will be replaced rather than inserted.

Action: Check the configuration i.e. the Object Descriptors.

ERR(STG111): Couldn't read script '*string1*' (cause: *string2*).

Explanation: The requested script couldn't be retrieved.

Action: Check the configuration i.e. the Object Descriptors, add the script if missing.

ERR(STG112): Couldn't get list of xml-files from LDAP (cause: *string1*).

Explanation: Encountered a problem while reading Object Descriptions from the LDAP store.

Action: Check the reason described in the detailed message. Check server and network connections.

WAR(STG150): Script ignored. Mandatory property 'name' is missing!

Explanation: Report configuration: Detected a script where the mandatory 'name' attribute is missing.

Action: Check the report configuration.

ERR(STG201): LDAP search returned error: *string1* ReturnCode=*string2*

Explanation: LDAP search failed.

Action: Check the reason described in the detailed message. Check server and network connections.

WAR(STG202): Server-side sorting is not supported. Using client sorting to build a sorted list!

Explanation: Sorting will be done by the client. This might take some time depending on the number of entries.

Action: None.

WAR(STG203): *string1* access is currently unavailable: *string2*. Falling back to sequential read.

Explanation: As a last resort, client performs a sequential read. This might take some time depending on the number of entries.

Action: None.

WAR(STG204): Server-side sorting is unavailable for sort attributes '*string1*': *string2*. Using client sorting to build a sorted list!

Explanation: Sorting will be done by the client. This might take some time depending on the number of entries.

Action: None.

WAR(STG205): Can’t read 'supportedControl' from the LDAP root entry:_string1_. Falling back to sequential read.

Explanation: Client performs a sequential read. This might take some time depending on the number of entries.

Action: None.

WAR(STG206): Can’t read properties from the LDAP root entry:_string1_. Falling back to sequential read.

Explanation: Client performs a sequential read. This might take some time depending on the number of entries.

Action: None.

ERR(STG207): Unexpected error while performing client-side sorting. ObjectCollection.Entry.compareTo: *string1*.

Explanation: Internal error, list of entries not properly sorted.

Action: Check the reason described in the detailed message, Inform DirX Identity technical support about the problem.

WAR(STG208): SequentialRead: Operation 'nextElement' reported '*string1*'.

Explanation: Failed to retrieve the next element while performing a SequentialRead.

Action: Check the reason described in the detailed message. Check server and network connections.

WAR(STG209): SimplePaged: Unexpected error '*string1*'.

Explanation: Failed to perform a SimplePaged read operation.

Action: Check the reason described in the detailed message. Check server and network connections.

WAR(STG210): VirtualListView: Unexpected error '*string1*'.

Explanation: Failed to perform a VirtualList read operation.

Action: Check the reason described in the detailed message. Check server and network connections.

ERR(STG250): Parsing error: *string1*.

Explanation: Internal error, failed to parse XML content.

Action: Check the reason described in the detailed message.

ERR(STG270): ObjectEnumeration returned too much entries: *string1* should be *string2*.

Explanation: Internal Error, ObjectEnumeration ignored sizelimit.

Action: None.

INF(STG280): Collection *string1* started: '*string2*'.

Explanation: Start of import, export or delete collection task.

Action: N/A.

INF(STG281): Collection *string1* finished in *string2* sec: '*string3*'.

Explanation: End of export or delete collection task.

Action: N/A.

ERR(STG282): Collection export of subtree '*string1*' hit time limit, export is incomplete.

Explanation: During export of an collection subtree the time limit was hit.

Action: Increase time limit in LDAP logon profile.

ERR(STG283): Converting DSML filter to LDAP filter failed: *string1*.

Explanation: Error converting of DSML to LDAP filter during collection export using rule.

Action: Check DSML filters in collection rule.

ERR(STG284): Import of file '*string1*' failed: *string2*.

Explanation: Error importing LDIF file.

Action: Check file accessibility, syntax and LDAP server availability.

WAR(STG301): Clear text password is not supported in encryption mode.

Explanation: Only encrypted passwords are allowed.

Action: Check the configuration (encryption mode). Check configuration for passwords represented as clear text.

ERR(STG302): Scrambled password is not supported in encryption mode.

Explanation: Only encrypted passwords are allowed.

Action: Check the configuration (encryption mode). Check configuration for scrambled passwords.

WAR(STG303): Encrypted password is not supported in non-encryption mode.

Explanation: Cannot decrypt encrypted passwords if encryption is switched off.

Action: Check the configuration correct 'encryption mode'.

WAR(STG350): No match-rules for *string1*

Explanation: Match-rule (i.e. mappings) link LDAP entries with their associated Object Descriptors. If this information is missing, LDAP entries won't be presented properly.

Action: Check the configuration.

WAR(STG351): DependsOn: property '*string1*' not found (objectdescriptor:'*string2*')!

Explanation: DependsOn references an undefined property.

Action: Check the configuration, correct the Object Description.

WAR(STG401): Transaction.getDefault(): ambiguous result, there is more than one open transaction!

Explanation: Internal error: There is more than one open transaction!

Action: None.

WAR(STG450): LookupNode: Couldn't select '*string1*' due to timeout (cause *string2*).

Explanation: Internal error: Failed to lookup node for a given StorageObject.

Action: None.

ERR(STG451): Failed to load children (cause: *string1*).

Explanation: Detected an exception, failed to load child nodes.

Action: Check the reason described in the detailed message.

WAR(STG501): Script ignored. Mandatory property 'name' is missing!

Explanation: Report configuration: Detected a script where the mandatory 'name' attribute is missing.

Action: Check the report configuration.

ERR(STG502): ReportWriter Factory error: *string1*

Explanation: Failed to set up the ReportWriter.

Action: Check the report configuration, check the reason described in the detailed message.

ERR(STG503): ReportWriter reported a configuration error: *string1*

Explanation: The report isn't configured properly.

Action: Check the report configuration, check the reason described in the detailed message.

ERR(STG504): ReportWriter reported a transformation error: *string1*

Explanation: Encountered problems while transforming data.

Action: Check the report configuration, check the reason described in the detailed message.

ERR(STG550): Can't initialize DirX message service. TagScanner is not available.

Explanation: Effect: Text-editors will not work properly since features like syntax detection are not available.

Action: Check the installation.

ERR(STG551): Bind failed to LDAP server *string1*: *string2* for user DN=*string3*. TagScanner is not available.

Explanation: Effect: Text-editors will not work properly since features like syntax detection are not available.

Action: Check server and network connections.

ERR(STG552): Bind failed to LDAP server *string1*: *string2* for user DN=*string3*. TagScanner is not available.

Explanation: Effect: Text editors will not work properly since features like syntax detection are not available.

Action: Check certificate install_path/gui/conf/cert8.db.

ERR(STG553): Can't load DLLs required for ScanTags: *string1*. TagScanner is not available.

Explanation: Effect: Text-editors will not work properly since features like syntax detection are not available.

Action: Check the installation for missing DLLs.

ERR(STG555): createFrom: Must be inside <object>.

Explanation: createFrom out of scope, should be enclosed in a 'object' tag.

Action: Check the configuration i.e. the Object Descriptors.

ERR(STG601): No IDS-J server configured in connectivity database.

Explanation: The IDS-J server configuration cannot be read. Thus, host and port of the IDS-J server cannot be determined. As a consequence, no request workflow is started.

Action: Run the Configuration Wizard to configure the IDS-J server. Check LDAP server and network connection.

WAR(STG602): More than one IDS-J server with enabled Request-Workflows configured in connectivity database for domain '*string1*'.

Explanation: More than one IDS-J server with enabled Request-Workflows is present in the configuration database supporting given domain. The Request-Workflow service URL cannot be read.

Action: Check configuration of all IDS-J server objects. The domain property has to be unique. Only one server per domain should have enabled Request-Workflows(LDAP attribute dmxReqWorkflowEnabled set). Correct via Identity Manager in Expert View.

ERR(STG603): Cannot read IDS-J server from connectivity database: *string1*.

Explanation: The LDAP search for IDS-J server configuration returned error(s). Thus, no request workflow is started.

Action: Check detailed message and remove error(s). Check server and network connections.

ERR(STG604): No service configured for IDS-J server DN=*string1*.

Explanation: The link to the service configuration cannot be read from the IDS-J server configuration. As a consequence, no request workflow is started.

Action: Configure IDS-J server and its related service/system properly.

ERR(STG605): No service found in search with filter=*string2* under base=*string3* in DirX Identity confdb.

Explanation: The service referenced by the message service configuration's dxmService-DN is not found. Thus, host and port of the message service cannot be determined.

Action: Correct the dxmService-DN link of the message service configuration.

ERR(STG606): Server ports dxmDataPort/dxmSecurePort are missing or empty in the service related to the IDS-J server.

Explanation: The dxmAddress/dxmSecurePort attributes are empty. As a consequence, no request workflow is started.

Action: Edit the service related to the message server. In case you use SSL, enter the secure port to dxmSecurePort and set ssl true in the specific attributes. Otherwise, enter the port to dxmDataPort.

ERR(STG607): No system found in search with filter=*string2* under base=*string3* in DirX Identity confdb.

Explanation: The system referenced by the service configuration's dxmSystem-DN is not found. Thus, host of the IDS-J service cannot be determined. No request workflow is started.

Action: Correct the dxmSystem-DN link of the service configuration.

ERR(STG608): Searching for attributes *string1* with filter=*string2* under base=*string3* on DirX Identity connectivity server failed: *string4*.

Explanation: Reading of attributes from the DirX Identity connectivity DB fails. Thus, the IDS-J server configuration is incomplete and no request workflow is started.

Action: Check the detailed message. Check for server and network connection.

ERR(STG609): IDS-J server host can neither be read from the system nor from the service.

Explanation: The dxmAddress attribute is empty, or the related configuration object cannot be read. As a consequence, no request workflow is started.

Action: Edit the system related to the IDS-J server. Enter the hostname in dxmAddress.

ERR(STG610): The system is not referenced by the IDS-J server configuration.

Explanation: The system link is not set at the IDS-J service. Therefore, the IDS-J server host cannot be read an no request workflows are started.

Action: Add a system to the IDS-J service. Enter the hostname in dxmAddress.

WAR(STG611): A temporary error occurred while trying to read IDS-J server SOAP configuration: *string1*. Retrying.

Explanation: Reading the IDS-J server configuration ran on a temporary error. After waiting 20s, this is retried, until the retry limit is exceeded.

Action: Check LDAP server and network connections.

WAR(STG612): LDAPException when searching with base = '*string1*' and filter = '*string2*': *string3*.

Explanation: The NormalPageEnumerator performed a normal LDAP search with given parameters. It tries again after re-connect.

Action: Check LDAP server and network connections. Check the search base and filter.

INF(STG613): Switching to paged read after reaching size limit '*string3*' when searching with base = '*string1*' and filter = '*string2*'.

Explanation: The NormalPageEnumerator performed a normal LDAP search and hit the size limit. It tries again using paged search.

Action: None.

WAR(STG614): LDAPException when reading next Entry from search result after searching with base = '*string1*' and filter = '*string2*': *string3*.

Explanation: The NormalPageEnumerator iterates through the search result after a normal LDAP search with given parameters. It ignores this entry.

Action: Check LDAP server and network connections. Check the entries covered by the search base and filter.

WAR(STG615): Specific attribute '*string1*' is single value. Extra value '*string2*' removed.

Explanation: A specific attribute defined as single value has more than one value. The additional values are removed.

Action: Correct the attribute value. Check and correct the source of multiple values.

WAR(STG616): Search for '*string1*' returned 'null' (Object doesn't exist or any other LDAP problems).

Explanation: Reading the given object returns a null result.

Action: Check existence of given DN in LDAP directory or check for LDAP errors.

WAR(STG617): Multiple values exist for single value property '*string1*' of DN=*string2*.

Explanation: Multiple values exist for an Attribute being declared single values in the object description. Thus, only one of those values is considered, leading to unexpected results e.g. in access policy evaluation.

Action: Correct object description or property value.

WAR(STG618): Retrying (*string1*.) to read object with search base *string2* and filter *string3*, last exception was *string4*.

Explanation: The calling component was unable to read described information about active message server. Retrying.

Action: If the issue continues check if your LDAP server for connectivity configuration is running and if e. g. no firewall blocks any connection.

ERR(STG619): Could not read object with search base *string1* and filter *string2* due to error *string3*.

Explanation: The calling component was unable to read described information about active message server due to an error.

Action: If the issue continues check if your LDAP server for connectivity configuration is running and if e. g. no firewall blocks any connection.

ERR(STG620): More configuration objects with search base *string1* and filter *string2* was found.

Explanation: The calling component was unable to read unique information about active message server.

Action: If the issue continues check if your JMS configuration is correctly configured, check activated service, host, port and SSL property of the JMS service.

ERR(STG621): No valid configuration object with search base *string1* and filter *string2* was found.

Explanation: The calling component was unable to read described information about active message server. No or multiple entries was found.

Action: If the issue continues check if your JMS configuration is correctly configured, check activated service, host, port and SSL property of the JMS service.

ERR(STG622): Could not use given format string *string1* as a valid input for java.text.SimpleDateFormat object due to the following error *string2*. Using default format (yyMMddHHmmss).

Explanation: A report definition is using invalid property timestampFormat which is used for creation of timestamp string using object java.text.SimpleDateFormat.

Action: Correct faulty configuration for timestampFormat containing the format string (see documentation for java.text.SimpleDateFormat) or delete it and use only default format instead.

ERR(STG623): Error while reading LDAP schema: *string1*.

Explanation: LDAP schema (namely the attribute types) could not be read successfully.

Action: Check the LDAP error message.

INF(STG650): Missing closing '*string1*' in expression '*string2*'. Replacement is not done.

Explanation: Expression can’t be evaluated due to a missing ")". Replacement is not done"

Action: Correct the expression.

16.2.8. Logging Messages (SVCMGMT)

This topic describes the logging messages returned by the Service management user hook for the DirX Identity Connector Framework.

INF(SVCMGMT001): Didn’t create RequestWorkflow request for Entry with ID='*string1*' because no modifications or attributes are present or relevant.

Explanation: It doesn’t make sense to create a workflow request for creating or modifying a subject without relevant or present attributes.

Action: Check the event and the mapping why no attributes are present.

INF(SVCMGMT002): Didn’t create RequestWorkflow request for Entry with ID='*string1*' because no event is present.

> **Explanation**: It is impossible to trigger proper RequestWorlflow request without the information about new or changed attributes.
>
> **Action**: Check the configuration of the workflow.

INF(SVCMGMT003): The event for Entry with ID='*string1*' contains an attribute *string2* that is not mapped for provisioning. Ignoring this attribute.

> **Explanation**: The described attribute is contained in the event but was not mapped for provisioning in the workflow configuration.
>
> **Action**: If you intend to provision this attribute extend the mapping list of the channel.

INF(SVCMGMT004): The modify request for Entry with ID='*string1*' was transformed to delete request.

> **Explanation**: The modify event contained the delete change of state driven by provisioning service.
>
> **Action**: None.

WAR(SVCMGMT101): Didn’t create RequestWorkflow request for Entry with ID='*string1*' because of invalid event type: *string2*.

> **Explanation**: It is impossible to trigger proper RequestWorlflow without the correct event type. Only add, modify or delete events are supported.
>
> **Action**: Contact the DirX Identity support.

WAR(SVCMGMT102): Could not find any mapped attributes for Entry with ID='*string1*'.

> **Explanation**: The mapped attributes along with the event are required to build proper request for RequestWorlflow.
>
> **Action**: Check the configuration of the mapped attributes for the relevant connectivity workflow.

WAR(SVCMGMT103): Could not read the object type from the source channel environment, using default value '*string1*'.

> **Explanation**: The RequestWorkflow requests need the information about object type set in the operational attributes.
>
> **Action**: Check the configuration of the source channel, the object type information must exist.

16.3. Logging Messages for the Connectivity Components

AAD... Logging messages for the agent clients using the ADSI interface (ADS agent, Exchange Agent).

ADS... Logging messages for the ADS connector.

AEP... Logging messages for the SAP EP UM agent/connector.

ALN... Logging messages for the IBM Notes agent.

ANF... Logging messages for the Notification agent.

ASC... Logging messages for the SAP ECC UM agent / connector using SAP JCo v3.

ASI... Logging messages for the SiPass agent.

C1_... Logging messages for the sample connector extension.

CSP... Logging messages for the Customer SharePoint Userhook for DirX Identity Join Engine.

EEC... Logging messages for the Evidian Esso Connector.

GAC... Logging messages for the Google Apps Connector.

JDB... Logging messages for the JDBC agent / connector.

JMS... Logging messages for the JMS connector.

LCO... Logging messages for the LDAP connector.

O365... Logging messages for the Office 365 connector.

ODB... Logging messages for the ODBC Agent.

OICF... Logging messages for the OpenICF connector.

PVC... Logging messages for the DirX Identity link checker agent.

RACF... Logging messages for the RACF connector.

RWC... Logging messages for the Request Workflow connector.

SAMP... Logging messages for the sample connector.

SIPORT... Logging messages for the SIPORT connector filter.

SLSF... Logging messages for the Salesforce connector.

SPCO... Logging messages for the Sharepoint Connector.

SPUH... Logging Messages for the Sharepoint Userhook.

SPV1V2... Logging Messages for the SMPLV1-toV2 connector

16.3.1. Logging Messages (AAD)

This topic describes the logging messages returned by the DirX Identity agent clients that access connected directories via the ADSI interface (ADS agent, Exchange agent). If an import operation to Active Directory or Exchange is performed, the data record that caused the log message from the ADSI interface is always written in front of the logging message.

AAD0001: ADSI/LDAP error.

Explanation: This error message appears if an ADSI interface function returns an error. The name of the interface function and its error code and error text is logged.

Action: Check the ADSI/LDAP error code and error text, which is written right behind this error message.

AAD0002: Configuration file could not be found.

Explanation: This message indicates that the agent client cannot find the configuration file under the specified name or path.

Action: Check the path of the configuration file in the command line. If the configuration file is specified as a relative path name and it is expected to be in the same directory as the agent executable, it must be specified as follows: "*.**ConfigurationFile*".

AAD0003: Log file could not be opened. Cause=*cause*.

Explanation: This message indicates that the agent client cannot open the log file under the specified name or path. The error code for the failure is shown in *cause*.

Action: Check the path of the log file in the command line and the access rights the user running the agent has on your computer, for example the access right to create files.

AAD0004: No SearchBase specified.

Explanation: This message indicates that a value for the SearchBase field has not been specified in the configuration file.

Action: Check the configuration file.

AAD0005: Delta export is selected, but no USN-Number can be found.

Explanation: This message indicates that a value for the USN-Number field has not been specified in the configuration file.

Action: Check the configuration file.

AAD0006: Delta export is selected, but USN-Changed>= cannot be found in LdapFilter.

Explanation: This message indicates that the agent client cannot find the USN-Changed>= field in the LdapFilter field. In version 1.00, USN-Changed>= must be contained in the LdapFilter field.

Action: Check the configuration file.

AAD0007: Delta Export is selected, but USN-Changed>= has a syntax error in LdapFilter.

Explanation: This message indicates that the USN-Changed>= field has a syntax error in the LdapFilter field.

Action: Check the configuration file.

AAD0008: In Active Directory, it must be uSNChanged, not USN-Changed in LdapFilter.

Explanation: This message indicates that there is a syntax error in the LdapFilter field.

Action: Check the configuration file.

AAD0009: Delta export is selected, but uSNChanged>= has a syntax error in LdapFilter.

Explanation: This message indicates that there is a syntax error in the LdapFilter field.

Action: Check the configuration file.

AAD0010: Export file could not be created. Cause=*error_code*

Explanation: This message indicates that the agent client cannot create the export file under the specified name or path. The error code for the failure is given in Cause.

Action: Check the path of the export file in the command line.

AAD0011: Number of selected attributes is too large.

Explanation: The number of attributes to be selected for export is currently limited to 200.

Action: Check the selected attributes in the configuration file.

AAD0012: If you search for Is-Deleted, you must append **cn=admin** to UserName.

Explanation: If you want to search for deleted objects in Exchange, the UserName field in the configuration file must be appended with **cn=admin**.

Action: Check the configuration file.

AAD0013: There is no rootDSE for getting the well-known deleted objects container.

Explanation: Binding to the rootDSE interface failed, so the name of the deleted objects container cannot be obtained.

Action: Check the network connection.

AAD0014: Cannot write USNChangedMax to Optionfile. Error code: *error_code*.

Explanation: This message indicates that the function WritePrivateProfileString has failed to write the highest USNChanged for Exchange to the configuration file.

Action: Check the rights that the user who is running the agent has on your computer, for example the right to modify files.

AAD0015: Cannot write highestCommittedUSN to Optionfile. Error code: *error_code*.

Explanation: This message indicates that the function WritePrivateProfileString has failed to write the highest CommittedUSN for Active Directory to the configuration file.

Action: Check the access rights that the user who is running the agent has on your computer, for example the right to modify files.

AAD0016: Cannot write LdapFilter to Optionfile. Error code: *error_code*.

Explanation: This message indicates that the function WritePrivateProfileString has failed to write the LDAP filter that was taken for the last search to the configuration file.

Action: Check the rights that the user who is running the agent has on your computer, for example the right to modify files.

AAD0017: GetNTAccountName failed. Cannot allocate memory.

Explanation: This message indicates that the agent client cannot allocate the memory need to perform the operation.

Action: Check the available memory on your system.

AAD0018: LookupAccountSid failed. Error code: *error_code*.

Explanation: This message indicates that the Windows security function LookupAccountSid for getting the SID representation for the account name has failed.

Action: Check the error code and whether the account that started the agent has appropriate access rights in its local domain to use security functions.

AAD0019: Return code of CKey::DecryptPW: *error_code*.

Explanation: This message indicates that the CKey::DecryptPW function has failed.

Action: Check the error code and the consistency of your security environment.

AAD0020: dxm_descramble_frombase64 returned error: *error_code*.

Explanation: This message indicates that the dxm_descramble_frombase64 function to unscramble the user password in the configuration file has failed.

Action: Check the error code and the consistency of your security environment.

AAD0021: TraceLevel is used, but no TraceFileName is specified. No trace file is written.

Explanation: This warning is given if a trace level is configured but no trace file name is specified. No trace file is written but the code executes.

Action: Check the configuration file.

AAD0022: Object CTrace cannot be created.

Explanation: This message indicates that the construction of a new CTrace object, which the agent client uses to write trace messages to the specified trace file, has failed.

Action: Check the path of the trace file specified and the rights that the user who is running the agent has on your computer, for example the right to create files.

AAD0023: CoInitialize failed.

Explanation: This message indicates that initialization of the COM library has failed. This error points to a basic problem in your environment.

Action: Check your installation.

AAD0024: OleInitialize failed.

Explanation: This message indicates that the initialization of OLE has failed. This error points to a basic problem in your environment.

Action: Check your installation.

AAD0025: Cannot open import file. Cause=*cause*.

Explanation: This message indicates that the agent client cannot open the import data file. The reason for this failure is given in Cause.

Action: Check the path of the import file specified in the command line.

AAD0026: Memory for read buffer could not be allocated.

Explanation: This message indicates that memory allocation has failed.

Action: Check the available memory on your system.

AAD0027: Memory for address buffer could not be allocated.

Explanation: This message indicates that memory allocation has failed.

Action: Check the available memory on your system.

AAD0028: realloc for a large data record of *xx* bytes failed.

Explanation: This message indicates that memory allocation has failed.

Action: Check the available memory on your system.

AAD0029: Memory buffer for ClearTextAddress could not be allocated.

Explanation: This message indicates that memory allocation has failed.

Action: Check the available memory on your system.

AAD0030: Return code of CKey::ChangeAddress(): *xx*.

Explanation: This message indicates that the function CKey::ChangeAddress has failed.

Action: Check the error code and the consistency of your security environment.

AAD0031: ImportAddresses failed. Cause=*cause*.

Explanation: This message indicates that the function ImportAddresses has failed.

Action: Check the Cause.

AAD0032: rdn is ignored. It cannot be modified/added.

Explanation: The rdn attribute in the data file is ignored because it is not allowed to be added or modified.

Action: Remove it or leave it in the data record. It is ignored.

AAD0033: Cannot find attribute type of *AttributeName*. Attribute ignored.

Explanation: This message is returned when an attribute type has not been configured in the configuration file for an attribute *AttributeName*. As a result, the attribute cannot be imported and is ignored.

Action: Specify the type of the attribute in the import configuration file.

AAD0034: AdsPathOld and ldapFilter are empty; changetype: move.

Explanation: To perform a move, either AdsPathOld or ldapFilter must exist as the source.

Action: Specify either AdsPathOld or ldapFilter in the data record.

AAD0035: Search with ldapFilter failed. *Cause*

Explanation: Instead of an AdsPath, an ldapFilter can be specified in the data record to identify the object. The search for the object that needs to be modified, moved or deleted failed because of *Cause*.

Action: Check the *Cause*.

AAD0036: AdsPath and ldapFilter are empty; changetype: modify.

Explanation: To perform a modify, either AdsPath or ldapFilter must exist in the data record.

Action: Specify either AdsPath or ldapFilter in the data record.

AAD0037: AdsPath and ldapFilter are empty; changetype: delete.

Explanation: To perform a delete, either AdsPath or ldapFilter must exist in the data record.

Action: Specify either AdsPath or ldapFilter in the data record.

AAD0038: AdsPath is empty; changetype: move.

Explanation: To perform a move, an AdsPath as the destination of the move operation must exist in the data record.

Action: Specify an AdsPath in the data record.

AAD0039: AdsPath, ldapFilter and changetype are empty.

Explanation: To add, modify or move an object, at least AdsPath or ldapFilter must exist in the data record.

Action: Specify one of these items in the data record.

AAD0040: AdsPath syntax is incorrect.

Explanation: This message indicates that the syntax of AdsPath in the data record is incorrect.

Action: Check AdsPath.

AAD0041: AdsPath syntax is incorrect. Rdn is empty.

Explanation: This message indicates that the syntax of AdsPath in the data record is incorrect because the rdn is empty.

Action: Check AdsPath.

AAD0042: AdsPath contains a character that is not allowed.

Explanation: This message indicates that the syntax of AdsPath in the data record is incorrect because contains characters in the rdn part that are in the RejectedCharacters list.

Action: Check the AdsPath.

AAD0043: AdsPath syntax is incorrect. Container is empty.

Explanation: This message indicates that the syntax of AdsPath in the data record is incorrect because the container is empty.

Action: Check the AdsPath.

AAD0044: Assigning/creating NT account failed. *Cause*.

Explanation: This message indicates that assigning or creating the Windows NT account specified in the data record for an Exchange mailbox has failed due to *Cause*.

Action: Check the *Cause*.

AAD0045: Obtaining the rdn from the GUID representation of object *AdsPath* failed.

Explanation: If the GUID is specified in AdsPath, the rdn of the object must be obtained to perform the desired operation. This action has failed.

Action: Check AdsPath and the GUID.

AAD0046: Deleting NT account failed. *Cause*

Explanation: This message indicates that deleting the Windows NT account specified in the data record for an Exchange mailbox has failed due to *Cause*.

Action: Check the *Cause*.

AAD0047: Building binary string failed. Cannot allocate memory.

Explanation: This message indicates that a binary string cannot be built because of a memory allocation failure.

Action: Check your system memory.

AAD0048: userCertificate larger than expected.

Explanation: This message indicates that the attribute userCertificate in the import data record is larger than expected.

Action: Check if it is a valid userCertificate.

AAD0049: The function ocl_frombase64 to convert from base64 to binary failed.

Explanation: This message indicates that an attribute of type octet in the import data file is expected to be base64-encoded and cannot be decoded to a binary string.

Action: Check for valid base64 encoded attributes of type octet.

AAD0050: Allocating memory from the stack for a binary string failed.

Explanation: For an attribute in the import data file of type octet, no memory from the stack can be allocated.

Action: Check the attributes of your data record.

16.3.2. Logging Messages (ADS)

This topic describes the logging messages returned by the Ads Connector for DirX Identity Connector Framework.

INF(ADS201): Add request contains mailbox enabling attribute msExchHomeServerName. Create a modify request with the mailbox security descriptor.

> **Explanation**: After successful add a modify request will be submitted with the MB security descriptor attribute set.
>
> **Action**: None.

INF(ADS202): Modify request setting the mailbox security descriptor succeeded.

> **Explanation**: Setting the MB security descriptor attribute after adding the user

succeeded.

Action: None.

INF(ADS203): Modify request contains mailbox enabling attribute msExchHomeServerName. Extend the modify request with the mailbox SD.

Explanation: The modify request will be extended with the MB security descriptor attribute.

Action: None.

INF(ADS204): Modify request contains attribute unicodePwd.

Explanation: The modify request will be corrected to contain the unicodePwd in the right form.

Action: None.

INF(ADS206): Delta cookie returned by the server: *string1*.

Explanation: This cookie which is returned in the SPML search response can be used for the next delta search.

Action: None.

INF(ADS207): Performed a kerberos authentication successfully.

Explanation: A kerberos authentication on the LDAP connection was performed.

Action: None.

INF(ADS208): Callback handler from kerberos authentication was called. *string1*.

Explanation: Windows server called callback handler passed in kerberos authentication.

Action: None.

INF(ADS209): Range search for attribute *string1* returned no values.

Explanation: Range search for the above multi value attribute returned no values.

Action: None.

INF(ADS210): Add request contains Exchange 2007 attribute msExchRecipientTypeDetails. Create a modify request with the mailbox guid.

Explanation: After successful add a modify request will be submitted with the MB guid attribute set.

Action: None.

INF(ADS211): Modify request setting the mailbox guid succeeded.

Explanation: Setting the mailbox guid attribute after adding the user succeeded.

Action: None.

INF(ADS212): Modify request contains Exchange 2007 attribute msExchRecipientTypeDetails. Read user to see if mailbox guid already exists.

Explanation: If mailbox guid does not exist yet, the modify request will be extended with the mailbox guid attribute.

Action: None.

INF(ADS213): The user does not have a mailbox guid yet. Extend modify request with generated mailbox guid.

Explanation: The modify request will be extended with the mailbox guid attribute.

Action: None.

INF(ADS214): The user already has a mailbox guid. Do not overwrite it. Modify request is not extended by msExchMailboxGuid.

Explanation: The modify request will not be extended with the mailbox guid attribute.

Action: None.

WAR(ADS400): Cannot access version information from jar file.

Explanation: Accessing version information from manifest failed.

Action: Inform DirX Identity technical support about the problem.

WAR(ADS403): The LDAP DirSync control contains no cookie.

Explanation: Since no delta cookie exists the next search will be a full search.

Action: Do a full search or inform DirX Identity technical support about the problem.

WAR(ADS405): Range search with base: *string1* for attribute: *string2* failed. Probably no values in this range. LDAP exception msg: *string3*.

Explanation: Range search for large multi value attributes failed. Probably there are just no values in this range.

Action: Check the LDAP exception message.

ERR(ADS500): Identifier missing in request.

Explanation: No identifier was specified in the request.

Action: Specify a DN in the identifier section of the SPML request.

ERR(ADS501): Modify request setting the mailbox security descriptor failed. Custom error message: *string1*, error message: *string2*.

Explanation: Setting the MB security descriptor attribute after adding the user failed.

Action: Check the error messages.

ERR(ADS502): Delta search failed. Loop finished at that point. Error response message: *string1*.

Explanation: Search operation with the DirSync Control failed. Cookie of last successful search is returned in SPML response.

Action: Check the error messages.

ERR(ADS503): Search result contained no LDAP controls. Processing of delta search aborted.

Explanation: Since no LDAP controls are returned by the server the delta search must be aborted.

Action: Do a full search or inform DirX Identity technical support about the problem.

ERR(ADS504): Search result contained no LDAP DirSync control. Processing of delta search aborted.

Explanation: Since no LDAP DirSync control is returned by the server the delta search must be aborted.

Action: Do a full search or inform DirX Identity technical support about the problem.

ERR(ADS505): Delta paged search with base: *string1*, scope: *string2* and filter: *string3* failed. Error msg: *string4*.

Explanation: Delta paged search failed.

Action: Check the LDAP error code.

ERR(ADS506): Kerberos authentication failed. Error msg: *string1*.

Explanation: Trying a kerberos authentication on the LDAP connection failed.

Action: Check the LDAP error code.

ERR(ADS507): Creation of kerberos SaslClient failed. Error msg: *string1*.

Explanation: Creation of kerberos SaslClient on the LDAP connection failed.

Action: Check the error code.

ERR(ADS508): Modify request setting the mailbox guid failed. Custom error message: *string1*, error message: *string2*.

Explanation: Setting the msExchMailboxGuid attribute after adding the user failed.

Action: Check the error messages.

FTL(ADS900): Exception while accessing version information from jar file.

Explanation: Accessing version information from manifest failed.

Action: Inform DirX Identity technical support about the problem.

FTL(ADS901): An unsupported encoding exception occurred with exception message: *string1*.

Explanation: An exception occurred with the given exception message.

Action: Check the exception message text.

FTL(ADS933): Exception occurred with exception message: *string1*.

Explanation: An exception occurred with the given exception message.

Action: Check the exception message text.

16.3.3. Logging Messages (AEP)

This topic describes the logging messages returned by the DirX Identity SAP EP UM agent/connector.

FTL(AEP0001): Exception while trying to read the version string from the jar file.

Explanation: Cannot read the version string from the jar file.

Action: Inform DirX Identity technical support about the problem.

ERR(AEP0100): Error in processing AddRequest: *string1*

Explanation: An Exception occurred when sending a request to the SOAP service.

Action: Check the details. Probably the response cannot be evaluated: is it well-formed? Does the responded SOAP version match that of the request?

WAR(AEP0300): Cannot convert string "*string1*" to integer. Configuration parameter "*string2*" ignored.

Explanation: See above.

Action: Correct the configuration parameter.

WAR(AEP0301): Cannot convert value "*string1*" for attribute "*string2*" from Unicode to UTF-8.

Explanation: The given value cannot be converted to UTF-8.

Action: Correct the value.

WAR(AEP0302): Attribute "*string1*" is multi-valued; attribute is ignored.

Explanation: Only single value is allowed.

Action: Correct the value.

WAR(AEP0303): Attribute "*string1*" has wrong value "*string2*".

Explanation: Only true or false are allowed.

Action: Correct the value.

WAR(AEP0304): Cannot set productive password in target system: *string1*.

Explanation: User is created but with an internally generated initial password.

Action: Action is depending on the provided system error message. Best is to reset the

password.

INF(AEP0500): *** Connector '*string1*', version '*string2*', started at *string3* (log level *string4*) ***

Explanation: Displays connector version and other information.

Action: None.

16.3.4. Logging Messages (ALN)

This topic describes the logging messages returned by the DirX Identity Lotus Notes Agent.

ALN0001: ChangeType of entry is add, but the entry is already available.

Explanation: The changetype of the entry is "add", but the entry is already available.

Action: Change the attribute **ChangeType** in the import data file.

ALN0002: ChangeType of entry is modify, but the entry is not available.

Explanation: The changetype of the entry is "modify", but the entry is not available.

Action: Change the attribute **ChangeType** in the import data file.

ALN0003: ChangeType of entry is delete, but the entry is not available.

Explanation: The changetype of the entry is "delete", but the entry is not available.

Action: Change the attribute **ChangeType** in the import data file or delete the entry in the import data file.

ALN0004: ChangeType of entry is not modify, but you have used operation code for attributes.

Explanation: The changetype of the entry is not "modify", but you have used the operation code for attributes.

Action: Change the attribute **ChangeType** in the import data file.

ALN0005: Data in import file are not contained in form: "Person".

Explanation: The entry in the import data file does not have the same form as it does in the configuration file.

Action: Check the form name in both files.

ALN0006: The attribute "Type" is not in import data file.

Explanation: The attribute "Type" is not in the entry in the import data.

Action: Check the attribute **Type** in the import data file.

ALN0007: The field Members contains too many characters (maximum is 32500).

Explanation: The maximum length of the attribute is 32500 characters.

Action: Check the attribute **Members** in the import data file.

ALN0008: The passwords are not encrypted in secure mode.

Explanation: The passwords are not encrypted in secure mode.

Action: Check the password handling.

ALN0009: The item is not contained in the document or the item is not of type Text, Number or DateTime.

Explanation: The item in the import data file is not contained in the form of the Notes document or it has another type.

Action: Check the attributes and attribute types of the Notes document. Use the field **TraceItemTypes** in the import configuration file to trace attributes type of the Notes document.

ALN0010: The mail file already exists with the user *user*.

Explanation: A mail file with this name already exists in the Notes system.

Action: Change the name for the mail file.

ALN0011: The user was not registered.

Explanation: The registration of the user failed because the mail file already exists.

Action: Change the name for the mail file.

ALN0012: Item is not multi-valued.

Explanation: This item is not a multi-valued attribute.

Action: Check the value of this attribute.

ALN0013: Cannot find the document: *document*

Explanation: The agent cannot find the document in the Notes database.

Action: Check the data file and the Notes database.

ALN0014: Found more than one document: *document*

Explanation: The agent has located more than once instance of the document.

Action: Check the data file and the Notes database.

ALN0015: Cannot open the view.

Explanation: The agent cannot open the view.

Action: Check the key **ViewFolder** in the import configuration file.

ALN0016: Cannot find the view.

Explanation: The agent cannot find the view.

Action: Check the key **ViewFolder** in the import configuration file.

ALN0017: The configuration file is not available.

Explanation: The configuration file is not available.

Action: Check the command line and the working directory.

16.3.5. Logging Messages (ANF)

This topic describes the logging messages returned by the DirX Identity notification agent.

ANF0001: Wrong command line, usage: "java siemens.dxm.notify.sendmail *xmlfile*."

Explanation: Incorrect command line syntax.

Action: Check the command line.

ANF0002: The Type field is unavailable.

Explanation: The **Type** field is unavailable in the configuration file.

Action: Check the configuration file.

ANF0003: The Host field is unavailable.

Explanation: The **Host** field is unavailable in the configuration file.

Action: Check the configuration file.

ANF0004: The From field is unavailable.

Explanation: The **From** field is unavailable in the configuration file.

Action: Check the configuration file.

ANF0005: The To field is unavailable.

Explanation: The **To** field is unavailable in the configuration file.

Action: Check the configuration file.

ANF0006: The Subject field is unavailable.

Explanation: The **Subject** field is unavailable in the configuration file.

Action: Check the configuration file.

ANF0007: The Type field has the wrong value.

Explanation: The value of the **Type** field can only be **email** or **SMS**.

Action: Check the configuration file.

ANF0008: SMS message has more than 160 characters.

Explanation: The maximum of characters in an SMS message is 160.

Action: Check the number of characters in the **Subject** field and in the **Body** field.

16.3.6. Logging Messages (ASC)

This topic describes the logging messages returned by the DirX Identity SAP R3 UM agent/connector (using SAP JCo v3).

FTL(ASC001): *string1*: '*string2*' (id: *string3*, parameter: *string4*, row: *string5*, field: *string6*, system: *string7*, user: *string8*) [*string9*]

Explanation: The R/3 system sent a fatal error message.

Action: Read the error message and perform appropriate actions.

FTL(ASC002): Feature "*string1*" not implemented yet.

Explanation: The specified feature is not yet implemented.

Action: None.

FTL(ASC003): Type "*string1*" not supported in table access.

Explanation: Fields of the specified type cannot be accessed.

Action: Choose other fields.

FTL(ASC004): Table "*string1*" field "*string2*" has unexpected length "*string3*". The agent needs to be updated.

Explanation: The specified field has an unexpected length.

Action: The agent needs to be updated. The schema in R/3 has changed.

FTL(ASC005): Table "*string1*" has no structure field "*string2*". The agent needs to be updated.

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC006): The configuration is incorrect. No role specified.

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC007): Incorrect role "*string1*" specified. Expected is "connector".

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC008): No connector name specified.

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC009): Incorrect connector name "*string1*" specified. Expected is "*string2*".

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC010): No connector version specified.

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC011): Incorrect connector version "*string1*" specified. Expected is "*string2*".

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC012): Connection type not specified.

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC013): SAP logon user name for connection "SAP_R3_UM" not specified.

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC014): SAP logon password for connection "SAP_R3_UM" not specified.

Explanation: The configuration is incorrect. A password is needed unless an SNC

connection is used.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC015): Unscrambling "*string1*" failed.

Explanation: The password might be empty or strongly encrypted.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC016): Host name of application server or mshost (host name of message server) of connection "SAP_R3_UM" not specified.

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC017): SAP logon client of connection "SAP_R3_UM" not specified.

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC018): SAP System ID (=system number) of connection "SAP_R3_UM" not specified.

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC019): Partner R/3 release "*string1*" not supported. Must be 46C or higher.

Explanation: The agent does not support the specified version.

Action: Install one of the supported versions of SAP R/3.

FTL(ASC020): Expected connection type "*string1*" or "*string2*" not found.

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC021): Exception while trying to read the version string from the jar file.

Explanation: Cannot read the version string from the jar file.

Action: Inform DirX Identity technical support about the problem.

FTL(ASC022): Cannot connect. Not all necessary parameters are provided (ASHOST or MSHOST: "*string1*", GWHOST: "*string2*").

Explanation: Not all necessary connection parameters are specified.

Action: Correct the connection parameters.

FTL(ASC023): Cannot connect. Not all necessary SNC parameters are provided (snc_partnername: "*string1*", snc_lib: "*string2*").

Explanation: Not all necessary connection parameters are specified.

Action: Correct the connection parameters.

FTL(ASC024): Incorrect logonVariant "*string1*" specified.

Explanation: The configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

FTL(ASC025): Internal error: JCo connection pool could not be opened.

Explanation: The JCO connection pool could not be acquired.

Action: Inform DirX Identity technical support about the problem.

FTL(ASC026): Internal error: type *string1* lasttab *string2* tab.length *string3* mismatch.

Explanation: Internal tables are not correct.

Action: Inform DirX Identity technical support about the problem.

FTL(ASC027): Internal error: method *string1* was called and open() was not finished successfully.

Explanation: The order of method calls is incorrect.

Action: Inform DirX Identity technical support about the problem.

FTL(ASC028): Option "setProductivePwdAtModDirectly" can only be activated if SNC mode is set.

Explanation: Incorrect settings are specified.

Action: Correct the connection parameters.

FTL(ASC050): Exception in constructor of class Configuration occurred: *string1*

Explanation: The filter configuration could not be initialized correctly.

Action: Check the exception message and/or inform DirX Identity technical support about the problem.

ERR(ASC100): *string1*: '*string2*' (id: *string3*, parameter: *string4*, row: *string5*, field: *string6*, system: *string7*, user: *string8*) [*string9*]

Explanation: The R/3 system sent an error message.

Action: Read the error message and perform appropriate actions.

ERR(ASC101): Unexpected message type '*string1*'.

Explanation: The agent received an unexpected message type in the response from the R/3 system.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC126): Initializing JCo pool failed: *string1*.

Explanation: Initializing the connection pool failed.

Action: Read the error message and perform appropriate actions.

ERR(ASC127): Incorrect identifier type "*string1*" specified. Expected is "*string2*".

Explanation: Configuration is incorrect.

Action: Change the configuration or inform DirX Identity technical support about the problem.

ERR(ASC128): Cannot convert value "*string1*" for attribute "*string2*" from Unicode to UTF-8.

Explanation: The specified value cannot be converted to UTF-8.

Action: Check the value and correct it.

ERR(ASC129): General JCo exception "*string1*" received.

Explanation: The JCo middleware sent an exception.

Action: Read the error message and perform appropriate actions.

ERR(ASC130): Requested field "*string1*" is undefined or not supported.

Explanation: The request is incorrect.

Action: Change the selected attributes.

ERR(ASC131): The requested structure/table/field "*string1*" is undefined or not supported.

Explanation: The request is incorrect.

Action: Change the selected attributes.

ERR(ASC132): At least one of the mandatory attributes to add an user is missing. Mandatory attributes are object identifier (valid R/3 user name), PASSWORD.BAPIPWD, and ADDRESS.LASTNAME.

Explanation: Mandatory attributes are object identifier (valid R/3 user name), PASSWORD.BAPIPWD, and ADDRESS.LASTNAME.

Action: Add the missing attribute.

ERR(ASC133): Could not find function "*string1*" in the repository.

Explanation: The specified function could not be found in the R/3 repository.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC134): Unsupported operation.

Explanation: The operation is not supported.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC135): Initializing JCo client failed: *string1*.

Explanation: The JCo middleware could not be initialized.

Action: Read the error message and check the JCo documentation.

ERR(ASC136): Mandatory attribute "*string1*" is missing.

Explanation: The given mandatory attribute is missing.

Action: Specify the missing mandatory attribute.

ERR(ASC137): Multi-values for the single-value attribute are specified.

Explanation: Multiple values for single-value attributes cannot be specified.

Action: Check why several values are specified.

ERR(ASC138): Insert of table row # *string1* in table "*string2*" failed: Errorflag: "*string3*".

Explanation: Insert into the table failed for the given row.

Action: See the error flag and check the R/3 documentation.

ERR(ASC139): Adding table data failed. See log file for more information.

Explanation: Data cannot be added to the table.

Action: See further messages in the error log and perform appropriate actions.

ERR(ASC140): Mismatch of specified values (# *string1*) for operation "*string2*"of attribute "*string3*".

Explanation: The specified operation cannot be performed for the specified attribute.

Action: Correct the input data.

ERR(ASC141): Internal error: function addValue: "*string1*" not in hash map.

Explanation: An internal error occurred.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC142): Internal error: unknown modification operation type "*string1*".

Explanation: An internal error occurred.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC143): No operation type specified in modifyRequest for attribute "*string1*".

Explanation: The request is incorrect.

Action: Change the request or inform DirX Identity technical support about the problem.

ERR(ASC144): Internal error: function get/addValue: index "*string1*" not in hash map for structure "*string2*".

Explanation: An internal error occurred.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC145): Internal error: function addValueWithVector: structure"*string1*" different sizes "*string2*" and "*string3*".

Explanation: An internal error occurred.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC146): Internal error: function addValue: exception: "*string1*" for structure "*string2*".

Explanation: An internal error occurred.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC147): Requested attribute "*string1*" is not supported in mode "CombinedRoleProfileSubsystem".

Explanation: The specified attribute is not supported in the specified mode.

Action: Correct the list of selected attributes.

ERR(ASC148): Internal error: attribute name "*string1*" does not contain any dot.

Explanation: An internal error occurred.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC149): Internal error: cannot find hash map for "*string1*".

Explanation: An internal error occurred.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC150): Received an exception "*string1*" while executing function "SUSR_USER_CHANGE_PASSWORD_RFC" for user "*string2*".

Explanation: The password cannot be changed. Read the error message and perform appropriate actions.

Action: Check the password policy on the R/3 system.

ERR(ASC151): Unsupported attribute in filter: "*string1*".

Explanation: Filtering not possible.

Action: Correct the filter.

ERR(ASC152): Unsupported extensible filter match.

Explanation: Extensible filter match is not supported.

Action: Correct the filter.

ERR(ASC153): Unsupported approximate filter match.

Explanation: Approximate filter match is not supported.

Action: Correct the filter.

ERR(ASC154): Tokenizing the filter string failed: *string1*.

Explanation: Tokenizing the filter string failed.

Action: Read the error message and perform appropriate actions.

ERR(ASC155): Pattern matching failed due to incorrect pattern: *string1*.

Explanation: A substring match cannot be processed.

Action: Read the error message and perform appropriate actions.

ERR(ASC156): The agent does not support the single entry search base of a system user "*string1*".

Explanation: Certain R/3 users cannot be exported by design. Check the manual about the list of exceptions.

Action: Correct your search.

ERR(ASC157): Single entry search base specified of a user "*string1*" but found *string2* matching entries.

Explanation: Single entry search base looks in user table for lock status of the single user but found multiple entries.

Action: Correct your search base.

ERR(ASC158): Exception in "*string1*".

Explanation: An exception occurred during the mentioned processing.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC159): Internal error structure.field *string1*.*string2*. Cannot determine associated x-flag structure.

Explanation: The necessary x-flag structure could not be determined.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC160): Pool specification for connection pool *string1* not accepted by the pool manager: *string2*.

Explanation: The pool manager does not accept the pool specification. Using another connector type may help.

Action: Check the detailed message and perform appropriate actions.

ERR(ASC161): Cannot return R/3 connection to host *string1* as user *string2* to connection pool *string3*: *string4*.

Explanation: The connection pool does not accept the returned connection.

Action: Check the detailed message and perform appropriate actions.

ERR(ASC162): Internal error while processing attributes in filters that are not in the list of selected attributes: "*string1*".

Explanation: An exception occurred while processing filter attributes.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC163): Internal error opening trace file for requests: "*string1*".

Explanation: An exception occurred while opening trace file.

Action: Inform DirX Identity technical support about the problem.

ERR(ASC164): Error while generating an temporary password: *string1*.

Explanation: An exception occurred while trying to generate an initial (temporary) password.

Action: Check the exception message. Probably parameters for the internal password generation have to be adapted.

ERR(ASC165): Unsupported present filter match.

Explanation: Present filter match is not supported.

Action: Correct the filter.

ERR(ASC166): Internal mismatch: selection_range table should be changed but current row pointer is not on last row.

Explanation: Unexpected table pointer value.

Action: Inform DirX Identity technical support about the problem.

WAR(ASC300): *string1*: '*string2*' (id: *string3*, parameter: *string4*, row: *string5*, field: *string6*, system: *string7*, user: *string8*) [*string9*]

Explanation: The R/3 system sent a warning message.

Action: Read the message to decide whether this is negligible or not.

WAR(ASC301): Cannot convert value "*string1*" for attribute "*string2*" from Unicode to UTF-8.

Explanation: The specified value cannot be converted to UTF-8.

Action: Correct the value.

WAR(ASC302): Internal error: Cannot extract value "*string1*" for attribute "*string2*" from row

(plen=*string3*, fi.len=*string4*, fi.rowOffset=*string5*).

Explanation: A multi-value attribute cannot be extracted.

Action: Inform DirX Identity technical support about the problem.

WAR(ASC303): General JCo exception "*string1*" received.

Explanation: The JCo middleware sent an exception.

Action: Inform DirX Identity technical support about the problem.

WAR(ASC304): Attribute dxrSTState with value "*string1*" ignored.

Explanation: Only ENABLED or DISABLED are allowed.

Action: Correct the value.

WAR(ASC305): Attribute "*string1*" is multi-valued; attribute is ignored.

Explanation: Only single-valued attributes are allowed.

Action: Correct the attribute.

WAR(ASC306): Attribute "*string1*" is not supported in request.

Explanation: The specified attribute is not supported (anymore).

Action: Correct the list of selected attributes.

WAR(ASC307): General JCo exception "*string1*" received.
Stack trace: *string2*

Explanation: An unexpected error occurred.

Action: Inform DirX Identity technical support about the problem.

WAR(ASC308): Received an exception "*string1*" while executing function "SUSR_USER_CHANGE_PASSWORD_RFC" for user "*string2*".

Explanation: Password cannot be changed. Read the error message and perform appropriate actions.

Action: Check the password policy on the R/3 system.

WAR(ASC309): Cannot convert string "*string1*" to integer. Configuration parameter "*string2*" ignored.

Explanation: Converting string to integer not supported.

Action: Correct the configuration parameter.

WAR(ASC310): No attribute values specified for attribute "*string1*". Attribute ignored.

Explanation: Attributes without values cannot be specified.

Action: Specify attribute values.

WAR(ASC311): Attribute value "*string1*" has no delimiter as expected in mode "CombinedRoleProfileSubsystem". Attribute value ignored.

Explanation: Attributes must be enclosed into delimiters.

Action: Enclose attributes into delimiters.

WAR(ASC312): Attribute "*string1*" is not supported unless mode "CombinedRoleProfileSubsystem" is set.

Explanation: The specified attribute is only supported for mode "CombinedRoleProfileSubsystem".

Action: Remove the attribute or specify the correct mode.

WAR(ASC313): Attribute "*string1*" is not supported in mode "CUA" access.

Explanation: The specified attribute is not supported for mode "CUA".

Action: Remove the attribute or specify the correct mode.

WAR(ASC315): JCo exception "*string1*" received. (Probably an attribute is not supported in the search request.)

Explanation: A specified attribute may not be supported.

Action: Either change selected attributes or inform DirX Identity technical support about the problem.

WAR(ASC316): File "conn.properties" could not be written: "*string1*".

Explanation: The file conn.properties contains the connection parameter and is written in log level 9. However the file could not be written due to the reason returned in the error message.

Action: See the returned error message. Perform appropriate actions.

WAR(ASC318): Incorrect ID type *string1* specified for search base.

Explanation: Only urn:oasis:names:tc:SPML:01:0#UserIDAndOrDomainName is accepted. Search base is ignored.

Action: Correct the ID type.

WAR(ASC319): Search base specified without any IdentifierChoice.

Explanation: Search base is ignored.

Action: Specify an IdentifierChoice.

WAR(ASC320): JCo exception "*string1*" received.

Explanation: JCo returned an exception.

Action: Either change attributes or inform DirX Identity technical support about the problem.

WAR(ASC321): Attribute "*string1*" is not supported if R/3 server release supports "*string2*".

Explanation: The R/3 server does not support the specified attribute.

Action: Remove the attribute.

WAR(ASC322): This R/3 server release does not support attribute "*string1*".

Explanation: The R/3 server does not support the specified attribute.

Action: Remove the attribute.

WAR(ASC323): Parsing error of date attribute "*string1*": *string2*.

Explanation: For filtering activity groups the FROM_DAT or TO_DAT is taken.

Action: Correct attribute values.

WAR(ASC324): Ending the stateful connection failed: *string2*.

Explanation: Ending a stateful call sequence for calls to the used destination failed.

Action: Correct attribute values.

WAR(ASC325): Could not set parameter CACHE_RESULTS to blank for BAPI_USER_GET_DETAIL: *string1*.

Explanation: Setting CACHE_RESULTS to blank should have the effect that the runtime of the BAPI_USER_GET_DETAIL call is constant and no data for the specified user remains in the buffer.

Action: See SAP note 1101858.

WAR(ASC326): Could not print request to file: *string1*.

Explanation: Tracing the request to file failed.

Action: Correct attribute values.

WAR(ASC327): Resetting the initial password to the given new password failed: *string1*.

Explanation: Even resetting the initial password failed. Before an internal initial password was set then setting a productive password failed.

Action: Only a SAP administrator can now reset the account password.

WAR(ASC328): Upgrade your JCo version to at least 3.0.6. You are using *string1*.

Explanation: In versions previous to 3.0.6 was an issue with communication errors in the R/3 system log.

Action: Check the given JCo version.

WAR(ASC329): General exception "*string1*" received.

Explanation: An internal or external exception occurred.

Action: Check the message text and eventually inform DirX Identity technical support about the problem.

WAR(ASC330): Function "BAPI_USER_CREATE1" has not the appropriate patch level: "*string1*". Option "setProductivePwdAtAddDirectly" can not be supported.

Explanation: The feature for setting the productive password directly can not be activated. Old 2 stage mechanism is used.

Action: Install the necessary SAP patches.

WAR(ASC331): Function "BAPI_USER_CHANGE" has not the appropriate patch level: "*string1*". Option "setProductivePwdAtModDirectly" can not be supported.

Explanation: The feature for modifying the productive password directly can not be activated. Old 2 stage mechanism is used.

Action: Install the necessary SAP patches.

WAR(ASC332): No SNC communication is set. Option "setProductivePwdAtModDirectly" can not be supported.

Explanation: The feature for modifying the productive password directly can not be activated. Old 2 stage mechanism is used.

Action: Set up an SNC communication.

WAR(ASC333): Preparing to delete an *string1*. Mandatory attribute *string2* - number of values *string3*, attribute *string4* - number of values *string5*.

Explanation: For deleting a group there are not all necessary key attribute values given. The numbers of values are not matching.

Action: Check the incoming SPML request.

INF(ASC500): *string1*: '*string2*' (id: *string3*, parameter: *string4*, row: *string5*, field: *string6*, system: *string7*, user: *string8*) [*string9*]

Explanation: The R/3 system returned an informal message.

Action: None.

INF(ASC501): Operation succeeded: '*string1*'

Explanation: Connector operation succeeded.

Action: None.

INF(ASC502): *** Connector '*string1*', version '*string2*', started at *string3* (log *string4*, CUA *string5*) ***

Explanation: Displays connector version and other information.

Action: None.

INF(ASC503): *** Connector ended *string1* ***

Explanation: Connector run finished.

Action: None.

INF(ASC504): Single entry search base specified for user "*string1*".

Explanation: Search for one user.

Action: None.

INF(ASC505): Reconnect to R/3 server: *string1* as user *string2* and client *string3* successful.

Explanation: The connection to the R/3 server has been re-established successfully.

Action: None.

INF(ASC506): Connecting to R/3 server: *string1* as user *string2* and client *string3* successful.

Explanation: The connection to the R/3 server has been established successfully.

Action: None.

INF(ASC507): Attribute "*string1*" is not supported in mode "non-CUA" access.

Explanation: The mode "non-CUA" does not support the specified attribute.

Action: Correct the configuration.

INF(ASC508): Single entry search base specified for SAP role "*string1*".

Explanation: Search for one SAP role.

Action: None.

INF(ASC509): Single entry search base specified for SAP profile "*string1*".

Explanation: Search for one SAP profile.

Action: None.

INF(ASC510): SPML search filter can be mapped to a BAPI_USER_GETLIST filter expression.

Explanation: Search for users will be done on the SAP system side.

Action: None.

INF(ASC511): SPML search filter can NOT be mapped to a BAPI_USER_GETLIST filter expression.

Explanation: Search for users will be done on the connector side.

Action: Your SPML filter can not be mapped to an valid BAPI_USER_GETLIST filter.

INF(ASC512): Configuration SetPwdDirectly: Add *string1*, Mod *string2*.

Explanation: Displays certain configuration information.

Action: None.

INF(ASC513): "*string1*" is a valid *string2* structure field with value "*string3*".

Explanation: Certain input fields are displayed.

Action: None.

INF(ASC514): Request identifier is "*string1*" (name capitalized by connector).

Explanation: The identifying name for the request is given.

Action: None.

INF(ASC515): Request to modify table with operation ADD for "*string1*" skipped because value is already in table (duplicate).

Explanation: Adding a row in table with an value that is already in the table is skipped to prevent duplicated rows.

Action: None.

INF(ASC516): User *string1* locked. Operation will be retried in *string2* milliseconds.

Explanation: User locked. Modification will be retried in the given time period.

Action: None.

INF(ASC517): User *string1* might still be locked but number of operation retries exceeded.

Explanation: Number of operation retries exceeded.

Action: Look for other error/warning messages.

16.3.7. Logging Messages (ASI)

This topic describes the logging messages returned by the DirX Identity SiPass agent.

ERR(ASI001): Unable to assign the specified Workgroup '*string1*', verify whether the Workgroup exists.

Explanation: The SiPass agent is unable to find and assign the specified Workgroup.

Action: Ensure that there is the specified Workgroup in the SiPass system.

ERR(ASI002): Unable to assign the specified AccessGroup '*string1*', verify whether the AccessGroup exists.

Explanation: The SiPass agent is unable to find and assign the specified AccessGroup.

Action: Ensure that there is the specified AccessGroup in the SiPass system.

ERR(ASI003): Could not add the attribute *string1* with the value '*string2*' to the SiPass user with the Reference (dn) '*string3*' due to SiPass COM API error '*string4*'.

Explanation: This message indicates an invalid attribute name or value.

Action: Check both the attribute name and its value. Check the local settings of the SiPass host. Correct the Import_Date_Format parameter in the configuration file if

needed.

ERR(ASI004): Could not add the attribute *string1* with the value '*string2*' to the SiPass user with the Reference (dn) '*string3*' due to invalid value '*string4*'.

Explanation: This message indicates an invalid attribute name or value.

Action: Check both the attribute name and its value. Check the local settings of the SiPass host. Correct the Import_Date_Format parameter in the configuration file if needed.

ERR(ASI005): Unable to add the card holder with the Reference (dn) '*string1*' due to invalid or missing Workgroup.

Explanation: Could not add the card holder with the specified Reference (dn) because no Workgroup was assigned or the assigned Workgroup is not valid.

Action: Check whether the Workgroup is mapped correctly. Check whether the name of the related Workgroup exists in SiPass.

ERR(ASI008): Could not modify the attribute *string1* with the value '*string2*' to the SiPass user with the Reference (dn) '*string3*' due to SiPass COM API error '*string4*'.

Explanation: This message indicates an invalid attribute name or value.

Action: Check both the attribute name and its value. Check the local settings of the SiPass host. Correct the Import_Date_Format parameter in the configuration file if needed.

ERR(ASI009): Could not modify the attribute *string1* with the value '*string2*' to the SiPass user with the Reference (dn) '*string3*' due to invalid value '*string4*'.

Explanation: This message indicates an invalid attribute name or value.

Action: Check both the attribute name and its value. Check the local settings of the SiPass host. Correct the Import_Date_Format parameter in the configuration file if needed.

ERR(ASI010): Unable to locate the desired card holder with the Reference (dn) '*string1*' to be deleted.

Explanation: Could not find the card holder with the specified Reference.

Action: Check whether the specified card holder exists in SiPass. Check its reference.

ERR(ASI011): Converting the SiPass date format to the generalized time format failed. An exception '*string1*' occurred.

Explanation: SiPass delivers an unexpected date format.

Action: Inform DirX Identity technical support about the problem.

ERR(ASI012): An exception occurred while reading the configuration file '*string1*', the file not found.

Explanation: The SiPass agent could not find the given configuration file.

Action: Check whether the path to the configuration file is correct. Check the access rights of the configuration file.

ERR(ASI013): An exception occurred while reading the configuration file '*string1*'. Probably directory does not exist or is not accessible.

Explanation: The SiPass agent could not find the given configuration file.

Action: Check whether the path to the configuration file is correct. Check the access rights of the configuration file.

ERR(ASI014): An exception occurred while reading the configuration file '*string1*', an XmlException '*string2*'.

Explanation: The configuration file is corrupted or is not in a valid XML format.

Action: Check XML format of the configuration file. For example, try to open it with a web browser.

ERR(ASI015): An exception occurred while reading the configuration file '*string1*', an IOException '*string2*'.

Explanation: An I/O error occurred while reading the configuration file.

Action: Analyze carefully this error message and try to avoid it. Check readability of the configuration file, check whether its path is correct.

ERR(ASI016): An exception occurred while reading the configuration file '*string1*', an Exception '*string2*'.

Explanation: A general error occurred while reading the configuration file.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI017): Cannot initialize logging.

Explanation: This message only follows to previously logged errors.

Action: Analyze previously logged errors.

ERR(ASI018): Corrupted configuration file '*string1*': missing the attributes server, user or password.

Explanation: The SiPass agent was not able to get information about the mandatory configuration parameters server, user, or password from the configuration file.

Action: Check validity of the given configuration file and check whether the related information is specified correctly.

ERR(ASI019): Corrupted configuration file '*string1*': missing the attribute filename for output file.

Explanation: The SiPass agent was not able to get information about the mandatory

configuration parameter filename from the configuration file.

Action: Check validity of the given configuration file and check whether the related information is specified correctly.

ERR(ASI020): Corrupted configuration file '*string1*': missing the format string Export_Date_Format for exporting date from SiPass.

Explanation: The SiPass agent was not able to get information about the mandatory configuration parameter Export_Date_Format from the configuration file.

Action: Check validity of the given configuration file and check whether the related information is specified correctly.

ERR(ASI021): Corrupted configuration file '*string1*': invalid date format for the Export_Date_Format parameter detected.

Explanation: The SiPass agent does not know the format of the Export_Date_Format parameter.

Action: Check validity of the given configuration file and check whether the related information is specified correctly. The parameter Export_Date_Format can only use a permutation of the letters 'm', 'd', and 'y' separated by a '/' or '.' (Example: m/d.y).

ERR(ASI022): Corrupted configuration file '*string1*': missing the attribute filename for input file.

Explanation: The SiPass agent was not able to get information about the mandatory configuration parameter filename from the configuration file.

Action: Check validity of the given configuration file and check whether the related information is specified correctly.

ERR(ASI023): Corrupted configuration file '*string1*': missing the format string Import_Date_Format for importing date to SiPass.

Explanation: The SiPass agent was not able to get information about the mandatory configuration parameter Import_Date_Format from the configuration file.

Action: Check validity of the given configuration file and check whether the related information is specified correctly.

ERR(ASI024): Corrupted configuration file '*string1*', invalid date format for the Import_Date_Format parameter detected.

Explanation: The SiPass agent does not know the format of the Import_Date_Format parameter.

Action: Check validity of the given configuration file and check whether the related information is specified correctly. The parameter Import_Date_Format can only use a permutation of the letters 'm', 'd', and 'y' separated by a '/' or '.' (Example: m/d.y).

ERR(ASI025): A COMException '*string1*' occurred.

Explanation: Initialization of the SiPass agent fails.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI026): SiPass reports unauthorized access. Check SiPass access rights for logged user.

Explanation: The logged user has not sufficient rights to access the SiPass API. This error can be caused also by an I/O error.

Action: Check the permissions of the user account used for logging to the SiPass API. Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI027): An Exception '*string1*' occurred.

Explanation: Initialization of the SiPass agent fails due to a general error.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI028): An exception occurred while parsing the input LDIF file '*string1*', the file not found.

Explanation: The SiPass agent could not find the specified input file.

Action: Check whether the path to the input file is correct. Check the access rights of the input file.

ERR(ASI029): An exception occurred while parsing the input LDIF file '*string1*'. Probably directory does not exist or is not accessible.

Explanation: The SiPass agent could not find the specified input file.

Action: Check whether the path to the input file is correct. Check the access rights of the input file.

ERR(ASI030): An exception occurred while parsing the input LDIF file '*string1*', an IOException '*string2*'.

Explanation: An I/O error occurred while reading the input file.

Action: Analyze carefully this error message and try to avoid it. Check readability of the input file, check whether its path is correct.

ERR(ASI031): An exception occurred while parsing the input LDIF file '*string1*', an Exception '*string2*'.

Explanation: A general error occurred while reading the input file.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI032): The unsupported action '*string1*' for the user with the Reference (dn) '*string2*'.

Explanation: This message indicates that the attribute 'changetype' is set to other value

than 'add', 'modify' or 'delete'.

Action: Check correctness of the mapping scripts.

ERR(ASI033): A COMException '*string1*' occurred while processing the user entry with the Reference (dn) '*string2*'.

Explanation: Could not finish the processing of the related user entry due to SiPass errors.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI034): A NullReferenceException '*string1*' occurred while processing the user entry with the Reference (dn) '*string2*'.

Explanation: Could not finish the processing of the related user entry due to invalid data or an internal error.

Action: Inform DirX Identity technical support about the problem.

ERR(ASI035): A MissingFieldException '*string1*' occurred while processing the user entry with the Reference (dn) '*string2*'.

Explanation: Could not finish the processing of the related user entry due to invalid data or an internal error.

Action: Inform DirX Identity technical support about the problem.

ERR(ASI036): An ArgumentException '*string1*' occurred while processing user the entry with the Reference (dn) '*string2*'.

Explanation: Could not finish the processing of the related user entry due to invalid data.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI037): An UnauthorizedAccessException '*string1*' occurred while processing the user entry with the Reference (dn) '*string2*'.

Explanation: This message describes an access error caused by another I/O error or a security error. It is caused by OS. Could not finish the processing of the related user entry.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI038): An Exception '*string1*' occurred while processing the user entry with the Reference (dn) '*string2*'.

Explanation: This message describes a general error. Could not finish the processing of the related user entry.

Action: Analyze carefully this error message and try to avoid it. If the problem persists

inform DirX Identity technical support about the problem.

ERR(ASI039): The changetype attribute is missing. Ignoring the user entry with the Reference (dn) '*string1*'.

Explanation: This message indicates that the attribute changetype is missing in the LDIF input file which describes the type of an operation (add, modify or delete).

Action: Check the mapping scripts.

ERR(ASI040): An exception occurred while parsing the file '*string1*' with SPML request, the file not found.

Explanation: The SiPass agent could not find the specified request file.

Action: Check whether the path to the request file is correct and whether it is accessible.

ERR(ASI041): An exception occurred while parsing the file '*string1*' with SPML request. Probably directory does not exist or is not accessible.

Explanation: The SiPass agent could not find the specified request file.

Action: Check whether the path to the configuration file is correct and whether it is accessible.

ERR(ASI042): An exception occurred while parsing the file '*string1*' with SPML request, an IOException '*string2*'.

Explanation: An I/O error occurred while reading the request file.

Action: Analyze carefully this error message and try to avoid it. Check the readability of the request file, check whether the path is correct.

ERR(ASI043): An exception occurred while parsing the file '*string1*' with SPML request, an XmlException '*string2*'.

Explanation: The specified request file is corrupted or not in a valid xml format.

Action: Check xml validity of the request file. For example try to open it in a web browser.

ERR(ASI044): An exception occurred while parsing the file '*string1*' with SPML request, an Exception '*string2*'.

Explanation: A general error occurred while reading the request file.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI045): An exception occurred while creating the output LDIF file '*string1*', a COMException '*string2*'.

Explanation: An error occurred when the SiPass agent was reading the SiPass data.

Action: Analyze carefully this error message and try to avoid it. If the problem persists

inform DirX Identity technical support about the problem.

ERR(ASI046): An exception occurred while creating the output LDIF file '*string1*', an ArgumentException '*string2*'.

Explanation: Could not finish creating of the output file due to invalid data.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI047): An exception occurred while creating the output LDIF file '*string1*', a NullReferenceException '*string2*'.

Explanation: Could not finish creating of the output file due to invalid data.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI048): An exception occurred while creating the output LDIF file '*string1*', the file not found.

Explanation: The SiPass agent could not get access to the specified output file.

Action: Check whether the path to the output file is correct and whether it is accessible.

ERR(ASI049): An exception occurred while creating the output LDIF file '*string1*'. Probably directory does not exist or is not accessible.

Explanation: The SiPass agent could not get access to the specified output file.

Action: Check whether the path to the output file is correct and whether it is accessible.

ERR(ASI050): An exception occurred while creating the output LDIF file '*string1*', an UnauthorizedAccessException '*string2*'.

Explanation: This message describes an access error caused by another I/O error or a security error. It is caused by OS. Could not finish creating of the output file.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI051): An exception occurred while creating the output LDIF file '*string1*', an Exception '*string2*'.

Explanation: A general error occurred while creating the output file.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI052): The invalid value '*string1*' for command line parameter '*string2*' was detected.

Explanation: An invalid value was specified for the parameter.

Action: Use only documented values for parameters. See documentation for more details.

ERR(ASI053): The type of the required action was not set on command line. Use 'Import' or 'Export'.

Explanation: The type of the action is missing on command line.

Action: Check specified command line parameters. Add the keyword 'Import' or 'Export' for action.

ERR(ASI054): The name of the configuration file was not set on command line.

Explanation: The name of the configuration file was not specified.

Action: Check specified command line parameters. Add the name of the configuration file if needed.

ERR(ASI055): The name of the file with SPML search request was not set on command line.

Explanation: The name of the request file was not specified.

Action: Check specified command line parameters. Add the name of the request file if needed.

ERR(ASI056): Path to file with SPML search request was not set on command line. Use parameter '-f path_to_search_request'.

Explanation: The path to the request file was not specified.

Action: Check specified command line parameters. Add missing parameters.

ERR(ASI057): Action or path to configuration file was not set on command line. Use parameter '-a Import' or '-a Export' for choosing .action and '-c path_to_config_file' for path to configuration file.

Explanation: The path to the configuration file or action type was not specified.

Action: Check specified command line parameters. Add missing parameters.

ERR(ASI058): The null value instead of a valid trace file name detected.

Explanation: This message describes an internal error of the SiPass agent.

Action: Inform DirX Identity technical support about the problem.

ERR(ASI059): Could not find the directory for the trace file '*string1*'. Probably directory does not exist or is not accessible.

Explanation: The SiPass agent could not access the directory with the trace file.

Action: Check whether the path to the trace file is correct and whether it is accessible.

ERR(ASI060): An IOException '*string1*' occurred for trace file '*string2*'.

Explanation: An I/O error occurred while creating the trace file.

Action: Analyze carefully this error message and try to avoid it. Check whether the path to the trace file is correct.

ERR(ASI061): An Exception '*string1*' occurred for trace file '*string2*'.

Explanation: A general error occurred while creating the trace file.

Action: Analyze carefully this error message and try to avoid it. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI062): Multiple values for the attribute dn or Reference, other value '*string1*' found. The whole entry '*string2*' will be ignored.

Explanation: Both the attributes dn and Reference are present in the input file for one entry.

Action: Correct the mapping scripts. Remove mapping for Reference attribute in input file.

ERR(ASI063): Multiple values for the attribute changetype, other value '*string1*' found. The whole entry '*string2*' will be ignored.

Explanation: Incorrect format of the input file. Multiple occurrences for the special attribute changetype are not allowed for one entry.

Action: Correct the mapping scripts.

ERR(ASI064): Multiple values for the attribute *string1* detected, delete request for this attribute is not possible. The whole entry '*string2*' will be ignored.

Explanation: Multiple values for an attribute specified. SiPass does not support multi-valued attributes.

Action: Check values of the mapped attributes.

ERR(ASI065): Multiple values for the attribute *string1* detected, other value '*string2*' found. The whole entry '*string3*' will be ignored.

Explanation: Multiple values for an attribute specified. SiPass does not support multi-valued attributes.

Action: Check values of the mapped attributes.

ERR(ASI066): Could not add the user entry to the user collection because the dn attribute is missing.

Explanation: The input file is not created correctly and no dn (used as Reference) was provided.

Action: Check the mapping scripts. For each entry a dn attribute must be specified.

ERR(ASI067): Could not get the message with identifier '*string1*' from SiPass Agent resource.

Explanation: This message describes an internal error of the SiPass agent.

Action: Inform DirX Identity technical support about the problem.

ERR(ASI080): Missing value for the required attribute '*string1*'.

Explanation: A value for the mandatory attribute CardNumber or Reference (dn) is missing in the input file.

Action: Check mapping scripts. For each entry a dn and CardNumber attribute value must be specified.

ERR(ASI081): The required attribute CardNumber is missing.

Explanation: The mandatory attribute CardNumber is missing in the input file.

Action: Check mapping scripts. For each entry a CardNumber attribute value must be specified.

ERR(ASI082): The desired card holder with the Reference (dn) '*string1*' already exists.

Explanation: Could not finish the add operation because the card holder with the specified Reference (dn) already exists.

Action: Check whether the Reference number is unique in the DirX IdentityStore for all SiPass users.

ERR(ASI083): Missing value for the required attribute '*string1*'.

Explanation: A value for the mandatory attribute CardNumber is missing in the input file.

Action: Check mapping scripts. For each entry a CardNumber attribute value must be specified.

ERR(ASI084): Could not locate the desired card holder with the reference (dn) '*string1*'.

Explanation: Could not finish the modify operation because the card holder with the specified Reference (dn) does not exist.

Action: Check existence of the related SiPass user.

ERR(ASI085): '*string1*' is an unrecognized object type.

Explanation: The specified search request file contains an incorrect object type that could not be exported.

Action: Check validity of the search request file, only the object types 'Employee' 'Workgroup' or 'AccessGroup' can be exported.

ERR(ASI086): Invalid value '*string1*' for year.

Explanation: Converting the SiPass date format to the generalized time format failed, the parsed value for year was invalid.

Action: Check the parameter 'Export_Date_Format' in the configuration file, try to change this format string to other values. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI087): Invalid value '*string1*' for month.

Explanation: Converting the SiPass date format to the generalized time format failed,

the parsed value for month was invalid.

Action: Check the parameter 'Export_Date_Format' in the configuration file, try to change this format string to other values. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI088): Invalid value '*string1*' for day.

Explanation: Converting the SiPass date format to the generalized time format failed, the parsed value for day was invalid.

Action: Check the parameter 'Export_Date_Format' in the configuration file, try to change this format string to other values. If the problem persists inform DirX Identity technical support about the problem.

ERR(ASI089): No value for the dn attribute was set.

Explanation: A user entry in the input file has no mandatory dn attribute.

Action: Check mapping scripts. For each entry a dn attribute value must be specified.

ERR(ASI090): No value for changetype attribute was set.

Explanation: A user entry in the input file has no mandatory changetype attribute.

Action: Check mapping scripts. For each entry a changetype attribute value must be specified.

ERR(ASI091): An attribute name for delete attribute request missing.

Explanation: A user entry in the input file has an invalid delete attribute value. The name of the attribute to be deleted is missing.

Action: Check mapping scripts.

ERR(ASI092): Error while retriving the attribute value '*string1*' for the *string2* entry '*string3*'. An exception '*string4*' occurred.

Explanation: The SiPass agent could not retrieve the attribute value for the specified entry.

Action: Check whether the attribute name is available in the SiPass.

ERR(ASI301): Could not correctly close the connection to the SiPass server. An exception '*string1*' occurred.

Explanation: This message describes the cause that prevented the SiPass agent from successful closure of the connection to the SiPass server.

Action: Check the error message and try to correct it. Inform DirX Identity technical support about the problem.

INF(ASI501): Opening connection to the SiPass server '*string1*'.

Explanation: This message is logged when establishing the connection to the SiPass server.

Action: No action needed.

INF(ASI502): Successfully connected to the SiPass server.

Explanation: Connection to the SiPass server was successfully established.

Action: No action needed.

INF(ASI503): Closing connection to the SiPass server.

Explanation: The connection to the SiPass server was closed.

Action: No action needed.

INF(ASI504): Attempt to add the card holder with the Reference (dn) '*string1*'.

Explanation: The add operation is started.

Action: No action needed.

INF(ASI505): Adding new the card holder completed.

Explanation: The add operation is finished.

Action: No action needed.

INF(ASI506): The card holder with the Reference (dn) '*string1*' already exists. Attempt to modify this user.

Explanation: Could not finish the add operation because the related user already exists. If the 'exact_add_action' parameter in the configuration is set to 'false', the modify operation is invoked instead.

Action: Check whether the Reference number is unique in the DirX IdentityStore for all SiPass users.

INF(ASI507): Attempt to modify the card holder with the Reference (dn) '*string1*'.

Explanation: The modify operation is started.

Action: No action needed.

INF(ASI508): Unable to locate the desired card holder with the Reference (dn) '*string1*'. Attempt to add this user.

Explanation: Could not finish the modify operation because the related user does not exist. If the 'exact_modify_action' parameter in the configuration is set to 'false', the add operation is invoked instead.

Action: Check the existence of the related SiPass user.

INF(ASI509): Generating the LDIF file with the SiPass card holders.

Explanation: Exporting of the SiPass card holders is started.

Action: No action needed.

INF(ASI510): Generating the LDIF file with the SiPass Workgroups.

Explanation: Exporting of the SiPass Worksgroups is started.

Action: No action needed.

INF(ASI511): Generating the LDIF file with SiPass AccessGroups.

Explanation: Exporting of the SiPass AccessGroups is started.

Action: No action needed.

INF(ASI512): Generating the LDIF file completed.

Explanation: The export operation is finished.

Action: No action needed.

INF(ASI513): The empty value of the Workgroup attribute detected, using the default value.

Explanation: The SiPass agent recognized an empty string instead of a valid Workgroup name. Because Workgroup is a mandatory attribute a default Workgroup will be assigned.

Action: Check mapping scripts and values mapped to the Workgroup attribute.

INF(ASI514): Exporting the card holder with the Reference (dn) '*string1*'.

Explanation: Exporting of the specified SiPass card holder is started.

Action: No action needed.

INF(ASI515): Exporting the Workgroup '*string1*'.

Explanation: Exporting of the specified SiPass Workgroup is started.

Action: No action needed.

INF(ASI516): The value '*string1*' has not generalized time format. Conversion disabled.

Explanation: The incoming StartDate or EndDate attribute is not in generalized time format. The internal conversion to SiPass format is disabled.

Action: If the incoming attribute value is already of a format accepted by SiPass no action is needed. Otherwise correct the mapping scripts to get the attribute value in generalized time format or in SiPass format.

INF(ASI517): SiPass returned unknown date format. Conversion disabled.

Explanation: The SiPass agent could not work with the format of date delivered by SiPass. The internal conversion to generalized time format is disabled.

Action: Inform DirX Identity technical support about the problem.

INF(ASI518): Using the default value '*string1*' for the EndDate attribute.

Explanation: The mandatory attribute EndDate is missing in the input file. The default

value (current day plus 100 years) is used.

Action: If the default value is not acceptable correct the mapping scripts for the EndDate attribute.

INF(ASI519): Using the default value '*string1*' for the StartDate attribute.

Explanation: The mandatory attribute StartDate is missing in the input file. The default value (current day) is used.

Action: If the default value is not acceptable correct the mapping scripts for the StartDate attribute.

INF(ASI520): Corrupted configuration file '*string1*': missing the logging level or filename attribute. Using the default settings 'trace.trc' for the trace file name and logging level '1'.

Explanation: The SiPass agent was not able to get information about the mandatory configuration parameters for the logging level or the trace file name from the configuration file. Default values are used.

Action: Check validity of the configuration file. Check whether the related information is specified correctly.

INF(ASI521): Processing the import action.

Explanation: Importing of the SiPass card holders is started.

Action: No action needed.

INF(ASI522): Processing the user collection.

Explanation: Parsing of the input LDIF file is finished and processing of parsed information is started.

Action: No action needed.

INF(ASI523): Processing the user with the Reference (dn) '*string1*'.

Explanation: Processing of the specified user entry is started.

Action: No action needed.

INF(ASI524): Trying the primary action 'add' card holder.

Explanation: The current user entry has the changetype attribute set to 'add'.

Action: No action needed.

INF(ASI525): Trying the primary action 'modify' card holder.

Explanation: The current user entry has the changetype attribute set to 'modify'.

Action: No action needed.

INF(ASI526): Trying the primary action 'delete' card holder.

Explanation: The current user entry has the changetype attribute set to 'delete'.

Action: No action needed.

INF(ASI527): Processing the export action.

Explanation: The export action is started.

Action: No action needed.

INF(ASI528): Parsing the file with SPML request.

Explanation: The export action has been started and the SiPass agent starts to parse the specified search request file.

Action: No action needed.

INF(ASI529): Parsing the file with SPML request completed.

Explanation: Parsing of the specified search request file was finished.

Action: No action needed.

INF(ASI530): Parsing the LDIF '*string1*'.

Explanation: The import action has been started and the SiPass agent starts to parse the specified input LDIF file.

Action: No action needed.

INF(ASI531): Parsing the LDIF completed.

Explanation: Parsing of the specified input LDIF file was finished.

Action: No action needed.

INF(ASI532): Processing the entry '*string1*'.

Explanation: This message indicates the parsing of the specific entry in the input LDIF file.

Action: No action needed.

INF(ASI533): Opening the SiPass agent version *string1*.

Explanation: The SiPass agent is started.

Action: No action needed.

INF(ASI534): The SiPass agent completed successfully without any errors.

Explanation: The SiPass agent detected no problems during processing its job.

Action: No action needed.

INF(ASI535): The SiPass agent completed with *string1* warning(s) and with *string2* error(s).

Explanation: The SiPass agent detected some problems during processing its job.

Action: Analyze the previously logged warning(s) and error(s).

INF(ASI536): Exiting the SiPass agent.

Explanation: The SiPass agent finished its job.

Action: No action needed.

INF(ASI537): The Workgroup '*string1*' to be deleted is not assigned to this user. This Workgroup cannot be deleted.

Explanation: The SiPass agent could not delete related group for this user.

Action: Check whether there is the name of the related Workgroup in SiPass.

INF(ASI538): Missing attribute value '*string1*' for the *string2* entry '*string3*'.

Explanation: The SiPass agent could not retrieve the attribute value for the specified entry.

Action: Check whether the attribute name is available in the SiPass.

16.3.8. Logging Messages (C1_)

This topic describes the logging messages returned by the DirX Identity C++-based Connector 1.

C1_2100 (INF): *string* (file: *string*, line: *number*)

Explanation: This is an informal notice.

Action: See the text for further information.

C1_2101 (INF): *string* (file: *string*, line: *number*)

Explanation: This is a verbose informal notice.

Action: See the text for further information.

C1_2102 (WAR): *string* (file: *string*, line: *number*)

Explanation: This is a warning notice.

Action: See the error text for further information.

C1_2103 (ERR): *string* (file: *string*, line: *number*)

Explanation: This is an error notice.

Action: See the error text for further information.

C1_2104 (FTL): *string* (file: *string*, line: *number*)

Explanation: This is a severe error notice.

Action: See the error text for further information.

16.3.9. Logging Messages (CSP)

This topic describes the logging messages returned by the Customer SharePoint User hook for the DirX Identity Join Engine.

INF(CSP250): Superior folder DN=*string1* created successfully.

Explanation: A superior folder has been created successfully.

Action: None.

ERR(CSP550): Searching for folder DN=*string1* returned an error: *string2*.

Explanation: The LDAP search to check the existence of the specified folder failed due to the reported error.

Action: Please check the detailed error message and perform appropriate actions.

ERR(CSP551): Cannot create folder DN=*string1* due to: *string2*.

Explanation: Adding a missing folder failed due to the reported error.

Action: Please check the detailed error message. Correct the error and run the validation workflow again.

16.3.10. Logging Messages (EEC)

This topic describes the logging messages returned by the Evidian Esso Connector.

INF(EEC401): Ignoring unknown attribute name *string1* during search.

Explanation: None.

Action: None.

INF(EEC402): Creation of *string1* successful.

Explanation: None.

Action: None.

INF(EEC403): Modification of *string1* successful.

Explanation: None.

Action: None.

INF(EEC404): Deletion of *string1* successful.

Explanation: None.

Action: None.

16.3.11. Logging Messages (GAC)

This topic describes the logging messages returned by the Google Apps Connector.

INF(GAC101): GoogleApps: connector opening...

Explanation: Google Apps connector is running the open procedure.

Action: None.

INF(GAC102): GoogleApps: connecting to domain *string1*.

Explanation: Google Apps connector is trying to connect to domain.

Action: None.

INF(GAC103): GoogleApps: connected to *string1*.

Explanation: Google Apps connector, established the connection to target domain.

Action: None.

INF(GAC104): Admin user provided is empty.

Explanation: Admin user provided is empty.

Action: Admin user provided is empty.

INF(GAC105): The domain provided is empty.

Explanation: The domain provided is empty.

Action: The domain provided is empty.

INF(GAC106): The client Id provided is empty.

Explanation: The client Id provided is empty.

Action: The client Id provided is empty.

INF(GAC107): The client secret key provided is empty.

Explanation: The client secret key provided is empty.

Action: The client secret key provided is empty.

INF(GAC108): The application name provided is empty.

Explanation: The application name provided is empty.

Action: The application name provided is empty.

INF(GAC109): The admin email provided is empty.

Explanation: The admin email provided is empty.

Action: The admin email provided is empty.

ERR(GAC110): Exception while processing the request - entry not found: *string1*.

Explanation: The entry which the system is trying to access or modify is not available.

Action: Change the request attributes and try again.

ERR(GAC201): GoogleApps: Error connecting to Directory API: adminMail: *string1* domain: *string2* exc: *string3*.

Explanation: Google Apps connector, could not establish the connection to target domain.

Action: Check connection parameters.

ERR(GAC202): GoogleApps: Connector parameters not configured correctly.

Explanation: Google Apps connector was not configured correctly.

Action: Check connection parameters, you might be missing some mandatory fields.

ERR(GAC203): GoogleApps: Cannot get trust certificates from Google: *string1*.

Explanation: Google Apps connector was not able to get the trust certificates from Google.

Action: Try again later.

ERR(GAC204): GoogleApps: Error while generating the Google private key: *string1*.

Explanation: An error occurred while trying to decode the Google private key.

Action: Check the exception message. It could be that the key wasn't generated correctly.

ERR(GAC205): Exception trying to connect using the HttpTransport provided: proxyHost: *string1*, proxyPort: *string2*.

Explanation: An exception occurred while trying to connect using the HttpTransport provided, proxy might be needed.

Action: Check to see if you need to setup a proxy, or inform DirX Identity technical support about the problem.

ERR(GAC206): Exception while processing the request due to bad input parameters: *string1*.

Explanation: An exception occurred while processing the request due to bad input parameters.

Action: Check the request, it might be incomplete or inform DirX Identity technical support about the problem.

ERR(GAC207): Exception while processing the request - insufficient permissions: *string1*.

Explanation: The user doesn't have permission to perform that action.

Action: Check the exception message or inform DirX Identity technical support about the problem.

ERR(GAC208): Exception while processing the request - forbidden access: *string1*.

Explanation: The user trying to perform the action does no longer have the necessary rights.

Action: Check your connection properties or check to see if anything changed with your administrator account.

ERR(GAC210): Exception while processing the request - duplicate entry: *string1*.

Explanation: The entry which the system is trying to create already exists.

Action: Change the request attributes and try again.

ERR(GAC211): GoogleApps: NULL User object returned.

Explanation: Google Apps returned an empty user object.

Action: Check the request attributes and try again.

ERR(GAC212): GoogleApps: NULL Group object returned.

Explanation: Google Apps returned an empty group object.

Action: Check the request attributes and try again.

ERR(GAC213): GoogleApps: Invalid group name: *string1*.

Explanation: The group name provided is empty, null or invalid.

Action: Check the group name introduced and try again.

ERR(GAC214): GoogleApps: User was suspended by Google for abuse: *string1*.

Explanation: The user was suspended automatically by google; this account can be re-enabled only by the user.

Action: User must log into GoogleApps and reactivate his account.

ERR(GAC215): GoogleApps: Action failed due to a back-end error: *string1*.

Explanation: There was a Google specific problem with processing your request.

Action: Check the exception message or inform DirX Identity technical support about the problem.

ERR(GAC216): Exception when parsing elements of type: *string1*.

Explanation: The JSON formant that you provided for the specified element(s) is incorrect.

Action: Provide a valid JSON element.

FTL(GAC301): Exception in connector during authenticating : *string1*.

Explanation: An exception occurred while authenticating.

Action: Check the exception message or inform DirX Identity technical support about the problem.

FTL(GAC302): An exception occurred while authenticating due to invalid credentials:
accountId:_string1_
privateKey
account user:_string2_
message:_string3_.

Explanation: Some of the provided credential might be invalid.

Action: Check the exception message or inform DirX Identity technical support about the problem.

WAR(GAC400): GoogleApps: connector cannot handle attribute *string1*.

Explanation: The attribute provided could not be handled by the connector.

Action: Check that the attribute is correct and try again.

WAR(GAC401): GoogleApps: Error performing request: *string1*.

Explanation: Response from Google came with errors.

Action: Check the exception message or inform DirX Identity technical support about the problem.

16.3.12. Logging Messages (JDB)

This topic describes the logging messages returned by the DirX Identity JDBC connector.

ERR(JDB0): At *string1*: add-argument error: *string2*: *string3*

Explanation: There is an error in the argument of an add operation that prevents the operation from being carried out. The report should give a more detailed indication of what has gone wrong.

Action: Correct the operation and re-try.

WAR(JDB1): At *string1*: add-argument warning: *string2*: *string3*

Explanation: There is an unusual element in the argument of an add operation that did not stop its execution, but may produce unexpected results.

Action: Check the operation and correct it if necessary. The report should give a more detailed indication of the warning.

ERR(JDB10): At *string1*: extended request error: *string2*: *string3*

Explanation: An extended request operation has been provided that does not specify "SP" (stored procedures) as the provider ID type, or is otherwise unsupported.

Action: Check the form of the extended request, correct, and re-try.

ERR(JDB12): At *string1*: JDBC select error: *string2*: *string3*

Explanation: A SELECT access to the JDBC database, using the indicated SQL, has failed (for example when attempting to find an entry during an add, delete, or modify

operation).This may be caused by a table or view that has been removed since the JDBC connector started. Otherwise this error should not happen.

Action: It may be valuable to execute the SQL on the database manually to determine what has gone wrong, or to seek advice otherwise. The report may suggest an explanation that will allow the problem to be circumvented.

ERR(JDB13): At *string1*: JDBC system error: *string2*: *string3*

Explanation: This error should not happen. The JDBC database access failed. The cause is indicated in the report.

Action: Seek advice. The report may suggest an explanation that will allow the problem to be circumvented.

ERR(JDB14): At *string1*: JDBC update error: *string2*: *string3*

Explanation: An error has occurred in the execution of SQL during an add or delete operation. The cause is indicated in the report.

Action: Seek advice. The report may suggest an explanation that will allow the problem to be circumvented.

WAR(JDB15): At *string1*: JDBC warning: *string2*: *string3*

Explanation: A problem occurred in a add operation (e.g. unable to find one of the specified primary key values), but the operation still worked, or in a delete operation (a sub-operation required by referential integrity failed), or in a search operation (a column may no longer be available).

Action: Check the operation and re-try if necessary.

ERR(JDB16): At *string1*: modify-argument error: *string2*: *string3*

Explanation: An error occurred within the modifications specified by a modify operation (e.g. refers to a unknown table or to a table different to the one specified by the name supplied, or attribute was not modifiable, or there were no modifications).

Action: Correct the argument and retry.

WAR(JDB17): At *string1*: modify-argument warning: *string2*: *string3*

Explanation: A modify operation was accepted but had unexpected characteristics (e.g. the modified attribute had two values).

Action: Check the operation and retry.

ERR(JDB18): At *string1*: modify error: *string2*: *string3*

Explanation: There was a major error in executing a modify operation that prevents the operation from being carried out. This may be due to an error in the name (it is missing, or has faulty syntax, or doesn't find a unique entry in the database, or entry doesn't exist, or name is of a table or view, or specifies an entry within a view, or name isn't correctly interpreted by the database). Or the modify operation affects zero entries or more than one entry. The report should give a more detailed indication of what has gone wrong.

Action: Correct the operation and re-try.

INF(JDB19): At *string1*: modify operation: *string2*: *string3*

Explanation: Used for modify operation tracing only.

Action: None.

ERR(JDB2): At *string1*: add error: *string2*: *string3*

Explanation: There is a major error in executing the add operation, often an error in the name (faulty syntax, or doesn't find a unique entry in the database, or entry already exists, or name is prohibited because it is auto-generated, or name isn't correctly interpreted by the database, or refers to a view rather than to a single entry in the table). The report should give a more detailed indication of what has gone wrong.

Action: Correct the name specified by the operation, checking that its intent is valid for the database, and re-try.

FTL(JDB20): At *string1*: schema error: *string2*: *string3*

Explanation: The connector was unable to reference schema metadata.

Action: Seek advice on compatibility of the database with the JDBC connector.

INF(JDB22): At *string1*: schema warning: *string2*: *string3*

Explanation: A non-fatal error occurred when checking abbreviations or their characteristics. The report should indicate in more detail what the problem was.

Action: Review the configuration and correct it if necessary.

ERR(JDB23): At *string1*: search error: *string2*: *string3*

Explanation: This can occur during configuration with a reporting-point "schema report" when an unexpected error occurs during an automatic scan of the database is carried out for schema purposes. Otherwise, there was a major error in executing a search operation that prevents the operation from being carried out. This may be due to an error in the search base name (it has faulty syntax, or * doesn't find a unique entry in the database, or entry doesn't exist, or name isn't correctly interpreted by the database). Or the search operation specified attributes none of which can be returned. Or it specifies a table or view that is not configured as searchable. Or the SQL used is faulty. Or an unexpected database error occurred. The report should give a more detailed indication of what has gone wrong.

Action: If the reporting point was "schema report", correct the configuration. If the reporting point was "search report" correct the operation, if possible, and try again. In the case of a SQL error, it may be useful to try the SQL out directly on the database to find out what might have gone wrong.

INF(JDB24): At *string1*: search operation: *string2*: *string3*

Explanation: Used for search operation tracing only.

Action: None

WAR(JDB25): At *string1*: search warning: *string2*: *string3*

Explanation: A non-fatal error occurred, e.g. the return specified unknown attributes (columns) that were ignored in the execution of the search. Or the primary key specifies is binary and cannot be used. The report should give a more detailed indication of what has gone wrong.

Action: Check the operation and re-try if necessary.

ERR(JDB26): At *string1*: select results error: *string2*: *string3*

Explanation: An unexpected error occurred when attempting to analyze search results. This should not happen. The report should give a more detailed indication of what has gone wrong.

Action: Seek advice.

ERR(JDB27): At *string1*: spml error: *string2*: *string3*

Explanation: There is an error in a search filter that was undetected by the SPML analyzer. The report should give a more detailed indication of what is wrong.

Action: Correct the supplied operation and re-try.

ERR(JDB29): At *string1*: stored procedures error: *string2*: *string3*

Explanation: This error can occur when the JDBC Connector determines that stored procedures are not supported by the database. Otherwise, it is returned if there is a failure to execute a stored function or procedure. The report should give a more detailed indication of what is wrong.

Action: If stored procedures are not supported, there is no remedy apart from reconfiguring the database (if possible), or trying another database. Otherwise, the indication suggested by the report should indicate the needed re-definition of the operation; correct and re-try.

INF(JDB3): At *string1*: add operation: *string2*: *string3*

Explanation: Used for add operation tracing only.

Action: None.

INF(JDB30): At *string1*: stored procedures operation: *string2*: *string3*

Explanation: Used for stored function or procedure tracing only.

Action: None.

WAR(JDB31): At *string1*: stored procedures warning: *string2*: *string3*

Explanation: There are currently no uses of this warning.

Action: None.

FTL(JDB4): At *string1*: configuration error: *string2*: *string3*

Explanation: A fatal error has been found in the XML configuration file for the JDBC Connector. This could be a problem with the connection (perhaps unknown, or multiple, or unknown driver or database), a table, view, abbreviation, key, or relationship (e.g. referenced table or view is not known), or a mandatory component is unspecified or an argument has an illegal value, or a function/procedure specification is missing an element or has incorrect syntax, or there is a problem with an SQL script used in schema exploration. The report should give a more detailed indication of what has gone wrong.

Action: Correct configuration and retry.

INF(JDB5): At *string1*: configuration warning: *string2*: *string3*

Explanation: There is a problem in configuration that does not prevent operations that do not involve the affected component (e.g. an unknown table specified in an abbreviation or view), or a duplicate specification (perhaps inconsistent). The report should give a more detailed indication of the problem.

Action: Check configuration and correct the problem.

FTL(JDB6): At *string1*: connection error: *string2*: *string3*

Explanation: There is a fatal configuration error: either the database URL is invalid or faulty in some way, and the database driver manager cannot make a connection, or a connection has been made but is unusable as an SQL database.

Action: Check the database details and correct the specified URL if necessary.

ERR(JDB7): At *string1*: delete error: *string2*: *string3*

Explanation: There was a major error in executing a delete operation that prevents the operation from being carried out. This may be due to an error in the name (it is missing, or has faulty syntax, or doesn't find a unique entry in the database, or entry doesn't exist, or name is of a table or view, or name isn't correctly interpreted by the database). The report should give a more detailed indication of what has gone wrong.

Action: Correct the operation and retry.

INF(JDB8): At *string1*: delete operation: *string2*: *string3*

Explanation: Used for delete operation tracing only.

Action: None.

16.3.13. Logging Messages (JMS)

This topic describes the logging messages returned by the JMS connector for DirX Identity Connector Framework.

WAR(JMS001): Exception when closing JMS publisher: *string1*.

Explanation: The JMS publisher threw an exception when it is closed.

Action: Check if the JMS provider is available.

FTL(JMS002): Inconsistent JMS connector configuration.

Explanation: JMS Configuration option is missing or is inconsistent with others.

Action: Check JMS connector documentation for mandatory options and allowed values.

FTL(JMS003): Exception when initializing JMS connector: *string1*.

Explanation: Error when initializing the messaging context of the connector.

Action: Check your messaging environment: message server started and available, event activity of the workflow, etc.

ERR(JMS004): Exception when serializing SPML request *string1*: *string2*.

Explanation: XML Request is serialized to string before passed to JMS publisher.

Action: Check message: probably not XML wellformed or violating SPML schema.

ERR(JMS005): Exception when publishing SPML request *string1*: *string2*.

Explanation: JMS publisher returned exception when publishing SPML request.

Action: Check environment: JMS provider available?

INF(JMS006): Publishing request (id = *string1*) to *string2*.

Explanation: JMS publisher publishes the SPML request with this ID.

Action: NONE.

FTL(JMS007): Exception when creating topic of JMS connector: *string1*.

Explanation: Error when creating topic of the connector.

Action: Check your messaging environment: message server started and available, event activity of the workflow, etc.

FTL(JMS008): Exception when creating publisher of JMS connector: *string1*.

Explanation: Error when creating publisher of the connector.

Action: Check your messaging environment: message server started and available, event activity of the workflow, etc.

FTL(JMS009): Exception when initializing context of JMS connector: *string1*.

Explanation: Error when initializing the messaging context of the connector.

Action: Check your messaging environment: message server started and available, event activity of the workflow, etc.

FTL(JMS010): Exception when creating session of JMS connector: *string1*.

Explanation: Error when creating publisher of the connector.

Action: Check your messaging environment: message server started and available, event activity of the workflow, etc.

16.3.14. Logging Messages (LCO)

This topic describes the logging messages returned by the LDAP Connector for DirX Identity Connector Framework.

INF(LCO225): Bind to LDAP server '*string1*' with port '*string2*' as user '*string3*' successful.

Explanation: The connection to the LDAP server has been established successfully.

Action: None.

INF(LCO226): Reconnect to LDAP Server '*string1*' with port '*string2*' as user '*string3*' successful.

Explanation: The connection to the LDAP server has been re-established successfully.

Action: None.

INF(LCO227): Reconnect to LDAP Server: *string1* with port: *string2* authentication successful.

Explanation: The connection to the LDAP server has been re-authenticated successfully.

Action: None.

INF(LCO230): Added entry: *string1* successfully.

Explanation: The current entry has been added successfully.

Action: None.

INF(LCO231): Modified entry: *string1* successfully.

Explanation: The current entry has been modified successfully.

Action: None.

INF(LCO232): Deleted entry: *string1* successfully.

Explanation: The current entry has been deleted successfully.

Action: None.

INF(LCO233): Filtered out illegal xml character with byte value:_string1_ at position:_string2_ from LDAP exception message:_string3_.

Explanation: The illegal xml char was filtered out from response message to avoid marshaller problems.

Action: None.

INF(LCO234): Paged results cookie returned by the server: *string1*.

Explanation: This cookie is taken for getting the next page.

Action: None.

INF(LCO235): The LDAP PagedResults control contains no cookie. Last page reached.

Explanation: Since no paged results cookie exists the search is finished.

Action: None.

INF(LCO236): Delta cookie returned by the server: *string1*.

Explanation: This cookie returned in the spml search response can be used for the next delta search.

Action: None.

INF(LCO237): Moved entry from *string1* to *string2* successfully.

Explanation: The current entry has been moved successfully to the described destination.

Action: None.

INF(LCO238): Renamed entry from *string1* to *string2* successfully.

Explanation: The current entry has been renamed successfully to the new rdn.

Action: None.

INF(LCO239): Starting paged search with page size=*string1*, base=*string2*, filter=*string3* and sort attr=*string4*.

Explanation: Starting paged search with the specified parameters.

Action: None.

INF(LCO240): Starting search with base=*string1*, filter=*string2* and sort attr=*string3*.

Explanation: Starting search with the specified parameters.

Action: None.

INF(LCO241): No user is defined in connection. Trying to establish an anonymous bind.

Explanation: Because no user is specified, an anonymous bind is tried.

Action: None.

INF(LCO242): Modify entry *string1* failed when trying to add a value that already existed. Message: *string2*.

Explanation: Modifying the entry failed because a value already existed.

Action: None.

INF(LCO243): Modify entry *string1* failed when trying to add a DN_value that already existed

in the attribute value list. Message: *string2*.

Explanation: Modifying the entry failed because a DN_value already existed.

Action: None.

INF(LCO244): LDAP get next entry from search result returned referrals: *string1*. Message: *string2*.

Explanation: The entry from the search result is from a foreign domain.

Action: None.

INF(LCO245): Modify entry *string1* failed when trying to delete a value that does not exist. Message: *string2*.

Explanation: Modifying the entry failed because a value to delete does not exist.

Action: None.

INF(LCO246): Modify entry *string1* failed because the server is unwilling to perform. Message: *string2*.

Explanation: Modifying the entry failed because the server is unwilling to perform. If a reason can be determined an additional message is logged.

Action: None.

INF(LCO247): Modify entry *string1* failed when trying to add a DN_value that does not exist as object in the directory. Message: *string2*.

Explanation: Modifying the entry failed because trying to add a DN_value that does not exist in the directory.

Action: None.

INF(LCO248): Performing single modifications for entry *string1*. Values with failed operations are logged.

Explanation: Performing single modifications for all attributes and all values of an entry.

Action: None.

INF(LCO249): Setting LDAP Relaxed Update Control as constraints failed. Error Message: *string1*.

Explanation: Setting LDAP Relaxed Update Control as constraints failed. Control is not used.

Action: None.

INF(LCO250): The LDAP Server supports the Relaxed Update Control. It is used now for subsequent modify operations.

Explanation: The Relaxed Update Control is used on modify operations.

Action: None.

INF(LCO251): The LDAP Server does not support the Relaxed Update Control. Usual modifications are done.

Explanation: The Relaxed Update Control is not supported.

Action: None.

INF(LCO252): Replace operation for naming attribute is not allowed. Replacing of *string1* values are skipped.

Explanation: Replacing naming attribute values is not allowed. This modification is skipped and not passed to the server.

Action: None.

INF(LCO253): Delete operation for the naming value of attribute *string2* is not allowed. Hence deleting *string2* value = *string1* is not tried.

Explanation: Deleting naming attribute value is not allowed. This modification is skipped and not passed to the server.

Action: None.

INF(LCO254): Add operation for the naming value of attribute *string2* is not allowed. Hence adding *string2* value = *string1* is not tried.

Explanation: Adding naming attribute value is not allowed. This modification is skipped and not passed to the server.

Action: None.

INF(LCO255): Number of entries returned: *string1*.

Explanation: Lists the number of found entries.

Action: None.

INF(LCO256): Setting LDAP Update Control as constraints failed. Error Message: *string1*.

Explanation: Setting LDAP Update Control as constraints failed. Control is not used.

Action: None.

INF(LCO257): The LDAP Server supports the PolicyHints Update Control. It is used now on modify operations.

Explanation: The PolicyHints Update Control is used on modify operations.

Action: None.

INF(LCO258): Reason for unwilling to modify entry *string1*: Error code=*string2*. Related error message=*string3*.

Explanation: The detailed error code extracted from the unwilling to perform message and the related error message is logged.

Action: None.

INF(LCO259): The LDAP server supports the Session Tracking Control. It is used now for all operations on this LDAP connection.

Explanation: The session tracking control is supported and set on this LDAP connection.

Action: None.

INF(LCO260): Trying to activate session tracking failed with Exception: *string1*. The LDAP connection just works without session tracking.

Explanation: The session tracking could not be activated on this LDAP connection.

Action: None.

INF(LCO261): The Session Tracking Control is used now for all operations on this LDAP connection.

Explanation: The session tracking control is set on this LDAP connection.

Action: None.

INF(LCO262): Deleted entry: *string1* with child objects successfully.

Explanation: The current entry with all child objects has been deleted successfully.

Action: None.

INF(LCO263): The Active Directory LDAP Server supports the Delete Subtree Control. It is used now on delete operations for non-leaf objects.

Explanation: The Delete Subtree Control is used on delete operations for non-leaf objects.

Action: None.

WAR(LCO410): Add entry *string1* failed: *string2*.

Explanation: Adding the entry failed.

Action: Check the LDAP error code.

WAR(LCO411): Modify entry *string1* failed: *string2*.

Explanation: Modifying the entry failed.

Action: Check the LDAP error code.

WAR(LCO412): Delete entry *string1* failed: *string2*.

Explanation: Deleting the entry failed.

Action: Check the LDAP error code.

WAR(LCO413): Cannot access version information from jar file.

Explanation: Accessing version information from manifest failed.

Action: Inform DirX Identity technical support about the problem.

WAR(LCO414): LDAP get next entry from search result failed. Message: *string1*

Explanation: LDAP get next entry from search result failed.

Action: Check the exception message.

WAR(LCO417): The LDAP DirSync control contains no cookie.

Explanation: Since no delta cookie exists the next search will be a full search.

Action: Do a full search or inform DirX Identity technical support about the problem.

WAR(LCO418): No port is defined in connection. Working with default port 389.

Explanation: Add a port="<port>" property to the connection tag.

Action: Correct the configuration file as described in explanation. Note that the system might build the configuration file automatically from a template that contains references. In this case you must care for proper reference resolution.

WAR(LCO419): An LDAP return code for add, modify, rename or delete was classified as temporary. See subsequent logging.

Explanation: An LDAP return code was classified as temporary.

Action: Check the subsequent detailed logging information.

WAR(LCO420): An LDAP exception occured when reading the supported LDAP controls. Exception message: *string1*.

Explanation: An LDAP exception occured when reading the supported LDAP controls from the LDAP Server.

Action: Nothing to do. The Relaxed LDAP control is just not used on modify operations.

WAR(LCO421): An LDAP exception occured when reading the LDAP controls to check if control=*string1* is supported by server. LDAP exception message:_string2_.

Explanation: Since no LDAP controls are returned by the server it cannot be determined which ones are supported.

Action: Nothing to do. The LDAP control is just not used.

ERR(LCO530): Cannot return LDAP connection *string1* to connection pool *string2*: *string3*.

Explanation: The LDAP connection pool does not accept the returned connection.

Action: Check the detailed message.

ERR(LCO531): Pool specification for connection pool *string1* not accepted by the pool manager: *string2*.

Explanation: The pool manager does not accept the pool specification. Use of another connector type may help.

Action: Check the detailed message.

ERR(LCO570): Add entry *string1* failed: *string2*.

Explanation: Adding the entry failed.

Action: Check the LDAP error code.

ERR(LCO571): Modify entry *string1* failed: *string2*.

Explanation: Modifying the entry failed.

Action: Check the LDAP error code.

ERR(LCO572): Delete entry *string1* failed: *string2*.

Explanation: Deleting the entry failed.

Action: Check the LDAP error code.

ERR(LCO573): Rename entry from *string1* to *string2* failed with error: *string3*.

Explanation: Renaming the entry failed.

Action: Check the LDAP error code.

ERR(LCO574): Incorrect LDAP Search Filter: *string1*.

Explanation: LDAP Search Filter is incorrect.

Action: Check the LDAP search filter.

ERR(LCO575): LDAP search with base: *string1*, scope: *string2* and filter: *string3* failed. Error code: *string4*, error msg: *string5*.

Explanation: LDAP search failed.

Action: Check the LDAP error code.

ERR(LCO576): Method *string1* not supported with paged read.

Explanation: When using paged read you have to use the enumeration object to get the search result entries.

Action: Use method enumerateSearchResultEntry().

ERR(LCO577): LDAP server *string1* is temporary unavailable.

Explanation: The server is unavailable at the moment.

Action: Check host/port and if server is up.

ERR(LCO578): LDAP paged search with base: *string1*, scope: *string2* and filter: *string3* failed. Error code: *string4*, error msg: *string5*.

Explanation: LDAP paged search failed.

Action: Check the LDAP error code.

ERR(LCO581): Search result contained no LDAP controls. Processing of paged search aborted.

Explanation: Since no LDAP controls are returned by the server the paged search must be aborted.

Action: Do a full search or inform DirX Identity technical support about the problem.

ERR(LCO582): Search result contained no LDAP PagedResults control. Processing of paged search aborted.

Explanation: Since no LDAP PagedResults control is returned by the server the paged search must be aborted.

Action: Do a full search or inform DirX Identity technical support about the problem.

ERR(LCO583): Search result contained no LDAP controls. Processing of delta search aborted.

Explanation: Since no LDAP controls are returned by the server the delta search must be aborted.

Action: Do a full search or inform DirX Identity technical support about the problem.

ERR(LCO584): Search result contained no LDAP DirSync control. Processing of delta search aborted.

Explanation: Since no LDAP DirSync control is returned by the server the delta search must be aborted.

Action: Do a full search or inform DirX Identity technical support about the problem.

ERR(LCO585): Move entry from *string1* to *string2* failed with error: *string3*.

Explanation: Moving the entry failed.

Action: Check the LDAP error code.

ERR(LCO586): Delete entry *string1* failed: *string2*. Delete LDAP subtree control is not supported.

Explanation: Deleting the entry failed with NOT_ALLOWED_ON_NONLEAF objects.

Action: Check the LDAP error code.

ERR(LCO587): Delete entry *string1* with child objects failed with exception: *string2*.

Explanation: Deleting the entry with child objects failed.

Action: Check the LDAP exception message.

FTL(LCO907): Exception in connector during open: *string1*.

Explanation: An exception occurred during open of the connector instance.

Action: Check the exception message or inform DirX Identity technical support about the problem.

FTL(LCO908): No connector type specified.

Explanation: Add a type="<connector type>" property to the connection tag.

Action: Correct the configuration file as described in explanation.

FTL(LCO910): Cannot determine address of LDAP server, no connection defined for the connector.

Explanation: Add a <connection> tag to the definition of the connector.

Action: Correct the configuration file as described in explanation. Note that the system might build the configuration file automatically from a template that contains references. In this case you must care for proper reference resolution.

FTL(LCO911): Cannot bind to LDAP server, no server defined in connection.

Explanation: Add a server="<server>" property to the connection tag with <server> containing the name of the LDAP server.

Action: Correct the configuration file as described in explanation. Note that the system might build the configuration file automatically from a template that contains references. In this case you must care for proper reference resolution.

FTL(LCO913): Cannot bind to LDAP server, a user but no password is defined in connection.

Explanation: Add a password="<password>" property to the connection tag with <password> containing the users password.

Action: Correct the configuration file as described in explanation. Note that the system might build the configuration file automatically from a template that contains references. In this case you must care for proper reference resolution.

FTL(LCO914): Bind to the LDAP server *string1* with port *string2* as user *string3* failed: *string4*.

Explanation: Binding to the LDAP server failed.

Action: Check the connection parameters and if server is running.

FTL(LCO915): Disconnect from LDAP server *string1* failed: *string2*.

Explanation: Disconnecting from the LDAP server failed.

Action: Check the LDAP error code.

FTL(LCO916): Connection to LDAP server *string1* is lost or open method of LDAP Connector was not called.

Explanation: Server might not be running or open method of connector has not been called.

Action: Check if LDAP server is running.

FTL(LCO917): Exception while accessing version information from jar file.

Explanation: Accessing version information from manifest failed.

Action: Inform DirX Identity technical support about the problem.

FTL(LCO918): An unsupported encoding exception occurred with exception message: *string1*.

Explanation: An exception occurred with the given exception message.

Action: Check the exception message text.

FTL(LCO933): Exception occurred with exception message: *string1*.

Explanation: An exception occurred with the given exception message.

Action: Check the exception message text.

FTL(LCO934): Exception in function DoLdapSearchAndBuildSre. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(LCO935): Exception in function DoLdapSearchPagedAndBuildSre. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(LCO936): Exception occured while converting ldap search result entry *string1* to spml search result entry. Exception message: *string2*.

Explanation: An exception occurred by converting ldap search result entry to spml search result entry.

Action: Check the exception message.

FTL(LCO937): Client SSL configuration requires the SSL flag to be set too.

Explanation: SSL flag and Client SSL flag need both to be set in the BIND profile.

Action: Set the SSL flag in the BIND profile.

FTL(LCO938): For Client SSL at least one of the parameters is not set: Key Store, Key Store Password, Trust Store, Trust Store Password.

Explanation: At least one of the required parameters for Client SSL is not set in the BIND profile.

Action: Set the missing parameters for Client SSL in the BIND profile.

16.3.15. Logging Messages (O365)

This topic describes the logging messages returned by the Office 365 connector.

INF(O365200): *string1*

Explanation: Generic info message.

Action: None.

INF(O365201): Search returned *string1* entries - *string2* additional entries are available.

Explanation: Returned entries and check if additional are available (nextPageUrl on response).

Action: None.

WAR(O365400): The connector could not get information about version because an exception has occurred: *string1*.

Explanation: The connector jar file is probably corrupted.

Action: Check integrity of the jar file containing the Office 365 connector.

WAR(O365401): The connector could not get identifier of the search result for an object type *string1*.

Explanation: The REST API was probably changed or does not work correctly.

Action: Contact the DirX Identity support.

ERR(O365500): Invalid or empty HTTP response.

Explanation: The connector could not get valid HTTP response on its request.

Action: Check the local network connection and availability of the REST API endpoint.

ERR(O365501): Invalid JSON format returned, could not locate tag *string1*.

Explanation: The connector could not parse returned JSON response due to missing tag.

Action: Contact the DirX Identity support.

ERR(O365502): Invalid JSON format returned, invalid type *string1* expecting *string2*.

Explanation: The connector could not parse returned JSON response due to unexpected data type.

Action: Contact the DirX Identity support.

FTL(O365600): Could not use configured property *string1*, an exception has occurred: *string2*.

Explanation: The connector configuration is invalid.

Action: The connector requires valid configuration properties. Check and correct connector configuration for described property.

FTL(O365601): The connector configuration properties do not contain mandatory property *string1*.

Explanation: The connector configuration is invalid.

Action: Correct the configuration and add missing configuration property.

FTL(O365602): Could not create JSON representation of OAuth request header, an exception has occurred: *string1*.

Explanation: The connector configuration is invalid or an internal error has occurred.

Action: Contact DirX Identity support.

FTL(O365603): Could not obtain OAuth response header for Office 365 API.

Explanation: The connector could not get OAuth header data required for access to Office 365 API.

Action: Check availability of the OAuth server, check status of your Office 365 licenses.

FTL(O365604): An exception has occurred while sending request to OAuth server: *string1*.

Explanation: The connector could not get OAuth header data required for access to Office 365 API due to an exception.

Action: Check availability of the OAuth server, check status of your Office 365 licenses, check other logged errors.

FTL(O365605): OAuth server returned an invalid response: *string1*.

Explanation: The connector could not get OAuth header data required for access to Office 365 API due to an invalid response.

Action: Check availability of the OAuth server, check status of your Office 365 licenses.

FTL(O365606): Internal error has occurred an invalid operation was invoked: *string1*.

Explanation: The connector does not support add, modify or delete operation on all object types. A mapping error caused invocation of an improper operation.

Action: Check mappings of the operational attributes, check also type of object being provisioned and check if the user hooks are correctly invoked.

16.3.16. Logging Messages (ODB)

This topic describes the logging messages returned by the DirX ODBC Agent.

FTL(ODB0001): At *string2*: Fatal ODBC error: *string3*

Explanation: Fatal ODBC error. Should never happen.

Action: Check ODBC status and version. Inform DirX Identity technical support about the problem.

ERR(ODB0002): At *string2*: Error in importer record: cannot process: *string3*

Explanation: Problem with processing import record.

Action: Check record against database, and correct.

ERR(ODB0003): At *string2*: Error in procedure record: cannot action: *string3*

Explanation: Problem with actioning procedure record.

Action: Check record (e.g. against database contents), and correct. Perhaps execute the record manually.

ERR(ODB0004): At *string2*: Cannot action record: *string3*

Explanation: Unidentified record error - cannot action. Shouldn't happen.

Action: Inform DirX Identity technical support about the problem.

FTL(ODB0005): At *string2*: Fatal command line error: *string3*

Explanation: Fatal command-line error.

Action: Correct the command-line and re-try.

ERR(ODB0006): At *string2*: Non-fatal schema error: *string3*

Explanation: Non-fatal schema error.

Action: Check schema configuration and correct if necessary.

FTL(ODB0007): At *string2*: Fatal password configuration error: *string3*

Explanation: Fatal password configuration error.

Action: Correct password in configuration.

FTL(ODB0008): At *string2*: Fatal configuration file error: *string3*

Explanation: Fatal configuration file error.

Action: Correct error in configuration file

FTL(ODB0009): At *string2*: Fatal unknown configuration file error: *string3*

Explanation: Fatal configuration file error. Shouldn't happen

Action: Inform DirX Identity technical support about the problem.

FTL(ODB0010): At *string2*: Fatal configuration file consistency error: *string3*

Explanation: Fatal configuration file consistency error - overlapping attribute lists.

Action: Correct configuration file.

INF(ODB0011): At *string2*: Database connection information: *string3*

Explanation: Database connection information.

Action: None.

FTL(ODB0012): At *string2*: Fatal output file write error: *string3*

Explanation: Error in writing to an output file.

Action: Check filestore structure and permissions.

FTL(ODB0013): At *string2*: Fatal sort file error: *string3*

Explanation: System error with sort file generation. Shouldn't happen.

Action: Check sort parameters, and try again. Inform DirX Identity technical support about the problem.

FTL(ODB0014): At *string2*: Fatal sort key error in database configuration: *string3*

Explanation: Database error: multiple key instances found in sort file. This indicates that the specified sort keys do not give unique entries.

Action: Change configuration to give unique keys.

FTL(ODB0015): At *string2*: Fatal procedures configuration error: *string3*

Explanation: Procedures configuration error with unsupported return SQL type.

Action: Check procedures configuration and correct return SQL type.

FTL(ODB0016): At *string2*: Unidentified fatal error: *string3*

Explanation: Unidentified fatal error. Shouldn't happen.

Action: Inform DirX Identity technical support about the problem.

FTL(ODB0017): At *string2*: Invalid import file: *string3*

Explanation: Fatal error in import file.

Action: Correct import file.

FTL(ODB0018): At *string2*: Fatal ODBC configuration error: *string3*

Explanation: Fatal ODBC configuration error.

Action: Correct ODBC configuration in configuration file.

FTL(ODB0019): At *string2*: ODBC driver connect failure: *string3*

Explanation: Fatal ODBC connection error.

Action: Correct database connection attributes.

FTL(ODB0020): At *string2*: Fatal sorter configuration error: *string3*

Explanation: Fatal sorter configuration error.

Action: Check and correct sort characteristics in the configuration file.

FTL(ODB0021): At *string2*: ODBC failure: *string3*

Explanation: Fatal ODBC operation error.

Action: Check ODBC version or configuration. Inform DirX Identity technical support about the problem.

ERR(ODB0022): At *string2*: Referential integrity failure for record: *string3*

Explanation: Referential integrity rules prevented import of the record.

Action: Check referential integrity and alter configuration if appropriate.

INF(ODB0023): At *string2*: ODBC version and other information: *string3*

Explanation: information about ODBC/database.

Action: None.

WAR(ODB0024): At *string2*: Import referential integrity warning: *string3*

Explanation: During import, a delete operation may have encountered a referential integrity problem, and may not have performed as expected.

Action: Consider returned information and check result.

WAR(ODB0025): At *string2*: ODBC import or procedure warning: *string3*

Explanation: During import or procedure call, the specified SQL operation has returned "success with info" and may not have performed as expected.

Action: Consider returned information and check result.

WAR(ODB0026): At *string2*: ODBC preparation warning: *string3*

Explanation: During preparation of the agent, the specified SQL operation has returned "success with info" and may not have been correctly set up.

Action: Consider returned information and check results.

WAR(ODB0027): At *string2*: Unexpected ODBC warning: *string3*

Explanation: Unexpected ODBC warning. Should never happen.

Action: Inform DirX Identity technical support about the problem.

INF(ODB0028): At *string2*: Record info: *string3*

Explanation: Record information.

Action: None.

WAR(ODB0029): At *string2*: Record warning: *string3*

Explanation: The record is processable for import or by procedure, but the operation may have an unexpected result, for the reason mentioned. This could be the result of configuration or data issues, but may be deliberate or not significant.

Action: Check import information against the result and correct if necessary.

FTL(ODB0030): At *string2*: Fatal procedure error: *string3*

Explanation: Fatal SQL error when executing a procedure. Should never happen.

Action: Inform DirX Identity technical support about the problem.

INF(ODB0031): At *string2*: SQL statement: *string3*

Explanation: SQL statement as generated by the agent.

Action: None.

INF(ODB0032): At *string2*: Summary: *string3*

Explanation: Information about the processed result(s)

Action: None.

INF(ODB0033): At *string2*: Requested column info: *string3*

Explanation: Information about database column schema

Action: None.

INF(ODB0034): At *string2*: Reference file info: *string3*

Explanation: Reference file report and information.

Action: None.

ERR(ODB0035): At *string2*: No such reference file: *string3*

Explanation: Supplied reference file not found.

Action: Check supplied reference file name.

ERR(ODB0036): At *string2*: Mismatched reference file: *string3*

Explanation: The supplied reference file characteristics do not correspond to the current operation.

Action: Correct supplied reference file name.

ERR(ODB0037): At *string2*: Severe reference file problem: *string3*

Explanation: Severe problem found in processing specified reference file.

Action: Check filestore, reference paths, temporary reference paths, permissions

FTL(ODB0038): At *string2*: Fatal schema error: *string3*

Explanation: Fatal schema problem.

Action: Correct configured schema.

WAR(ODB0039): At *string2*: Non-fatal schema problem: *string3*

Explanation: Non-fatal schema problem, possibly producing unexpected results.

Action: Check configured schema.

INF(ODB0040): At *string2*: Import statistics: *string3*

Explanation: Agent statistics.

Action: None.

WAR(ODB0041): At *string2*: Config file warning: *string3*

Explanation: Non-fatal configuration-file problem.

Action: Check configuration file and correct if necessary.

FTL(ODB0042): At *string2*: Fatal security system error: *string3*

Explanation: Fatal system error in security configuration.

Action: Inform DirX Identity technical support about the problem.

FTL(ODB0043): At *string2*: Fatal security error: *string3*

Explanation: Fatal but potentially correctable error in security configuration.

Action: Check configuration and correct

INF(ODB0044): At *string2*: SQL statement unicode: *string3*

Explanation: SQL Unicode statement information

Action: None.

FTL(ODB0045): At *string2*: Unicode UTF8 conversion: *string3*

Explanation: Bad UTF8 in column-name conversion.

Action: Correct column-name configuration.

FTL(ODB0046): At *string2*: Fatal procedure configuration error: *string3*

Explanation: Fatal procedure configuration error.

Action: Correct procedure configuration.

INF(ODB0047): At *string2*: Procedure info: *string3*

Explanation: Procedure information.

Action: None.

ERR(ODB0048): At *string2*: Unknown error: *string3*

Explanation: Unknown error.

Action: Check detail. Inform DirX Identity technical support about the problem.

INF(ODB0049): At *string2*: Version info: *string3*

Explanation: Version information.

Action: None.

INF(ODB0050): *string2*

Explanation: Trace information.

Action: None.

16.3.17. Logging Messages (OICF)

This topic describes the logging messages returned by the DirX Identity OpenICF connector.

INF(OICF200): Connecting to the OpenICF server running on *string1*:_string2_ (SSL: *string3*).

Explanation: The connector tries to establish connection with OpenICF server.

Action: None.

INF(OICF201): Reading the configuration information for remote connector (*string1*, *string2*, *string3*).

Explanation: The connector tries to read configuration information of the remote OpenICF connector.

Action: None.

INF(OICF202): Analyzing the configuration information for remote connector:

Explanation: The connector tries to analyze the configuration information of the remote OpenICF connector.

Action: None.

INF(OICF203): *string1* property *string2* of type *string3* is supported.

Explanation: The connector tries to analyze the configuration information of the remote OpenICF connector.

Action: None.

INF(OICF204): Analyzing the attribute configuration information for *string1*:

Explanation: The connector tries to analyze the attribute configuration information of the selected object type.

Action: None.

INF(OICF205): *string1 string2* value property *string3* of type *string4* is supported.

Explanation: The connector tries to analyze the attribute configuration information of the selected object type.

Action: None.

INF(OICF206): *string1* received an SPML *string2* with id *string3*.

Explanation: The connector received a new request.

Action: None.

INF(OICF207): *string1* returned an SPML *string2* with id *string3*.

Explanation: The connector returned a response.

Action: None.

INF(OICF208): The attribute *string1* was not found in attribute configuration schema for *string2* entry type of the remote connector, using default type *string3*.

Explanation: The connector could not reveal expected type of the attribute in the incoming SPML request and uses default type.

Action: None if the remote server can work correctly with the attribute type. Remove the attribute mapping if the connector fails using it.

WAR(OICF400): The server configuration contains invalid port value, using default value *string1*.

Explanation: The connector configuration is invalid.

Action: Correct the configuration property for server timeout.

WAR(OICF401): The connector could not get information about version because *string1*.

Explanation: The connector jar file is probably corrupted.

Action: Check integrity of the jar file containing the OpenICF connector.

ERR(OICF500): An exception occurred when processing SPML *string1* with id *string2*: *string3*.

Explanation: The connector could not successfully process the incoming request.

Action: Check the exception message and try to resolve it. Check also OpenICF connector server log files for more details.

ERR(OICF501): An invalid SPML *string1* with id *string2* was detected, missing *string3*.

Explanation: The connector could not process incoming SPML request because it is invalid.

Action: Check the attribute mapping and workflow configuration.

ERR(OICF502): Could not rename *string1* entry from *string2* to *string3*.

Explanation: The connector could not rename entry.

Action: Verify that the remote connector supports the rename operation. Check also OpenICF connector server log files for more details.

ERR(OICF503): Could not rename *string1* entry from *string2* to *string3* due to an exception: *string4*.

Explanation: The connector could not rename entry.

Action: Verify that the remote connector supports the rename operation. Check the exception message and try to resolve it. Check also OpenICF connector server log files for more details.

FTL(OICF601): Could not open connector due to the invalid number of connection elements.

Explanation: The connector configuration is invalid.

Action: The connector requires two connection elements - one of type OpenIcfServer and one of type OpenIcfConnector. Check and correct connector configuration.

FTL(OICF602): Could not open connector due to an invalid type of connection elements.

Explanation: The connector configuration is invalid.

Action: The connector requires two connection elements - one of type OpenIcfServer and one of type OpenIcfConnector. Check and correct connector configuration.

FTL(OICF603): The *string1* configuration properties do not contain mandatory property *string2*.

Explanation: The connector configuration is invalid.

Action: Correct the configuration and add missing configuration property.

FTL(OICF604): The server configuration contains invalid port value.

Explanation: The connector configuration is invalid.

Action: Correct the configuration property for server port.

FTL(OICF605): An exception was encountered while opening the connector: *string1*.

Explanation: The connector failed during initialization phase.

Action: Check the exception message and try to resolve it. Check also OpenICF connector server log files for more details.

FTL(OICF606): Could not locate OpenICF connector bundle with the specification *string1* on the configured OpenICF connector server running on *string2*:_string3_.

Explanation: The remote OpenICF connector server could not locate given connector bundle.

Action: Check if the desired OpenICF connector bundle is correctly deployed on selected OpenICF server. Check also OpenICF connector server log files for more details.

16.3.18. Logging Messages (PVC)

This topic describes the logging messages returned by the DirX Identity link checker agent.

INF(PVC201): Program '*string1*', Version '*string2*' of '*string3*' started ***.

Explanation: Start message of the link checker agent.

Action: None.

INF(PVC202): Program '*string1*' ended with *string2* fatalError(s), *string3* error(s), and *string4* warning(s).

Explanation: Informs the user that the program has ended correctly. Shows an error statistics.

Action: None.

ERR(PVC211): ***** Rule to Check: *string1*
Rule Error: *string2* .

Explanation: This rule might cause problems.

Action: Do an edit and save of this rule via the DirX Identity Manager.

FTL(PVC901): Usage: ProvisioningRuleChecker <Configuration File>.

Explanation: The program was called with the wrong number of arguments.

Action: unlikely to occur. Call the checker with correct number of arguments.

FTL(PVC902): Configuration file '*string1*' does not exist.

Explanation: The configuration file passed to the agent does not exist.

Action: Correct the file's pathname or create a new configuration file at the given pathname.

FTL(PVC903): Error(s) while processing configuration file '*string1*': *string2*.

Explanation: Error(s) occurred while processing the configuration file.

Action: Check the details, correct errors and re-run checker.

FTL(PVC905): Cannot initialize storage: *string1*.

Explanation: initialize of storage failed.

Action: Check the details, correct errors and re-run checker.

FTL(PVC907): No trace file defined in the agent.ini file.

Explanation: The definition for trace.filename is missing.

Action: Add a line trace.filename=<pathname> in agent.ini where <pathname is the absolute pathname to the trace file.

FTL(PVC908): Cannot open trace file '*string1*' for writing: *string2*.

Explanation: The trace file defined in the agent's ini file cannot be opened for writing.

Action: Check pathname and create missing directories. Check file creation ACLs.

16.3.19. Logging Messages (RACF)

This topic describes the logging messages returned by the RACF Connector.

LOG(RACF201): Object found: *string1*

Explanation: Object found in LDAP.

Action: None.

LOG(RACF202): Object not found: *string1*

Explanation: Object not found in LDAP.

Action: None.

LOG(RACF203): Object successfully created: *string1*

Explanation: Object CONNECT successfully created in LDAP.

Action: None.

LOG(RACF204): Object already exists: *string1*

Explanation: Object CONNECT already exists in LDAP.

Action: None.

LOG(RACF205): Object successfully deleted: *string1*

Explanation: Object CONNECT successfully deleted in LDAP.

Action: None.

LOG(RACF206): Object does not exists: *string1*

Explanation: Object does not exists in LDAP.

Action: None.

WAR(RACF401): Cannot access version information from jar file.

Explanation: Accessing version information from manifest failed.

Action: Inform the DirX Identity support group.

WAR(RACF402): The REPLACE operation is not supported. Only ADD and DELETE are allowed.

Explanation: Only ADD and DELETE operations are allowed for membership.

Action: Check the SPML request.

ERR(RACF501): RACF server '*string1*' is temporary unavailable.

Explanation: The server is unavailable at the moment.

Action: Check host/port and if server is up.

ERR(RACF502): Identifier missing in request.

Explanation: Identifier component has not been found in the request.

Action: Check the incoming SPML request.

ERR(RACF503): Password missing in request.

Explanation: Password component has not been found in the request.

Action: Check the incoming SPML request.

FTL(RACF901): Bind to the RACF server for user *string1* failed: *string2*.

Explanation: Connecting to the RACF server failed.

Action: Check the error reason.

FTL(RACF902): Disconnect from ldap server for user *string1* failed: *string2*.

Explanation: Disconnecting from the RACF server failed.

Action: Check the error reason.

FTL(RACF903): Exception while accessing version information from jar file.

Explanation: Accessing version information from manifest failed.

Action: Inform the DirX Identity support group.

16.3.20. Logging Messages (RWC)

This topic describes the logging messages returned by the DirX Identity Request Workflow Connector.

INF(RWC001): Created RequestWorkflow '*string1*' with ID='*string2*' for subject='*string3*'.

Explanation: The request workflow connector created a new request workflow with the given account / group as the subject.

Action: None.

WAR(RWC101): *string1* during serializing subject.request: *string2*.

Explanation: The request workflow connector tried to write a debug log and needed to marshall the order intended to become the subject of the new workflow. This failed.

Action: Check the trigger event: probably it is not well-formed XML or not representing a SPML request.

WAR(RWC102): Could not read the object type from the operational attributes of the request, using default value '*string1*'.

Explanation: The RequestWorkflow requests need the information about object type set in the operational attributes.

Action: Check the connectivity workflow configuration of the source channel, the object type information must exist.

ERR(RWC201): *string1* when creating RequestWorkflow '*string2*' for subject='*string3*': *string4*.

Explanation: The request workflow connector created a new request workflow with the given account / group as the subject, but the Soap request returned with a failure.

Action: Check if the workflow has been created nevertheless.

ERR(RWC202): No endpoint (URL) configured; use default instead: *string1*.

Explanation: A SOAP client requires address and port of the remote web service. The connector continues with the default URL.

Action: Add URL or ssl, server, port and path to configuration.

ERR(RWC203): Exception while opening RequestWorkflowConnector: *string1*.

Explanation: An exception occurred while reading the configuration options of the RequestWorkflow connector.

Action: Check the configuration of the Connector and availability of necessary libraries.

ERR(RWC204): *string1* when creating request workflow service for endpoint=*string2*, user=*string3*: *string4*.

Explanation: The RequestWorkflow connector tried to create a connection to the RequestWorkflow Service of the IDS-J server.

Action: Check URL, user and password configured for the Connector and availability of necessary libraries.

ERR(RWC205): Could not create new request workflow instance on endpoint '*string1*'.

Explanation: The request workflow instance could not be created.

Action: Check the previously logged errors.

16.3.21. Logging Messages (SAMP)

This topic describes the logging messages returned by the Sample connector for DirX Identity Connector Framework.

FTL(SAMPCONN001): Cannot determine filename of *string1* file, no connection defined for the connector.

Explanation: The connector needs a filename to read responses from.

Action: Add a <connection> tag to the definition of the connector.

ERR(SAMPCONN002): Exception while unmarshalling responses from file: '*string1*'.

Explanation: Obviously SPML response in file is not well-formed or not valid SPML.

Action: Check responses in input file!

INF(SAMPCONN003): *string1*: No response found for request *string2* with id: *string3*.

Explanation: Searching response for incoming request failed. Now using default response.

Action: Add missing response to response file!

16.3.22. Logging Messages (SIPORT)

This topic describes the logging messages returned by the SIPORT connector filter.

WAR(SIPORT400): The filter could not process configuration for attribute mapping line: *string1*, ignoring it.

Explanation: The filter configuration is invalid, the line will be ignored.

Action: Delete or correct the invalid line to the syntax '<attributeNameInTheSourceTable> <attributeNameInTheImportTable>'.

WAR(SIPORT401): No mapping for the attribute *string1* was defined for the import table, using original name.

Explanation: The connector filter could not convert the given attribute name to the name used in the import table because of missing mapping definition.

Action: Add the mapping in the filter configuration for the given attribute if the JDBC connector cannot use it for the import table.

WAR(SIPORT402): The attribute *string1* cannot be modified because of an invalid operation type, only the replace operation is supported, ignoring it.

Explanation: The connector filter could not process a modification for the given attribute because of an invalid operation type.

Action: Correct the mapping for the named attribute, use only the replace operation.

WAR(SIPORT403): The attribute *string1* is defined as a key attribute for the import table and cannot be changed, ignoring it.

Explanation: The connector filter could not process a modification for the attribute defined as the key for the import table.

Action: Correct the mapping for the named attribute, this attribute is cannot be changed.

WAR(SIPORT500): The incoming request does not contain any attributes or modifications.

Explanation: The workflow configuration is invalid since a request without any usable data was issued.

Action: Try to correct the workflow configuration. If the problem persists, inform DirX Identity technical support about the problem.

WAR(SIPORT501): The incoming request does not contain the key attribute within the operational attributes.

Explanation: The workflow configuration is invalid since a request without the key attribute within the operational attributes was issued.

Action: Correct the workflow configuration. The configured key attribute must be part of the operational attributes.

WAR(SIPORT502): The JDBC connector returned no response.

Explanation: The JDBC returned no response, the filter cannot transform it to the correct type.

Action: Check the JDBC connector errors. Try to correct the workflow configuration. If the problem persists, inform DirX Identity technical support about the problem.

16.3.23. Logging Messages (SLSF)

This topic describes the logging messages returned by the Salesforce Connector.

INF(SLSF200): *string1*.

Explanation: Generic informational message.

Action: None.

WAR(SLSF400): No version information available in 'dxmSalesforceConnector.jar'.

Explanation: Accessing version information from manifest failed.

Action: Inform DirX Identity technical support about the problem.

WAR(SLSF401): Unknown or missing object type '*string1*' in SPML request.

Explanation: The operational attribute 'objtype' either is not present or holds an unknown object type.

Action: Check why the SPML request is not supported.

WAR(SLSF402): The object type 'group' will not be synchronized.

Explanation: The given object is not handled by the Salesforce connector.

Action: Check why the Salesforce connector receives a SPML request for object type 'group'.

WAR(SLSF403): Too many value for attribute 'ProfileId': *string1*; only 1 is allowed.

Explanation: You only can assign one group to a user and thereby assign one ProfileId for a Salesforce user.

Action: Assign only one group to a user in Identity Store.

ERR(SLSF500): Error while converting SPML request; missing identifier.

Explanation: An illegal SMPL request has been sent to the Salesforce connector. The identifier is a mandatory parameter in the request.

Action: Check why an illegal request has been received.

FTL(SLSF900): Exception occurred with exception message: *string1*.

Explanation: An exception occurred with the given exception message.

Action: Check the exception message text.

FTL(SLSF901): Login to Salesforce failed; error message: *string1*.

Explanation: Login to Salesforce failed for the given reason.

Action: Check the Salesforce error message.

16.3.24. Logging Messages (SPCO)

This topic describes the logging messages returned by the SharePoint Connector for the DirX Identity Connector Framework.

INF(SPCO201): Connected to SharePoint Server: '*string1*' with port: '*string2*' to Site: '*string3*' successfully.

Explanation: The connection to the SharePoint server has been established successfully.

Action: None.

INF(SPCO202): Added entry: '*string1*' on Site '*string2*' successfully.

Explanation: The entry was added successfully.

Action: None.

INF(SPCO203): No owner specified in the add request for group '*string1*' on site '*string2*'. Setting owner to bind account '*string3*'

Explanation: The add request does not contain an owner identifier and an owner type for the new group.

Action: Check the mapping if you want to pass the owner information to the connector.

INF(SPCO204): No default login specified in the add request for group '*string1*' on site '*string2*'. Setting default login to bind account '*string1*'

Explanation: The add request does not contain a default user login name for the new group.

Action: Check the mapping if you want to pass the default login to the connector.

INF(SPCO205): The entry: '*string1*' on Site '*string2*' was modified successfully.

Explanation: The entry was modified successfully.

Action: None.

INF(SPCO206): Operation '*string1*' on local member '*string2*' for group '*string3*' on site '*string4*' ignored.

Explanation: Modifications on local members are ignored.

Action: None.

INF(SPCO207): The entry: '*string1*' on site '*string2*' was deleted successfully.

Explanation: The entry was deleted successfully.

Action: None.

INF(SPCO208): Added bind account domain '*string1*' to username '*string2*'.

Explanation: A username did not contain the domain. Therefore the domain from the bind account is used.

Action: Check the mapping if the domain should be contained in the username.

INF(SPCO209): Groups will be searched from the specified (sub)site, not from the site collection.

Explanation: All group searches are performed on the specified site. This is the default.

Action: None.

INF(SPCO210): Groups will be searched from the complete site collection, not only from the specified (sub)site.

Explanation: All group searches are performed on the complete site collection.

Action: None.

INF(SPCO211): Account names are extended by the prefix *string1*.

Explanation: A prefix is configured at the connected directory that is added to account names in SharePoint.

Action: None.

INF(SPCO212): No prefix is configured to extend account names.

Explanation: No account name prefix is configured to be added in SharePoint.

Action: None.

INF(SPCO213): Add the configured prefix to the group owner=*string1*.

Explanation: If an index is configured, the flag for adding the prefix to the group owner is reported.

Action: None.

INF(SPCO214): Add the configured prefix to the UserLoginName=*string1*.

Explanation: If an index is configured, the flag for adding the prefix to the

UserLoginName is reported.

Action: None.

INF(SPCO215): The configured prefix *string1* is missing at owner *string2* of group *string3* returned when searching site *string4*.

Explanation: A prefix is configured for the current site. Thus, the owners are expected to start with the configured prefix. This prefix is not present at the owner, it is returned 'as is' without modifications.

Action: Check and correct the configuration.

INF(SPCO216): Writing search result for group '*string1*' to spml response.

Explanation: Reporting for which group the search result info is written to the spml response.

Action: None.

INF(SPCO217): Reading members of group '*string1*'.

Explanation: Reading members of reported group.

Action: None.

INF(SPCO218): Reading roles of group '*string1*'.

Explanation: Reading roles of reported group.

Action: None.

INF(SPCO219): Try to establish connection to SharePoint Server: '*string1*' with port: '*string2*' to Site: '*string3*'.

Explanation: Try connection to SharePoint server.

Action: None.

WAR(SPCO400): Cannot access version information from jar file.

Explanation: Accessing version information from manifest failed.

Action: Inform DirX Identity technical support about the problem.

WAR(SPCO401): Connection endpoint *string1* does not end with site *string2*.

Explanation: The endpoint of the SOAP connection is expected to end with the site name.

Action: Check and correct the configuration.

WAR(SPCO402): The configured prefix *string1* is missing at member *string2* of group *string3* returned when searching site *string4*.

Explanation: A prefix is configured for the current site. Thus, the members are expected to start with the configured prefix. This prefix is not present at the member, it is

returned 'as is' without modifications.

Action: Check and correct the configuration.

ERR(SPCO501): Adding entry '*string1*' on site '*string2*' failed: '*string3*'.

Explanation: Adding the entry failed.

Action: Check the SharePoint error code and perform appropriate actions.

ERR(SPCO502): Adding entry '*string1*' on site '*string2*' failed: The object class '*string3*' is not supported.

Explanation: Adding the entry failed due to unsupported object class. Only the object class group is supported.

Action: Add only entries with object class group.

ERR(SPCO503): Modifying entry '*string1*' on site '*string2*' failed: '*string3*'.

Explanation: Modifying the entry failed.

Action: Check the SharePoint error code and perform appropriate actions.

ERR(SPCO504): Deleting entry '*string1*' on site '*string2*' failed: '*string3*'.

Explanation: Deleting the entry failed.

Action: Check the SharePoint error code and perform appropriate actions.

ERR(SPCO505): Incorrect SharePoint Search Filter: '*string1*' on site '*string2*'.

Explanation: The SharePoint Search Filter is incorrect.

Action: Check and correct the SharePoint search filter.

ERR(SPCO506): Unknown SharePoint Search Attribute: '*string1*' on site '*string2*'.

Explanation: The SharePoint Search Attribute is incorrect.

Action: Check and correct the requested SharePoint search attributes.

ERR(SPCO507): The SharePoint search on site '*string1*' failed: '*string2*'.

Explanation: The SharePoint search failed.

Action: Check the SharePoint error code and perform appropriate actions.

ERR(SPCO508): The Owner Identifier for the group: '*string1*' on site '*string2*' could not be resolved.

Explanation: The Attribute "ownerIdentifier" was requested. The ownerID could not be resolved to an owner identifier.

Action: Check and correct the group in SharePoint.

ERR(SPCO509): Adding member '*string1*' to group '*string2*' on site '*string3*' failed: '*string4*'.

Explanation: Adding a member to the SharePoint group failed. Presumably the account does not exist.

Action: Check the message details. A later validation run will fix this problem once the account exists.

ERR(SPCO510): Adding role '*string1*' to group '*string2*' on site '*string3*' failed: '*string4*'.

Explanation: Adding a role to the SharePoint group failed. Presumably the role does not exist.

Action: Check the message details. Create the role in SharePoint or rename it in DirX Identity.

ERR(SPCO511): Deleting member '*string1*' from group '*string2*' on site '*string3*' failed: '*string4*'.

Explanation: Deleting a member from the SharePoint group failed. Presumably the account does not exist.

Action: Check the message details. A later validation run will fix this problem once the account exists.

ERR(SPCO512): Deleting role '*string1*' from group '*string2*' on site '*string3*' failed: '*string4*'.

Explanation: Deleting a role from the SharePoint group failed. Presumably the role does not exist.

Action: Check the message details. Create the role in SharePoint or rename it in Identity.

ERR(SPCO513): Getting the group members for group '*string1*' failed: '*string2*'. Performing the other groups is continued.

Explanation: Getting the group members for the indicated group failed.

Action: Check the permission settings for that group in SharePoint and perform appropriate actions.

ERR(SPCO514): Getting the group roles for group '*string1*' failed: '*string2*'. Performing the other groups is continued.

Explanation: Getting the group roles for the indicated group failed.

Action: Check the permission settings for that group in SharePoint and perform appropriate actions.

ERR(SPCO515): Writing search result for group '*string1*' to spml response failed with exception: *string2*.

Explanation: Writing the search result to the response for the reported group failed.

Action: Check the exception message and correct the error.

FTL(SPCO901): Exception in connector while processing open method: '*string1*'.

Explanation: An exception occurred when opening the connector instance.

Action: Check the exception message and correct the error. If the problem persists inform DirX Identity technical.

FTL(SPCO902): No connector type was specified.

Explanation: The connector type is missing in the connection configuration.

Action: Specify the connector type in the configuration.

FTL(SPCO903): An incorrect connector type was specified: '*string1*'

Explanation: The connector type is incorrect. The type must be SharePoint.

Action: Specify connector type SharePoint in the configuration.

FTL(SPCO904): Cannot determine the address of the SharePoint server; a connection is not specified for the connector.

Explanation: The <connection> tag is missing in the definition of the connector.

Action: Add a <connection> tag to the definition of the connector in the configuration file. Note that the system might build the configuration file automatically from a template that contains references. In this case you must take care for proper reference resolution.

FTL(SPCO905): Cannot bind to SharePoint server; no server definition is specified for the connection.

Explanation: The name of the SharePoint server is not specified for the connection.

Action: Add a server="<server>" property to the connection tag where <server> specifies the name of the SharePoint server. Note that the system might build the configuration file automatically from a template that contains references. In this case you must take care for proper reference resolution.

FTL(SPCO906): Cannot bind to SharePoint server; no site definition is specified for the connection to server '*string1*'.

Explanation: The name of the SharePoint site is not specified for the connection.

Action: Add a <property name="site" value="<site>"/> property to the connection tag where <site> specifies the name of the SharePoint site. Note that the system might build the configuration file automatically from a template that contains references. In this case you must take care for proper reference resolution.

FTL(SPCO907): Cannot bind to SharePoint server; no user definition is specified in connection to site '*string1*'.

Explanation: The user property is not specified for the connection.

Action: Add a user="<user>" property to the connection tag where <user> specifies the user with the appropriate rights. Note that the system might build the configuration file automatically from a template that contains references. In this case you must take care

for proper reference resolution.

FTL(SPCO908): Cannot bind to SharePoint server; no password definition is specified for the connection to site '*string1*'.

Explanation: The password is not specified for the connection.

Action: Add a password="<password>" property to the connection tag where <password> specifies the users password. Note that the system might build the configuration file automatically from a template that contains references. In this case you must take care for proper reference resolution.

FTL(SPCO909): Bind to the SharePoint server failed: '*string1*'.

Explanation: Binding to the SharePoint server failed.

Action: Check the connection parameters. Check whether the server is running.

FTL(SPCO910): Exception while accessing version information from jar file.

Explanation: Accessing version information from manifest failed.

Action: Inform DirX Identity technical support about the problem.

FTL(SPCO911): Cannot bind to SharePoint server; no domain definition is specified on user for the connection to site '*string1*'.

Explanation: The domain is not specified for the connection.

Action: Add a domain to the username.

16.3.25. Logging Messages (SPUH)

This topic describes the logging messages returned by the SharePoint user hook for the DirX Identity Connector Framework.

INF(SPUH200): Group DN=*string1* was deleted in DirX Identity. The group is not deleted in SharePoint because the option 'delete_group_enabled' is not set.

Explanation: Delete is DISABLED. To enable delete, set the option 'delete_group_enabled' to true.

Action: None.

INF(SPUH201): Search DirX Identity account for SharePoint user *string1* failed for group *string2*. Adding *string3*='*string4*'.

Explanation: A SharePoint user does not exist in the peer target systems. Thus, its value is added to the 'remote' attribute.

Action: Create an account for the user in a peer target system to enable full provisioning functionality.

WAR(SPUH401): No referenced values for search value *string1*.

Explanation: The account mapping search could not find a referenced value for the specified search value.

Action: Please check your connector configuration.

ERR(SPUH501): Invalid target system configuration: No TS-DN specified and no value for role_ts_group_base set in environment.

Explanation: The DN for the target system could not be read from your configuration.

Action: Please check your target system configuration.

ERR(SPUH502): Invalid target system structure: TS-DN could not be read from DN *string1*.

Explanation: The DN for the target system is not valid.

Action: Please check your target system configuration.

ERR(SPUH503): Configuration error for target system DN=*string1*: No value configured for dxrpeerts.

Explanation: A value for dxrpeerts is required for the target system DN.

Action: Please check your target system configuration.

ERR(SPUH504): Search for account failed for *string1=string2* of group *string3*. Member not processed.

Explanation: A member cannot be mapped to an account name because the user has no account in one of the peer target systems. Thus, the member is not processed in SharePoint.

Action: Check the user's privileges. Add a role resulting in an active directory account if the user should have an account.

ERR(SPUH505): Search for account failed for owner=*string1* of group *string2*. Owner not processed.

Explanation: An owner cannot be mapped to an account name because the owner has no account in one of the peer target systems. Thus, the owner is not processed in SharePoint.

Action: Check the owner's privileges. Add a role resulting in an active directory account if the owner should have an account there.

ERR(SPUH506): Cannot map member '*string1*' to account name: Both '*string2*' and '*string3*' are empty for account DN=*string4*.

Explanation: A member cannot be mapped to a SharePoint user because the account name and the domain name are empty in the peer account.

Action: Check the peer account and the AD workflows.

ERR(SPUH507): Cannot map member '*string1*' to account name: '*string2*' is empty for account DN=*string3*.

Explanation: A member cannot be mapped to a SharePoint user since account name or domain name is empty in the peer account.

Action: Check the peer account and the AD workflows.

ERR(SPUH509): Userhook 'preUpdate' failed for source object '*string1*', error: *string2*.

Explanation: Preupdate failed with an exception. Synchronization of the given object is not executed.

Action: Check the error messages.

FTL(SPUH901): An error occurred while converting the search string from LDAP to SPML: *string1*.

Explanation: The search in mapAccountData could not be performed because an exception occurred in FilterLdap2Spml.buildSPMLFilter.

Action: Inform DirX Identity support about the problem.

16.3.26. Logging Messages (SPV1V2)

This topic describes the logging messages returned by the Spmlv1-to-v2 connector in DirX Identity.

ERR(SPV1V2001): No endpoint (URL) configured.

Explanation: A SOAP client requires address and port of the remote web service.

Action: Add URL or ssl, server, port and path to configuration.

INF(SPV1V2002): Authenticating as *string1* with *string2* password.

Explanation: SpmlV1ToV2 connector uses HTTP basic authentication with configured user and password.

Action: None.

ERR(SPV1V2003): Exception while opening SpmlV1ToV2 connector: *string1*.

Explanation: An exception occurred while opening the SpmlV1ToV2 connector.

Action: Check the configuration of the Connector and availability of necessary libraries.

INF(SPV1V2004): Sending *string1* with id *string2* to url *string3*.

Explanation: The SPMLv2toV1Connector is about to send the SPMLv2 request with the given request identifier to the given URL.

Action: None.

ERR(SPV1V2005): SpmlV1ToV2 connector received SOAP Fault Exception. Code = *string1*, FaultString = *string2*, Detail: *string3*

Explanation: The requested Web Service is not available or returned a SOAP Fault.

Action: Check SOAP Service: is it started and runs without errors?

ERR(SPV1V2006): Exception *string1* in SpmlV1ToV2Connector while unmarshalling response for *string2* with id '*string3*': *string4*.

Explanation: An Exception occurred when sending a request to the SOAP server and reading its response.

Action: Check the details. Probably the response cannot be evaluated: is it well-formed? Does the responded SOAP version match that of the request?

ERR(SPV1V2007): Exception *string1* in SpmlV1ToV2Connector while sending request *string2* with id *string3*: *string4*

Explanation: An Exception occurred when sending a request to the SOAP service.

Action: Check SOAP Service: address, port, is it started?

ERR(SPV1V2008): SpmlV1ToV2Connector received null response for sent request *string2* with id *string3*.

Explanation: The SOAP Service at least has to return a Fault response, but not null.

Action: Check the Spmlv2 Service logs.

ERR(SPV1V2009): Axis Fault received in SpmlV1ToV2Connector as response for request *string1*: *string2*.

Explanation: The remote Web Service returned an Axis Fault response.

Action: Check the message details and correct the Web Service.

ERR(SPV1V2010): SOAP body missing in response for request *string1*.

Explanation: The remote Web Service returned a SOAP response without body.

Action: Correct the Web Service.

ERR(SPV1V2011): Can’t extract message element from SOAP body in response for request *string1*.

Explanation: The remote Web Service returned a SOAP response without message within body tag.

Action: Correct the Web Service.

ERR(SPV1V2012): SpmlV1ToV2Connector cannot extract DOM from message element in response to request *string1*. Message element: *string2*.

Explanation: The message element in the SOAP response body seems to be wrong.

Action: Check and correct the response format of the remote Web Service.

INF(SPV1V2013): SpmlV1ToV2Connector received *string1* with id *string2* and status = *string3*.

Explanation: The connector received a SPMLv2 response and is going to transform it to

Spmlv1.

Action: None.

ERR(SPV1V2014): SpmlV1ToV2Connector received a SPMLv1 DeleteRequest without Identifier. Requestid = *string1*.

Explanation: A DeleteRequest needs an identifier both in SPMLv1 and v2.

Action: Make sure the SPMLv1 DeleteRequest has an identifier.

ERR(SPV1V2015): SpmlV1ToV2Connector received a SPMLv1 ModifyRequest without Identifier. Requestid = *string1*.

Explanation: A ModifyRequest needs an identifier both in SPMLv1 and v2.

Action: Make sure the SPMLv1 ModifyRequest has an identifier.

ERR(SPV1V2016): SpmlV1ToV2Connector received a SPMLv1 SearchRequest without Identifier. Requestid = *string1*.

Explanation: A SPMLv2 LookupRequest or SearchRequest needs an identifier both in SPMLv1 and v2.

Action: Make sure the SPMLv1 SearchRequest has an identifier.

ERR(SPV1V2017): Exception *string1* while unmarshalling a received SPMLv2 response for *string2* with id '*string3*': *string4*.

Explanation: SpmlV1ToV2Connector tried to unmarshall a response string received from the remote web service and caught an exception.

Action: Make sure the SPMLv2 web service returns SPMLv2 compatible responses.

ERR(SPV1V2018): Exception *string1* while casting a received SPMLv2 response to Iterate response: *string2*.

Explanation: SpmlV1ToV2Connector tried to cast the response for an iterate request and caught an exception.

Action: Make sure the SPMLv2 web service returns a valid SPMLv2 iterate response for an iterate request.

ERR(SPV1V2019): IterateRequest for iteratorID = '*string1*' failed with errorCode = '*string2*' and error message = '*string3*'.

Explanation: SpmlV1ToV2Connector received an iterate response with status code "failed".

Action: Check the SPMLv2 web service for the reason.

ERR(SPV1V2020): Instantiating Reference handler with class name = '*string1*' failed: *string2*.

Explanation: SpmlV1ToV2Connector tried to create a class implementing the interface 'Spmlv2ReferenceHandler'.

Action: Make sure the class implements the interface and is in the classpath.

ERR(SPV1V2021): Exception *string1* in SpmlV1ToV2Connector while unmarshalling Search- or LookupResponse for *string2* with id '*string3*': *string4*.

Explanation: An Exception occurred when handling a SPMLv1 SearchRequest or its SPMLv2 response.

Action: Probably the XML of the request or response is not well-formed?

ERR(SPV1V2022): Instantiating password handler with class name = '*string1*' failed: *string2*.

Explanation: SpmlV1ToV2Connector tried to create a class implementing the interface 'Spmlv2PasswordHandler'.

Action: Make sure the class implements the interface and is in the classpath.

ERR(SPV1V2023): Instantiating capability handler with class name = '*string1*' failed: *string2*.

Explanation: SpmlV1ToV2Connector tried to create a class implementing the interface 'Spmlv2CapabilityHandler'.

Action: Make sure the class implements the interface and is in the classpath.

ERR(SPV1V2024): Capability attributes configured (*string1*), but no handler.

Explanation: SpmlV1ToV2Connector needs a handler for capability attributes. There is no Default.tried to create a class implementing the interface .

Action: Make sure to configure a property 'capabilityHandler' with the name of a class that implements the interface 'Spmlv2CapabilityHandler'.

ERR(SPV1V2025): Exception *string1* in SpmlV1ToV2Connector while setting password via handler *string2* for request with id '*string3*': *string4*.

Explanation: The configured password handler is expected to produce a SPMLv2 setPasswordRequest, but threw an exception.

Action: Check the handler implementation and its log messages.

ERR(SPV1V2026): *string1* when loading class with name '*string2*': *string3*.

Explanation: When instantiating a configured user hook the referred class could not be instantiated. Job loading fails.

Action: Check the jar files and the classpath.

INF(SPV1V2027): SpmlV1ToV2Connector received IterateResponse: #PSOs = *string1*, IteratorID = '*string2*', #Entries collected so far: *string3*.

Explanation: The connector received a SPMLv2 iterate response and is going to send another iterate request to obtain missing result entries.

Action: None.

INF(SPV1V2028): SpmlV1ToV2Connector received IterateResponse with IteratorID = '*string1*'

and requestID = '*string2*', but containing no result entries! Number Entries collected so far: *string3*.

Explanation: This is a non-conformant behaviour of the SPMLv2 Web Service. But the connector ignores it and considers the search finished.

Action: Try to correct the SPMLv2 Web Service.

ERR(SPV1V2029): *string1* in SpmlV1ToV2Connector calling handler *string2* for *string3*: *string4*.

Explanation: The configured handler is called to transform the configured attributes from the SPMLv1 request to attributes in a SPMLv2 request.

Action: Check the handler implementation and its log messages.

ERR(SPV1V2030): *string1* in SpmlV1ToV2Connector when setting options in handler *string2*: *string3*.

Explanation: The configured handler throws an exception in its method 'setOptions'.

Action: Check the handler implementation and its log messages.

16.4. Logging Messages for High Availability

ADC... Logging messages returned by the DirX Identity Admin Client.

16.4.1. Logging Messages (ADC)

This topic describes the logging messages returned by the DirX Identity Admin Client.

INF(ADC002): User: *string1* is not member of ServerAdmins group.

Explanation: Could not authenticate as given user is not member of ServerAdmins group.

Action: None.

INF(ADC003): Wait *string1* s to give other servers time to start up.

Explanation: Supervisor waits before starting monitoring.

Action: None.

INF(ADC004): Starting supervision loop.

Explanation: Supervisor starts monitoring.

Action: None.

WAR(ADC005): *string1* while Supervisor is waiting to start next monitor cycle: *string2*.

Explanation: Supervisor was interrupted during initial wait or within monitor loop. Monitoring will be cancelled.

Action: Check logging.

WAR(ADC006): Interrupted during wait. Class: *string1* Error: *string2*.

Explanation: Supervisor was interrupted during wait in supervision loop. Supervision loop stops.

Action: Check logging.

INF(ADC007): Move Monitor ATS Flag from '*string1*' to '*string2*' necessary.

Explanation: Supervisor will try to move the monitor ATS flag. As from server seems to be down, the to server should monitor C++ based servers.

Action: None.

WAR(ADC008): Move of Monitor ATS Flag from '*string1*' to '*string2*' failed: '*string3*'.

Explanation: Supervisor could not move the monitor ATS flag. The task monitoring C++ based servers is not overtaken by to server.

Action: None.

INF(ADC009): Move of ATS necessary: C++ based server '*string1*' is down.

Explanation: Supervisor detects that the C++ based server hosting the ATS is down. It will try to move the ATS and Status Tracker.

Action: None.

WAR(ADC010): *string1*, no move is done."

Explanation: Supervisor could not evaluate the fail-over C++ based Server for ATS. ATS might not work as the C++ based server hosting it seems to be down.

Action: Check configuration(Alternate Messaging Service at Message Service Object and C Server Object).

WAR(ADC011): Move ATS from: '*string1*' to: '*string2*' failed: *string3*.

Explanation: Supervisor could not move ATS to the fail-over C++ based Server. ATS might not work as the C++ based server hosting it seems to be down.

Action: Check Error message.

WAR(ADC012): Recover messages from inactive server: '*string1*' on: '*string2*' failed: *string3*.

Explanation: Supervisor could not recover the messages of failed server.

Action: Check Error message.

INF(ADC013): Recover messages from inactive server: '*string1*' on: '*string2*' successfully initiated.

Explanation: Supervisor had successfully initiated recovery of ActiveMQ and JGroups.

Action: None.

INF(ADC200): *string1*

Explanation: Generic error message

Action: None.

INF(ADC213): getCServersToCheck: skip CServer: '*string1*' it is not registered.

Explanation: Method getCServersToCheck skips given C++-based server. Register flag is not set, assuming normal shutdown.

Action: None.

INF(ADC214): Moving Tcl workflows from Server '*string1*' to Server '*string2*'.

Explanation: All TCL workflows and activities are moved to the given server. This means the corresponding LDAP attribute for the DirX Identity server is updated in the activity and workflow objects.

Action: None.

INF(ADC215): Moving dxmRunsScheduler flag from Server '*string1*' to Server '*string2*'.

Explanation: Java scheduler is moved to the given server. This means the corresponding LDAP attribute dxmRunsScheduler is updated at the corresponding ids-j server objects.

Action: None.

INF(ADC216): Moving 'Monitor C++-based Servers' flag from Server '*string1*' to Server '*string2*'.

Explanation: The flag 'Monitor C++-based Servers' is moved to the given server. This means the corresponding LDAP attribute dxmMonitorATS is updated at the corresponding ids-j server objects.

Action: None.

WAR(ADC300): *string1*

Explanation: Generic error message

Action: None.

ERR(ADC500): *string1*

Explanation: Generic error message

Action: None.

ERR(ADC801): "No Server found with this name: '*string1*'.

Explanation: AdminClient did not find given server by name.

Action: None.

ERR(ADC802): "Adaptor '*string1*' is moveable. Cannot set active flag..

Explanation: Moveable adaptors cannot be set to active via the setAdapterActive method.

Action: For moveable adaptors use one of the moveAdaptor-methods.

16.5. Logging Messages for the Java-based Server Components

ACKUPD... Logging messages of the Java-based DirX Identity server acknowledge update job integration.

BCK... Logging messages of the IDSJ extension for backup.

CCC... Logging messages of the DirX Identity Consistency Check Control.

CPGCCTRL... Logging messages of the DirX Identity Campaign Control.

CPGSVC... Logging messages of the DirX Identity Campaign Service.

DCFG... Logging messages of the IDSJ extension for domain configuration.

EBR... Logging messages for the event based rules.

ELM... Logging messages of the DirX Identity Entry Lock Manager.

ICF... Logging messages for the framework job integration component.

IDSJ... Logging messages for the IDSJ extension component.

IDSJADP... Logging messages for the IDSJ server adaptors.

IDSJAPI... Logging messages for the IDSJ server API.

IDSJBS... Logging messages for the BackupSlaveListener.

IDSJCFG... Logging messages for the IDSJ server configuration.

IDSJJOB... Logging messages for the IDSJ server builtin jobs.

IDSJUTL... Logging messages for the IDSJ extension utilities component.

IOJ... Logging messages for the order job integration component.

IPO... Logging messages for the pool manager component.

JAL... Logging messages for the IDSJ extension for JMS audit log handler.

JINI... Logging messages for the JINI infrastructure component.

JINIAPI... Logging messages for the JINI infrastructure API.

JOIN... Logging messages for the join engine component.

LCP... Logging messages of the DirX Identity ldap connection pool.

REQWF... Logging messages of the IDSJ extension for request workflows.

RSKCTRL... Logging messages of the DirX Identity Risk Governance Controller.

RTWF... Logging messages of the IDSJ extension for realtime workflows.

SCE... Logging messages for the cryptographic library (Java code).

SCKJOB... Logging messages of the DirX Identity Socketed Job.

TCK... Logging messages of the DirX Identity ticket control.

WEBADM... Logging messages of the IDSJ extension for WebAdmin.

16.5.1. Logging Messages (ACKUPD)

This topic describes the logging messages returned by the Java-based DirX Identity server acknowledge update job integration.

INF(ACKUPD200): Cannot get value for *string1* attribute from order in add order request for *string2*.

> **Explanation**: The acknowledge update job could not update the attribute according to attribute values taken from add order. Add order request need not to contain this attribute.
>
> **Action**: None.

INF(ACKUPD201): Cannot get value for *string1* attribute from order in modify order request for *string2*.

> **Explanation**: The acknowledge update job could not update the attribute according to attribute values taken from modify order. Modify order request need not to contain this attribute on all changes.
>
> **Action**: None.

ERR(ACKUPD500): Invalid order type, only *string1*, *string2* and *string4* are supported.

> **Explanation**: The acknowledge update job supports only described type of orders.
>
> **Action**: Check the request workflow setting, use this job only for correct order types.

ERR(ACKUPD501): Cannot get value for dxrState attribute from order in add order request for *string1*.

> **Explanation**: The acknowledge update job could not update the dxrTSState attribute according to dxrState attribute values taken from add order.
>
> **Action**: Check the configuration of the part where the add order is built (connectivity workflow). Be sure that dxrState is always part of the add order requests.

ERR(ACKUPD502): Empty add or modify order request for *string1*.

Explanation: The acknowledge update job could not update state attributes (dxrTSState and dxrGroupMember*) according to attributes values taken from add or modify order because no attributes are present.

Action: Check the configuration of the part where the add and modify orders are built (connectivity workflow). Be sure that only add and modify order requests with valid attribute or modification list are generated.

ERR(ACKUPD503): Could not write object *string1* and attribute *string2* to LDAP.

Explanation: The acknowledge update job could not update attributes values in LDAP.

Action: Check the cause of the linked exception.

ERR(ACKUPD504): Could not read object *string1* or its attribute from LDAP.

Explanation: The acknowledge update job could not read the current values from LDAP.

Action: Check the cause of the linked exception.

ERR(ACKUPD505): Could not read object *string1*, object is null.

Explanation: The acknowledge update job could not read the current values from LDAP.

Action: Check the existence of the object.

ERR(ACKUPD506): Could not read dxrGroupMemberOK, dxrGroupMemberAdd and dxrGroupMemberDelete attributes for *string1*.

Explanation: The acknowledge update job could not read the current attribute values from LDAP.

Action: Check the validity of these attributes for this object.

16.5.2. Logging Messages (BCK)

This topic describes the logging messages returned by the IdS-J extension for Backup.

ERR(BCK001): Internal error: *string1*

Explanation: An unexpected error occurred during a backup operation.

Action: Inform DirX Identity technical support about the problem.

WAR(BCK002): Could not initialize JMX support for extension '*string1*'.

Explanation: Repository will not be manageable via JMX due to an initialization error.

Action: Inform the system administrator.

WAR(BCK003): Could not initialize JMX support for repository '*string1*'.

Explanation: Repository will not be manageable via JMX due to an initialization error.

Action: Inform the system administrator.

INF(BCK004): Repository 'Backup feature' for Joint Backup is not enabled.

Explanation: Extension Backup can only work if backup for the server is enabled. Note that High Availability and Joint Backup can not be activated both.

Action: Enable server backup in the configuration and restart the service.

ERR(BCK005): DirX Messaging server backup failed due to: *string1*

Explanation: Backup of the DirX Messaging server failed due to the reported error.

Action: Check the detailed message and perform appropriate actions.

ERR(BCK006): DirX Directory server backup of *string1* failed due to: *string2*

Explanation: Backup of the DirX Server server failed due to the reported error.

Action: Check the detailed message and perform appropriate actions.

16.5.3. Logging Messages (CCC)

This topic describes the logging messages returned by the DirX Identity Consistency Check Control.

ERR(CCC000): *string1* unsupported operation: '*string2*'.

Explanation: The controller does not implement this operation.

Action: Make sure the operation is not passed to the controller.

INF(CCC001): *string1* received *string2* with id = '*string3*' and topic = *string4*.

Explanation: The activity reader has passed this event to the activity controller.

Action: None.

ERR(CCC002): Connection option missing: server = '*string1*', port = '*string2*', user = '*string3*', domain = '*string4*'.

Explanation: The <connection> element of the job controller does not contain all necessary parameters.

Action: Make sure all needed connection parameters are provided.

ERR(CCC004): Cannot bind to domain *string1* on '*string2*:_string3_' as '*string4* with ssl = '*string5*'; ErrorCode = *string6*.

Explanation: Controller cannot bind to DirX Identity domain.

Action: Check address and bind credentials. Is LDAP server available?

ERR(CCC005): Exception *string1* when loading configuration from domain '*string2*': *string3*.

Explanation: Controller cannot load configuration from DirX Identity domain.

Action: Check domain configuration: Object descriptions, etc.

WAR(CCC006): Exception *string1* when closing connection to domain '*string2*': *string3*.

Explanation: On close controller tried to dispose and disconnect the session to the DirX Identity domain.

Action: Check if the LDAP server is still available.

INF(CCC007): No tickets to process beneath '*string1*'.

Explanation: At the moment there are no tickets available that need to be processed.

Action: None.

ERR(CCC008): createTicketProcessor failed Exception: *string1*.

Explanation: Unable to create Ticket processor. Processing of tickets is not possible.

Action: Check the details of the exception.

ERR(CCC009): Save of Ticket '*string1*' failed with rc: *string2*.

Explanation: Ticket was successfully processed but the save to set the State APPLY_CHANGE_COMPLETED failed.

Action: Check rc and perform appropriate action.

ERR(CCC010): applyTicket for '*string1*' failed.

Explanation: applyTicket to process the orders of given ticket failed. State of ticket does not change.

Action: Check logging of TicketProcessor.

ERR(CCC011): Exception *string1* during applyTicket for DN = '*string2*'.

Explanation: applyTicket to process the orders of given ticket throws an exception. State of ticket does not change.

Action: Check the details of the exception.

ERR(CCC012): Exception *string1* in *string2* while reading configuration options: *string3*.

Explanation: The activity controller tried to read configured options and address and bind credentials from LDAP reader.

Action: Check the details of the exception. Check the configuration options.

ERR(CCC013): Exception '*string1*' during search for tickets to process: baseDN = '*string2*' filter='*string3*'.

Explanation: Search for tickets to be processed failed. No tickets were processed.

Action: Check the details of the exception.

INF(CCC014): Ticket '*string1*' successfully processed, state changed to '*string2*'.

Explanation: Ticket successfully processed. All orders applied. State changed to APPLY_CHANGE_COMPLETED.

Action: None.

WAR(CCC015): Saving the orders of approval workflow '*string1*' at ticket '*string2*' failed with rc *string3*.

Explanation: Approver may have changed some attribute values of the order. These changes should be stored at the ticket. But the save operations fails. Changes of approvers may be lost.

Action: Check logging.

INF(CCC016): Orders of approval workflow '*string1*' successfully transferred to ticket '*string2*'.

Explanation: Approver may have changed some attribute values of the order. These changes are now also stored at the ticket.

Action: None.

WAR(CCC017): Exception '*string1*' while fetching workflow instance '*string2*'of ticket ' *string3*'.

Explanation: Ticket has set a dxrWorkflowLink attribute which seems not to reference an existing workflow instance. DxrWorkflowLink is ignored. Ticket processing continues.

Action: Check the exception details. Also check the dxrWorkflowLink attribute.

INF(CCC018): Setting state of ticket '*string1*' to '*string2*'. Reason: approval workflow '*string3*' failed with state *string4*.

Explanation: Approval workflow failed. The state of the ticket changes as indicated in the message.

Action: None.

INF(CCC019): Setting state of ticket '*string1*' to '*string2*'. Reason: approval workflow '*string3*' succeeded with applicationstate *string4*.

Explanation: Approval workflow succeeded but ticket in state in_approval. The state of the ticket changes as indicated in the message.

Action: None.

WAR(CCC020): Saving state changes at ticket '*string1*' with rc : *string2*.

Explanation: State of approval workflow causes a state change of the ticket. The save fails. Ticket processing continues.

Action: Check the message details.

INF(CCC021): State of ticket '*string1*' successfully changed.

Explanation: State of approval workflow causes a state change of the ticket. State changed successfully.

Action: None.

INF(CCC022): Applying Master Ticket first '*string1*'.

Explanation: Process Master Ticket first. It's a prerequisite for the Sub Ticket.

Action: None.

INF(CCC023): Ticket '*string1*' : master ticket '*string2*' not processed, skip it.

Explanation: Prerequisite Master Ticket could not be processed. Sub Ticket is not processed.

Action: Check state of Master ticket.

INF(CCC024): Ticket '*string1*' : master ticket '*string2*' processing failed, skip it.

Explanation: Prerequisite Master Ticket processing failed. Sub Ticket is not processed.

Action: Check state of Master ticket.

ERR(CCC025): Exception *string1* resolving user with DN = '*string2*' due to an addEvent: *string3*.

Explanation: After the controller received an addEvent it tried to resolve the user, but got a runtime exception.

Action: Check the message details.

ERR(CCC026): Exception *string1* resolving user with DN = '*string2*' due to a modifyEvent: *string3*.

Explanation: After the controller received a modifyEvent it tried to resolve the user, but got a runtime exception.

Action: Check the message details.

ERR(CCC027): Exception *string1* while reading object of type *string2* with DN = '*string3*': *string4*.

Explanation: The controller tries to read the node from the domain using the identifier from the event.

Action: Check the message details. Does the object exist and is it of expected type? Is the class allowed to read the node?

ERR(CCC028): Exception *string1* processing modify event for object of type *string2* and DN = '*string3*': *string4*.

Explanation: After the controller received a modifyEvent it tried to read and process the node, but got a runtime exception.

Action: Check the message details.

ERR(CCC029): Exception *string1* when reading LDAP entry with DN='*string2*' during link check of object with DN = '*string3*': *string4*.

Explanation: While processing a change event for the object, the controller tries to read all referenced LDAP entries and received an exception different to NO_SUCH_OBJECT.

Action: Check the LDAP entry.

ERR(CCC030): The LDAP entry with DN='*string1*' referenced from object with DN='*string2*' in attribute '*string3*' does not exist.

Explanation: While processing a change event for the object, the controller tries to read all referenced LDAP entries to detect broken links and received an LDAP exception code NO_SUCH_OBJECT. The link is deleted.

Action: Check the LDAP entry.

ERR(CCC031): Exception *string1* while saving object with DN = '*string2*': *string3*.

Explanation: The controller tries to save changes to the object referenced in the change event.

Action: Check the message details and the object to be changed.

ERR(CCC032): Exception *string1* while processing an *string2*Event with requestID = '*string4*' for object of type *string3* with DN = '*string5*': *string6*.

Explanation: The event controller received an exception while processing the change event.

Action: Check the message details and the object which has been changed and processed.

ERR(CCC033): User hook configuration for event controller '*string1*' does not contain a class name.

Explanation: The controller configuration contains a <userhook> element, but the class name for the user hook is missing.

Action: Enter a class name for the user hook or drop the <userhook> element.

ERR(CCC034): Event controller '*string1*' cannot load user hook with class name '*string2*': *string3*.

Explanation: The class loader returns an exception when trying to load the class.

Action: Check the class name for the user hook in the <userhook> element and make sure the respective jar file is in the classpath.

ERR(CCC035): The user hook with class name '*string2*' returned a '*string1*' when calling its open method: *string3*.

Explanation: In the open method the user hook evaluates its configuration and initializes itself.

Action: Check for more detailed log messages of the user hook. Check the user hook

configuration.

WAR(CCC036): The user hook with class name '*string2*' returned a '*string1*' when calling its close method: *string3*.

Explanation: In the close method the user hook may terminate its resources.

Action: Check for more detailed log messages of the user hook.

ERR(CCC037): The user hook with class name '*string3*' returned a '*string1*' when processing an *string2*Event for Entry with DN='*string4*': *string5*.

Explanation: The event controller passed the event and the affected LDAP entry to the user hook. It ignores the exception and stores its own modifications.

Action: Check the user hook implementation and more detailed log messages of the user hook.

INF(CCC038): *string1* statistics: eventsProcessedOk = *string2*, eventsProcessedFailed = *string3*, updates = *string4*, upToDate = *string5*.

Explanation: The event controller informs about the number, type and result of processed events within one workflow run.

Action: None.

INF(CCC039): *string1* finished processing a *string2* for Entry with DN='*string3*:' *string4*ms.

Explanation: Time the event controller needed for processing this event.

Action: None.

ERR(CCC040): *string1* expected associated entry for '*string4*' of type *string2*, but received entry of class *string3*. DN or string representation of entry: *string5*.

Explanation: The event controller searched for associated objects and checked their types.

Action: Check the associated object: either remove the reference to the entry handled in the event or change to another type.

INF(CCC041): *string1* is updating associated entry of class *string2* with DN '*string3*.

Explanation: The event controller is going to change the associated entry, because a reference or other attribute needs to be deleted or changed.

Action: None.

ERR(CCC042): *string1* received *string2* when initializing rule context: *string3*.

Explanation: On startup the event controller searches consistency rules and initializes the rule context.

Action: Check the LDAP connection or inform DirX Identity Support about the problem.

ERR(CCC043): *string1* occurred when applying rule with DN='*string2*' to entry with

DN='*string3*': *string4*.

Explanation: The event runs the consistency rule for the entry identified in the change event. It ignores the problem.

Action: Check the rule implementation.

WAR(CCC045): Expected consistency rule but obtained object with DN = '*string1*' of class '*string2*'.

Explanation: The rule provider searched for consistency rules. The returned object is a storage object but not consistency rule. It ignores this object.

Action: Check and correct the LDAP entry with the given DN.

ERR(CCC046): Skipping processing of consistency rule with name = '*string1*' after initialization failure: *string2*.

Explanation: The consistency rule provider tried to read search filter, base and operation from found consistency rule entry.

Action: Check the message details. Check the referenced operator and the filter definition.

ERR(CCC047): Exception *string1* while searching consistency rules using filter = '*string2*': *string3*.

Explanation: The consistency rule provider searches consistency rules, but did not get a result.

Action: Check the message details.

ERR(CCC048): Cannot read consistency rules due to missing session.

Explanation: The consistency rule provider was requested to read consistency rules for the first time or after timeout, but did not get a session with LDAP access.

Action: Internal problem. Inform DirX Identity support.

INF(CCC049): *string1* updated entry of class *string2* with DN '*string3*.

Explanation: The event controller changed the entry identified in the event and updated it in LDAP.

Action: None.

WAR(CCC050): Expected validation rule but obtained object with name = '*string1*' of class '*string2*'.

Explanation: The rule provider searched for validation rules (accept, cleanup, validate). The returned object is a storage object but not validation rule. It ignores this object.

Action: Check and correct the LDAP entry with the given DN.

ERR(CCC051): Skipping processing of validation rule with Name = '*string1*' after initialization failure: *string2*.

Explanation: The validation rule provider tried to read search filter, base and operation from found validation rule entry.

Action: Check the message details. Check the referenced operator and the filter definition.

WAR(CCC053): Exception *string1* when searching associated user for account with DN = '*string2*' and search filter = '*string3*': *string4*.

Explanation: The Account Event Controller searches the associated user for an account. It tries the next set of attributes.

Action: Check the message details and the search filter.

ERR(CCC054): Cannot read provisioning rules due to missing session.

Explanation: The provisioning rule provider was requested to read provisioning rules for the first time or after timeout, but did not get a session with LDAP access.

Action: Internal problem. Inform DirX Identity support.

ERR(CCC055): Exception *string1* when cloning session: '*string2*'.

Explanation: At the start of a new job the controller tries to clone an existing SvcSession object to the DirX Identity domain. The controller tries to create a new session.

Action: Check the message details. Check availability of the LDAP server.

INF(CCC056): Successfully cloned Session for domain *string1* and effective user with DN '*string2*'.

Explanation: The event controller cloned an existing session to Identity domain.

Action: None.

ERR(CCC057): Cannot read validation rules due to missing session.

Explanation: The validation rule provider was requested to read validation rules for the first time or after timeout, but did not get a session with LDAP access.

Action: Internal problem. Inform DirX Identity support.

INF(CCC058): Added *string1* rule to cache: *string2*.

Explanation: The validation rule provider read this rule and added it to its cache.

Action: None.

ERR(CCC059): Exception *string1* while searching validation rules using filter = '*string2*': *string3*.

Explanation: The validation rule provider searches validation rules, but did not get a result.

Action: Check the message details.

ERR(CCC060): *string1* while reading validation filter from group '*string2*': *string3*.

Explanation: The validation rule provider tries to get the validation filter for this group in order to check whether the imported or ignored group member matches the filter.

Action: Check the validation filter definition of the group.

INF(CCC061): Adding account '*string1*' to group '*string2*' as consequence of rule '*string5*'. Old state was: *string3*, new state = *string4*.

Explanation: The validation rule provider adds the account to the group because they match the rule subject and privilege filters respectively.

Action: None.

ERR(CCC062): *string1* while adding account '*string2*' to group '*string3*': *string4*.

Explanation: The validation rule provider could not add the account to the group or save the group. The problem is ignored.

Action: Check the message details and try to add the account manually.

ERR(CCC063): *string1* while reading user '*string2*' of account '*string3*': *string4*.

Explanation: The validation rule provider could not read the user with the given DN. The onAssignment rules for the account are applied without the user.

Action: Check the message details. Often the user is not needed for the operation.

INF(CCC064): Deleting account '*string1*' from group '*string2*' as consequence of rule ' *string4*'. Old state was: *string3*.

Explanation: The validation rule provider removes the account from the group because it is a cleanup rule or they do not match any more the rule subject and privilege filters of a validation rule respectively.

Action: None.

ERR(CCC065): *string1* while removing account '*string2*' from group '*string3*': *string4*.

Explanation: The validation rule provider could not remove the account from the group or save the group. The problem is ignored.

Action: Check the message details and try to remove the account manually.

ERR(CCC066): *string1* when checking validation filter of group with DN = '*string2*' for account with DN = '*string3*' and validation filter = '*string4*': *string5*.

Explanation: The validation rule provider checked if the account matches the validation filter of the group. It assumes there is no match.

Action: Check the message details. Maybe you have to change the validation filter.

ERR(CCC067): *string1* when checking validation filter of obligations referencing the group with DN = '*string2*' for account with DN = '*string3*':_string4_.

Explanation: The validation rule provider checked if the account matches the validation filter of the obligations referencing the group. It assumes there is no match.

Action: Check the message details. Maybe you have to change the obligations.

WAR(CCC068): *string1* when searching for groups with search base = '*string2*' and filter = '*string3*' obtained from rule '*string4*': *string5*.

Explanation: The validation rule provider searches for groups matching the privilege filter(s) of the validation rule. It ignores the offended filter and continues with the next one.

Action: Check the message details and the filters.

WAR(CCC069): *string1* when casting obligationLink = '*string2*' of group with DN = '*string3*' to an obligation: *string4*.

Explanation: The validation rule provider instantiates the obligation links of the group as a SvcObligation object. It ignores the link and continues with the next one.

Action: Check the message details and the obligation link.

WAR(CCC070): *string1* when evaluating validation filter of obligation with DN = '*string2*': *string3*.

Explanation: The validation rule provider transforms the String value in the validation attribute of the obligation to a LDAP filter. It ignores the filter.

Action: Check the message details and the filter.

WAR(CCC071): Obligation link value '*string1*' of group with DN = '*string2*' is neither a String nor an array of Strings or links to Storage Objects.

Explanation: The validation rule provider tries to interpret the value as an obligation link. It ignores the link.

Action: Check the message details and the link.

WAR(CCC072): *string1* when reading obligation with DN = '*string2*' of group with DN = '*string3*': *string4*.

Explanation: The validation rule provider reads the obligation from LDAP. It ignores the link.

Action: Check the message details and the entry with the given DN.

WAR(CCC073): *string1* when reading groups where account with DN = '*string2*' is a member in state IMPORTED: *string3*.

Explanation: Before applying accept/cleanup rules the validation rule provider checks if the account is an imported member of some group. It does not apply any accept/cleanup rule on the account.

Action: Check the message details. Is the referencing in the target system correct?

INF(CCC074): Updating membership state of account '*string1*' in group '*string2*' to *string3* based on rule '*string4*'.

Explanation: By applying the accept/cleanup rule the validation rule provider changed the member state of the account from IMPORTED to IGNORE resp. DELETE.

Action: None.

WAR(CCC075): *string1* when updating membership state of account '*string2*' in group '*string3*': *string4*.

Explanation: The validation rule provider tries to change the membership state from IMPORTED to IGNORE resp. DELETE and save the group if it is necessary.

Action: Check the message details. Try to change the state manually if still necessary.

INF(CCC076): Associated user '*string2*' with account '*string1*'.

Explanation: The Account Event Controller succeeded in finding a user for the account matching the configured account attributes.

Action: None.

WAR(CCC077): *string1* when passing failed event for entry with DN '*string2*' to *string3*: *string4*.

Explanation: The event controller failed to process the change event. It tried to pass the event together with an error response to either the retry or error connector.

Action: Check the configuration for the retry or error connector. Check if the identifier section of the event is correct.

ERR(CCC078): *string1* while searching with baseDN = '*string2*' and filter = '*string3*': *string4*.

Explanation: The Storage Utilities performed a LDAP search resulting in this exception. The search internally switches to paged mode in case of sizelimit reached.

Action: Check the message details.

ERR(CCC079): *string1* when creating a session to Identity domain *string2*: *string3*.

Explanation: On start of the job the controller failed to clone an existing SvcSession or create a new one. The job is terminated.

Action: Check availability of the LDAP server, address and bind credentials.

INF(CCC080): Running consistency rule '*string1*' for Entry *string2* ...

Explanation: The Event Maintenance Controller applies the consistency rule to the given entry.

Action: None.

ERR(CCC081): *string1* when reading LDAP entry with DN='*string2*' during link check of object with DN = '*string3*': *string4*.

Explanation: While processing a change event for the object, the controller tries to read all referenced LDAP entries and received an exception different to NO_SUCH_OBJECT.

Action: Check the LDAP entries involved.

INF(CCC082): *string1* reads provision rule with DN = *string2*.

Explanation: The Event Maintenance Controller reads the provisioning rules to be applied on changed users.

Action: None.

WAR(CCC083): *string1* received *string2* during processing of *string3* for Entry with id = '*string4*': *string5*.

Explanation: The controller cannot process the event any more and tries to put it into the error channel.

Action: Check the message details.

INF(CCC084): *string1* puts an event for Entry with DN = '*string2*' to the retry channel after a *string3*: *string4*.

Explanation: The controller cannot process the event, but puts it into the retry channel for later retry.

Action: Check the message details.

INF(CCC085): *string1* puts an event for Entry with DN = '*string2*' to the error channel after a *string3*: *string4*.

Explanation: The controller cannot process the event and puts it into the error channel.

Action: Check the message details.

ERR(CCC086): A unexpected exception has occurred with storage id=*string1*: *string2*.

Explanation: A sporadic problem has occurred.

Action: Repeat processing of the object.

INF(CCC087): A new LDAP session with ID=*string1* is created.

Explanation: The workflow uses this session.

Action: None.

INF(CCC089): The LDAP session with ID=*string1* is reused.

Explanation: The workflow uses this session.

Action: None.

INF(CCC090): The LDAP session with ID=*string1* is returned to the pool.

Explanation: The workflow releases this session.

Action: None.

WAR(CCC091): *string1* during close of the job controller '*string2*': *string3*.

Explanation: The close method of the parent class of the event maintenance controller returned with an exception. It is ignored.

Action: Check for more detailed log messages of the parent controller.

INF(CCC092): Failure removing privilege with DN = '*string1*' from user with DN = '*string2*' based on rule with DN = '*string3*'.

Explanation: The user has the privilege and it must be removed due to the deny rule, but this failed. Reason is not known here.

Action: Check other error messages before this one and check also the user and the privilege objects.

INF(CCC200): *string1*

Explanation: Generic log message

Action: None.

INF(CCC202): Applying provisioning rules '*string1*' finished with *string2* fatalError(s), *string3* error(s), and *string4* warning(s).

Explanation: Applying provisioning rules finished.

Action: None.

INF(CCC207): Processing of user DN=*string1* ended with *string2* error(s), *string3* warning(s) at *string4*.

Explanation: Summary message for given user.

Action: None.

INF(CCC208): Changes in group resolution for user DN=*string1* successfully stored.

Explanation: Success message of user group resolution.

Action: None.

INF(CCC212): processModifiedUsers: Counter = *string1*.

Explanation: Given number of users to analyze were successfully handled.

Action: None.

ERR(CCC307): Invalid permission parameter "*string1*" .

Explanation: Given permission parameter of Group is not valid.

Action: Check given permission parameter of group. See proposal lists for valid values.

INF(CCC309): Inconsistent: user assigned but flagged as not needed.

Explanation: The account has a user assigned but is flagged as managed in target system only.

Action: Check account.

INF(CCC315): Account "*string1*" not element of TS.

Explanation: The given account is referenced by this group, but could not be found in the target system.

Action: Check the reference in group. Check account.

INF(CCC318): No junior role and no permission assigned.

Explanation: The role is not valid.

Action: Check role.

ERR(CCC319): Invalid value "*string1*" for property "*string2*".

Explanation: An invalid value for dxrState or dxrTSState attribute was detected.

Action: Check state attribute.

ERR(CCC320): BrokenReference obligation "*string1*" not existing.

Explanation: Group has a broken obligation link.

Action: Check dxrObligationLink attribute of group.

ERR(CCC321): BrokenReference obligation "*string1*" is not of type obligation.

Explanation: The dxrObligationLink attribute does not point to an obligation.

Action: Check dxrObligationLink attribute.

INF(CCC322): Account "*string1*" not element of TS; moved to delete.

Explanation: An account in groupmemberstate OK or ADD is missing. It is moved to DELETE state.

Action: None.

INF(CCC323): Inconsistent: no user assigned and not TS local.

Explanation: The account has no user assigned and is not flagged as managed in target system only.

Action: Check account.

ERR(CCC361): User to group resolution failed for user DN=*string1*: *string2*.

Explanation: An unexpected exception occurred during user to group resolution.

Action: Check the detailed message. Correct the error condition and resolve user manually in Identity Manager.

WAR(CCC362): No LDAP lock could be set for user '*string1*'. User will be skipped.

Explanation: There are problems when setting the LDAP lock..

Action: Check the message.

ERR(CCC400): *string1*

Explanation: Generic error message

Action: None.

INF(CCC400): Segregation of duty checks are enabled, but role resolution is disabled. Enabling role resolution.

Explanation: Role resolution is a precondition for segregation of duty checks. Thus, role resolution is enabled, despite it is disabled by the setting of the assignOnly option.

Action: Set assignment mode to assignAndResolve or disable dxrSodActive at the domain root.

INF(CCC500): User to group resolution started for user DN=*string1* at *string2*.

Explanation: Start message for user to group resolution.

Action: None.

16.5.4. Logging Messages (CPGCTRL)

This topic describes the logging messages returned by the DirX Identity Campaign Control.

INF(CPGCTRL000): *string1* The campaign controller open at '*string2*'.

Explanation: The campaign controller open.

Action: None.

INF(CPGCTRL001): *string1* The campaign controller start at '*string2*'.

Explanation: The campaign controller start.

Action: None.

INF(CPGCTRL002): *string1* The campaign controller close at '*string2*' - processTime for open-close: '*string3*' ms.

Explanation: The campaign controller close.

Action: None.

INF(CPGCTRL003): *string1* The LDAP session with ID=*string2* is returned to the pool.

Explanation: The workflow releases this session.

Action: None.

INF(CPGCTRL004): *string1* The LDAP session with ID=*string2* is reused.

Explanation: The workflow uses this session.

Action: None.

WAR(CPGCTRL005): The user hook with class name '*string2*' returned a '*string1*' when calling it's close method: *string3*.

Explanation: In the close method the user hook may terminate its resources.

Action: Check for more details the log messages of the user hook.

ERR(CPGCTRL006): *string1* Error starting campaign with cn=*string2*. Exception: *string3*.

Explanation: Campaign failed to start due to one or more invalid attribute values.

Action: Check values for attributes of the campaign entry.

ERR(CPGCTRL007): *string1* One or more campaigns failed to start. Exception: *string2*.

Explanation: Some campaigns contains invalid data.

Action: Check values for attributes of the campaign entries.

ERR(CPGCTRL008): *string1* Error on attempt to analyse campaign with cn=*string2*. Exception: *string3*.

Explanation: Campaign failed to be analysed due to invalid attribute values.

Action: Check values for attributes of the campaign entries.

ERR(CPGCTRL009): *string1* One or more campaigns failed to be analysed. Exception: *string2*.

Explanation: Some campaigns contains invalid data.

Action: Check values for attributes of the campaign entries.

ERR(CPGCTRL010): *string1* unsupported operation: '*string2*'.

Explanation: The controller does not implement this operation.

Action: Make sure the operation is not passed to the controller.

ERR(CPGCTRL011): Connection option missing: server = '*string1*', port = '*string2*', user = '*string3*', domain = '*string4*'.

Explanation: The <connection> element of the job controller does not contain all necessary parameters.

Action: Make sure all needed connection parameters are provided.

ERR(CPGCTRL012): *string1* caught exception *string2* while reading configuration options: '*string3*'.

Explanation: In its open() method the controller caught an exception.

Action: Make sure all needed configuration parameters are provided.

ERR(CPGCTRL013): *string1* when creating a session to Identity domain *string2*: *string3*.

Explanation: On start of the job the controller failed to clone an existing SvcSession or

create a new one. The job is terminated.

Action: Check availability of the LDAP server, address and bind credentials.

ERR(CPGCTRL014): User hook configuration for event controller '*string1*' does not contain a class name.

Explanation: The controller configuration contains a <userhook> element, but the class name for the user hook is missing.

Action: Enter a class name for the user hook or drop the <userhook> element.

ERR(CPGCTRL016): *string1* One or more campaigns failed to restart. Exception: *string2*.

Explanation: Some campaigns contains invalid data.

Action: Check values for attributes of the campaign entries.

INF(CPGCTRL017): *string1* Statistics: eventsProcessedOk = *string2*, eventsProcessedFailed = *string3*, updates = *string4*, upToDate = *string5*.

Explanation: The event controller informs about the number, type and result of processed events within one workflow run.

Action: None.

16.5.5. Logging Messages (CPGSVC)

This topic describes the logging messages returned by the DirX Identity Campaign Service.

INF(CPGSVC000): *string1* Campaign service instantiated with session id='*string2*'.

Explanation: The campaign service instantiated.

Action: None.

INF(CPGSVC001): *string1* Create campaign with name '*string2*' and owner '*string3*'.

Explanation: The campaign is created.

Action: None.

ERR(CPGSVC002): *string1* Invalid campaign data with name '*string2*' and owner '*string3*'. Exception: '*string4*'.

Explanation: An error occur when campaign was created.

Action: Check if campaign parameters are correct.

ERR(CPGSVC003): *string1* Campaign parameter '*string2*' is null or empty.

Explanation: A mandatory parameter for campaign is missing or is empty.

Action: Provide correct value for required parameter.

INF(CPGSVC004): *string1* Starting campaign '*string2*'.

Explanation: Campaign is starting.

Action: None.

ERR(CPGSVC005): *string1* Object from DN '*string2*' is not type SvcCertificationCampaign.

Explanation: The entry from LDAP is not a Certification Campaign entry.

Action: Check is object class values are correct.

INF(CPGSVC006): *string1* Campaign '*string2*' is started. Start date '*string3*', Expiration date '*string4*'.

Explanation: Campaign is started.

Action: None.

ERR(CPGSVC007): *string1* Campaign with cn '*string2*' could not be saved. Exception: '*string3*'.

Explanation: Campaign object could not be save to LDAP directory.

Action: Check if LDAP directory is available.

ERR(CPGSVC008): *string1* Campaign with cn '*string2*' could not be started due to missing or invalid parameters. Exception: '*string3*'.

Explanation: Some parameters are missing or values are invalid.

Action: Check logs above about details about missing or invalid parameters.

ERR(CPGSVC009): *string1* Error code from Session '*string2*' searching certifications with cn '*string3*'.

Explanation: Campaign certification search could not be performed to LDAP directory.

Action: Check if LDAP directory is available.

ERR(CPGSVC012): *string1* Could not read subject '*string2*' for campaign '*string3*', Exception: '*string4*'.

Explanation: Campaign certification service cannot read a subject from LDAP directory.

Action: Check if all DirX Identity components are available and subject exists.

ERR(CPGSVC013): *string1* Invalid campaign date '*string2*', Exception: '*string3*'.

Explanation: Campaign certification date is missing or has an invalid value.

Action: Provide a correct date (Generalized time format).

INF(CPGSVC014): *string1* Create certification entries for campaign: '*string2*' (type: '*string3*').

Explanation: Campaign service creates certification entries for subjects.

Action: None.

ERR(CPGSVC015): *string1* Invalid search base or search filter for campaign: '*string2*'.

Explanation: Campaign service failed to perform a search due to invalid search base or filter.

Action: Provide a valid search base and filter.

INF(CPGSVC016): *string1* Start Certification Campaign thread '*string2*' terminated successfully for campaign cn= '*string3*'.

Explanation: None.

Action: None.

ERR(CPGSVC017): *string1* Start Certification Campaign thread '*string2*' failed to terminate for campaign cn= '*string3*'. Exception: '*string4*'.

Explanation: Start Certification Campaign failed to terminate.

Action: Check the system load. Maybe the system is overloaded.

ERR(CPGSVC018): *string1* Executor Service Pool did not terminate correctly.

Explanation: Start Certification Campaign executor service pool failed to terminate.

Action: Check the system load. Maybe the system is overloaded.

ERR(CPGSVC019): *string1* User hook exception in '*string2*'. Exception: '*string3*'.

Explanation: Exception in campaign user hook.

Action: Check user hook implementation in provided location.

ERR(CPGSVC020): *string1* Expected campaign with cn '*string2*', but was returned node was null.

Explanation: Campaign entry not found in LDAP directory.

Action: Check if cn exist in LDAP.

INF(CPGSVC021): *string1* Apply changes Certification Campaign thread '*string2*' terminated successfully for campaign cn= '*string3*'.

Explanation: None.

Action: None.

ERR(CPGSVC022): *string1* Apply changes Certification Campaign thread '*string2*' failed to terminate for campaign cn= '*string3*'. Exception: '*string4*'.

Explanation: Apply changes Certification Campaign failed to terminate.

Action: Check the system load. Maybe the system is overloaded.

ERR(CPGSVC023): *string1* Executor Service Pool did not terminate correctly.

Explanation: Apply changes Certification Campaign executor service pool failed to

terminate.

Action: Check the system load. Maybe the system is overloaded.

ERR(CPGSVC024): *string1* Failed to change certification '*string2*' for campaign cn= '*string3*'. Exception: '*string4*'.

Explanation: Failed to apply changes to an entry in LDAP directory.

Action: Check if certification entry is available to LDAP directory.

ERR(CPGSVC025): *string1* Campaign cn='*string2*' has an invalid end time: '*string3*'. Exception: '*string4*'.

Explanation: Certification campaign has an invalid value for end time.

Action: Correct the Certification Campaign end date value.

ERR(CPGSVC026): *string1* Status Expiration Date is before end date. Reset to default value: End Date + '*string2*' days.

Explanation: Status Expiration Date is before Certification Campaign end date.

Action: Set a proper date after end date. Status Expiration Date must be after Certification Campaign end date.

ERR(CPGSVC027): *string1* Object from '*string2*' is not type SvcCertificationCampaign

Explanation: Delete object during the delete campaign operation cannot be performed because the object in not a campaign object.

Action: Check if entry is available valid and available to LDAP directory.

ERR(CPGSVC028): *string1* Object from '*string2*' could not be deleted (changed state to DELETED). Exception: '*string3*'.

Explanation: Failed to apply changes to an entry in LDAP directory.

Action: Check if entry is available to LDAP directory.

WAR(CPGSVC029): *string1* Campaign user hook class '*string2*' not found. Exception: ' *string3*'.

Explanation: User hook library file (jar) could not be find.

Action: Place the user hook library file in the correct location and restart the IDS-J server.

WAR(CPGSVC030): *string1* Invalid campaign user hook class '*string2*'. Exception: '*string3*'.

Explanation: User hook library file is not compatible with Certification Campaign controller

Action: Certification Campaign user hook must implement specific Java interfaces.

WAR(CPGSVC031): *string1* Cannot instantiate campaign user hook class '*string2*'. Exception: '*string3*'.

Explanation: Problems with main class of the user hook.

Action: Check constructor of the user hook implementation.

WAR(CPGSVC032): *string1* Current object '*string2*' does not have assignments.

Explanation: Object is not type dxrUser or derived type

Action: Enhance filter type by adding a proper object classes filter

ERR(CPGSVC033): *string1* Object from DN '*string2*' is not type SvcCertificationCampaign.

Explanation: The entry from LDAP is not a Certification Campaign entry.

Action: Check is object class values are correct.

ERR(CPGSVC034): *string1* Campaign with cn '*string2*' is could not be saved. Exception: '*string3*'.

Explanation: Campaign object could not be save to LDAP directory.

Action: Check if LDAP directory is available.

WAR(CPGSVC035): *string1* Campaign with cn '*string2*' does not have a start date.

Explanation: Campaign object does not have a campaign start date set.

Action: Check the campaign object and set a start date for the campaign.

WAR(CPGSVC036): *string1* For campaign with cn '*string2*' could not found the subjects folder.

Explanation: Subjects folder is missing. Certification campaign will attempt to restart the campaign without subjects folder backup.

Action: None.

ERR(CPGSVC037): *string1* Campaign with cn '*string2*' failed to start because due date is in the past.

Explanation: Certification campaign due date is already past and campaign should be automatically closed.

Action: Check start and due date values for Certification Campaign object.

ERR(CPGSVC038): *string1* Error in service for campaign '*string2*', Exception: '*string3*'.

Explanation: DirX Identity service failed with exception.

Action: Check if all DirX Identity components are available.

INF(CPGSVC099): *string1* The process subject '*string2*'.

Explanation: The campaign service process a user or a privilege.

Action: None.

16.5.6. Logging Messages (DCFG)

This topic describes the logging messages returned by the IDSJ extension for DomainConfiguration.

WAR(DCFG001): Couldn't initialize JMX support for extension '*string1*'.

Explanation: Extension won't be manageable via JMX due to initialization error.

Action: Inform the system administrator!

WAR(DCFG002): Couldn't initialize JMX support for repository '*string1*'

Explanation: Repository won't be manageable via JMX due to initialization error.

Action: Inform the system administrator!

16.5.7. Logging Messages (EBR)

This topic describes the logging messages returned by the DirX Identity Event Based Rules.

ERR(EBR000): *string1* unsupported operation: '*string2*'.

Explanation: The controller does not implement this operation.

Action: Make sure the operation is not passed to the controller.

INF(EBR001): *string1* received *string2* with id = '*string3*' and topic = '*string4*' from application '*string5*'.

Explanation: The activity reader has passed this event to the activity controller.

Action: None.

ERR(EBR002): Connection option missing: server = '*string1*', port = '*string2*', user = '*string3*', domain = '*string4*'.

Explanation: The <connection> element of the job controller does not contain all necessary parameters.

Action: Make sure all needed connection parameters are provided.

ERR(EBR004): Cannot bind to domain *string1* on '*string2*:_string3_' as '*string4* with ssl = '*string5*'; ErrorCode = *string6*.

Explanation: Controller cannot bind to DirX Identity domain.

Action: Check address and bind credentials. Is LDAP server available?

ERR(EBR005): Exception *string1* when loading configuration from domain '*string2*': *string3*.

Explanation: Controller cannot load configuration from DirX Identity domain.

Action: Check domain configuration: Object descriptions, etc.

WAR(EBR006): Exception *string1* when closing connection to domain '*string2*': *string3*.

Explanation: On close controller tried to dispose and disconnect the session to the DirX Identity domain.

Action: Check if the LDAP server is still available.

ERR(EBR007): Unable to read permission parameters from domain '*string1*': *string2*.

Explanation: Controller cannot attribute dxrRoleParams from domain root.

Action: Check domain root object for dxrRoleParams attribute.

INF(EBR008): No permission parameters configured for domain '*string1*'.

Explanation: The attribute dxrRoleParams of domain root is empty.

Action: None.

ERR(EBR009): Exception *string1* when reading permission parameters from domain '*string2*': *string3*.

Explanation: Controller cannot instantiate object for domain root and read attribute dxrRoleParams.

Action: Check domain root object for dxrRoleParams attribute.

WAR(EBR010): *string1* expected *string2*, but received *string3* with id = '*string4*'.

Explanation: The controller ignores this request.

Action: Check event to workflow dispatching.

INF(EBR011): *string1* resolves user with DN = '*string2*'.

Explanation: The user state, assigned groups and accounts are updated.

Action: None.

ERR(EBR012): Exception *string1* in *string2* while reading configuration options: *string3*.

Explanation: The activity controller tried to read configured options and address and bind credentials from LDAP reader.

Action: Check the details of the exception. Check the configuration options.

INF(EBR013): *string1* updates accounts of user with DN = '*string2*'.

Explanation: UserResolveController reads the accounts of the user and updates their mastered attributes.

Action: None.

INF(EBR014): *string1* updates account '*string2*' of user with DN = '*string3*'.

Explanation: UserResolveController updates the mastered attributes of this account.

Action: None.

WAR(EBR015): *string1* ignored *string2* with id = '*string3*'.

Explanation: The controller ignores some requests intentionally: e.g. delete.

Action: Check event to workflow dispatching, if event was not among the intentionally ignored.

ERR(EBR016): Exception *string1* while resolving user with DN = '*string2*': *string3*.

Explanation: The controller tries to read the user and resolve the privileges.

Action: Check the message details.

ERR(EBR017): Exception *string1* while updating accounts of user with DN = '*string2*': *string3*.

Explanation: The controller tries to update the accounts of the user.

Action: Check the message details.

ERR(EBR018): Exception *string1* while searching provisioning rules using filter = '*string2*': *string3*.

Explanation: The provision rule provider searches provisioning rules.

Action: Check the message details.

ERR(EBR019): initialization of provisioning rule *string1* failed: *string2*. Skipping processing of rule.

Explanation: The provision rule provider tried to read search filter, base and privileges from found provision rule entry.

Action: Check the message details.

ERR(EBR020): Exception *string1* while reading user with DN = '*string2*': *string3*.

Explanation: The controller tries to read the user from the domain using the identifier from the event.

Action: Check the message details.

WAR(EBR021): Exception *string1* while assigning privilege with DN = '*string2*' to user with DN = '*string3*' based on rule with DN = '*string4*': *string5*.

Explanation: The privilege cannot be assigned to the user. The controller continues with other rules and resolves and stores other modifications.

Action: Check the message details: not enough access rights, wrong privilege DN.

INF(EBR022): Removing privilege with DN = '*string1*' from user with DN = '*string2*' based on rule with DN = '*string3*'.

Explanation: The user has the privilege, but it must be removed due to the deny rule.

Action: None.

INF(EBR023): Assigning privilege with DN = '*string1*' to user with DN = '*string2*' based on rule

with DN = '*string3*'.

Explanation: The user is granted the privilege due to the grant rule.

Action: None.

ERR(EBR024): *string1* cannot get a session to Identity domain *string2*.

Explanation: On start of the job the controller failed to clone an existing SvcSession or create a new one. The job is terminated.

Action: Check availability of the LDAP server, address and bind credentials.

ERR(EBR025): Exception *string1* resolving user with DN = '*string2*' due to an addEvent: *string3*.

Explanation: After the controller received an addEvent it tried to resolve the user, but got a runtime exception.

Action: Check the message details.

ERR(EBR026): Exception *string1* resolving user with DN = '*string2*' due to a modifyEvent: *string3*.

Explanation: After the controller received a modifyEvent it tried to resolve the user, but got a runtime exception.

Action: Check the message details.

INF(EBR027): Exception *string1* while reading object of type *string2* with DN = '*string3*': *string4*.

Explanation: The controller tries to read the node from the domain using the identifier from the event.

Action: Check the message details. Does the object exist and is it of expected type? Is the class allowed to read the node?

ERR(EBR028): Exception *string1* processing modify event for object of type *string2* and DN = '*string3*': *string4*.

Explanation: After the controller received a modifyEvent it tried to read and process the node, but got a runtime exception.

Action: Check the message details.

ERR(EBR029): Exception *string1* when reading LDAP entry with DN='*string2*' during link check of object with DN = '*string3*': *string4*.

Explanation: While processing a change event for the object, the controller tries to read all referenced LDAP entries and received an exception different to NO_SUCH_OBJECT.

Action: Check the LDAP entry.

ERR(EBR030): The LDAP entry with DN='*string1*' referenced from object with DN='*string2*' in attribute '*string3*' does not exist.

Explanation: While processing a change event for the object, the controller tries to read all referenced LDAP entries to detect broken links and received an LDAP exception code NO_SUCH_OBJECT. The link is deleted.

Action: Check the LDAP entry.

ERR(EBR031): Exception *string1* while saving object with DN = '*string2*': *string3*.

Explanation: The controller tries to save changes to the object referenced in the change event.

Action: Check the message details and the object to be changed.

ERR(EBR032): Exception *string1* while processing an *string2*Event with requestID = '*string4*' for object of type *string3* with DN = '*string5*': *string6*.

Explanation: The event controller received an exception while processing the change event.

Action: Check the message details and the object which has been changed and processed.

ERR(EBR033): User hook configuration for event controller '*string1*' does not contain a class name.

Explanation: The controller configuration contains a <userhook> element, but the class name for the user hook is missing.

Action: Enter a class name for the user hook or drop the <userhook> element.

ERR(EBR034): Event controller '*string1*' cannot load user hook with class name '*string2*': *string3*.

Explanation: The class loader returns an exception when trying to load the class.

Action: Check the class name for the user hook in the <userhook> element and make sure the respective jar file is in the classpath.

ERR(EBR035): The user hook with class name '*string2*' returned a '*string1*' when calling it's open method: *string3*.

Explanation: In the open method the user hook evaluates its configuration and initializes itself.

Action: Check for more detailed log messages of the user hook. Check the user hook configuration.

WAR(EBR036): The user hook with class name '*string2*' returned a '*string1*' when calling it's close method: *string3*.

Explanation: In the close method the user hook may terminate its resources.

Action: Check for more detailed log messages of the user hook.

ERR(EBR037): The user hook with class name '*string3*' returned a '*string1*' when processing an *string2*Event for Entry with DN='*string4*': *string5*.

Explanation: The event controller passed the event and the affected LDAP entry to the user hook. It ignores the exception and stores it's own modifications.

Action: Check the user hook implementation and more detailed log messages of the user hook.

INF(EBR038): *string1* statistics: eventsProcessedOk = *string2*, eventsProcessedFailed = *string3*, updates = *string4*, upToDate = *string5*.

Explanation: The event controller informs about the number, type and result of processed events within one workflow run.

Action: None.

INF(EBR039): *string1* finished processing a *string2* for Entry with DN='*string3*:' *string4*ms.

Explanation: Time the event controller needed for processing this event.

Action: None.

ERR(EBR040): *string1* expected associated entry for '*string4*' of type *string2*, but received entry of class *string3*. DN or string representation of entry: *string5*.

Explanation: The event controller searched for associated objects and checked their types.

Action: Check the associated object: either remove the reference to the entry handled in the event or change to another type.

INF(EBR041): *string1* is updating associated entry of class *string2* with DN '*string3*.

Explanation: The event controller is going to change the associated entry, because a reference or other attribute needs to be deleted or changed.

Action: None.

ERR(EBR042): *string1* received *string2* when initializing rule context: *string3*.

Explanation: On startup the event controller searches consistency rules and initializes the rule context.

Action: Check the LDAP connection or inform DirX Identity Support about the problem.

ERR(EBR043): *string1* occurred when applying rule with DN='*string2*' to entry with DN='*string3*': *string4*.

Explanation: The event runs the consistency rule for the entry identified in the change event. It ignores the problem.

Action: Check the rule implementation.

WAR(EBR045): Expected consistency rule but obtained object with DN = '*string1*' of class '*string2*'.

Explanation: The rule provider searched for consistency rules. The returned object is a storage object but not consistency rule. It ignores this object.

Action: Check and correct the LDAP entry with the given DN.

ERR(EBR046): Skipping processing of consistency rule with name = '*string1*' after initialization failure: *string2*.

Explanation: The consistency rule provider tried to read search filter, base and operation from found consistency rule entry.

Action: Check the message details. Check the referenced operator and the filter definition.

ERR(EBR047): Exception *string1* while searching consistency rules using filter = '*string2*': *string3*.

Explanation: The consistency rule provider searches consistency rules, but did not get a result.

Action: Check the message details.

ERR(EBR048): Cannot read consistency rules due to missing session.

Explanation: The consistency rule provider was requested to read consistency rules for the first time or after timeout, but did not get a session with LDAP access.

Action: Internal problem. Inform DirX Identity support.

INF(EBR049): *string1* updated entry of class *string2* with DN '*string3*.

Explanation: The event controller changed the entry identified in the event and updated it in LDAP.

Action: None.

WAR(EBR050): Expected validation rule but obtained object with name = '*string1*' of class '*string2*'.

Explanation: The rule provider searched for validation rules (accept, cleanup, validate). The returned object is a storage object but not validation rule. It ignores this object.

Action: Check and correct the LDAP entry with the given DN.

ERR(EBR051): Skipping processing of validation rule with Name = '*string1*' after initialization failure: *string2*.

Explanation: The validation rule provider tried to read search filter, base and operation from found validation rule entry.

Action: Check the message details. Check the referenced operator and the filter definition.

WAR(EBR053): Exception *string1* when searching associated user for account with DN = '*string2*' and search filter = '*string3*': *string4*.

Explanation: The Account Event Controller searches the associated user for an account. It tries the next set of attributes.

Action: Check the message details and the search filter.

ERR(EBR054): Cannot read provisioning rules due to missing session.

Explanation: The provisioning rule provider was requested to read provisioning rules for the first time or after timeout, but did not get a session with LDAP access.

Action: Internal problem. Inform DirX Identity support.

ERR(EBR055): Exception *string1* when cloning session: '*string2*'.

Explanation: At the start of a new job the controller tries to clone an existing SvcSession object to the DirX Identity domain. The controller tries to create a new session.

Action: Check the message details. Check availability of the LDAP server.

INF(EBR056): Successfully cloned Session for domain *string1* and effective user with DN '*string2*'.

Explanation: The event controller cloned an existing session to Identity domain.

Action: None.

ERR(EBR057): Cannot read validation rules due to missing session.

Explanation: The validation rule provider was requested to read validation rules for the first time or after timeout, but did not get a session with LDAP access.

Action: Internal problem. Inform DirX Identity support.

INF(EBR058): Added *string1* rule to cache: *string2*.

Explanation: The validation rule provider read this rule and added it to its cache.

Action: None.

ERR(EBR059): Exception *string1* while searching validation rules using filter = '*string2*': *string3*.

Explanation: The validation rule provider searches validation rules, but did not get a result.

Action: Check the message details.

ERR(EBR060): *string1* while reading validation filter from group '*string2*': *string3*.

Explanation: The validation rule provider tries to get the validation filter for this group in order to check whether the imported or ignored group member matches the filter.

Action: Check the validation filter definition of the group.

INF(EBR061): Adding account '*string1*' to group '*string2*' as consequence of rule '*string5*'. Old state was: *string3*, new state = *string4*.

Explanation: The validation rule provider adds the account to the group because they match the rule subject and privilege filters respectively.

Action: None.

ERR(EBR062): *string1* while adding account '*string2*' to group '*string3*': *string4*.

Explanation: The validation rule provider could not add the account to the group or save the group. The problem is ignored.

Action: Check the message details and try to add the account manually.

ERR(EBR063): *string1* while reading user '*string2*' of account '*string3*': *string4*.

Explanation: The validation rule provider could not read the user with the given DN. The onAssignment rules for the account are applied without the user.

Action: Check the message details. Often the user is not needed for the operation.

INF(EBR064): Deleting account '*string1*' from group '*string2*' as consequence of rule ' *string4*'. Old state was: *string3*.

Explanation: The validation rule provider removes the account from the group because it is a cleanup rule or they do not match any more the rule subject and privilege filters of a validation rule respectively.

Action: None.

ERR(EBR065): *string1* while removing account '*string2*' from group '*string3*': *string4*.

Explanation: The validation rule provider could not remove the account from the group or save the group. The problem is ignored.

Action: Check the message details and try to remove the account manually.

ERR(EBR066): *string1* when checking validation filter of group with DN = '*string2*' for account with DN = '*string3*' and validation filter = '*string4*': *string5*.

Explanation: The validation rule provider checked if the account matches the validation filter of the group. It assumes there is no match.

Action: Check the message details. Maybe you have to change the validation filter.

ERR(EBR067): *string1* when checking validation filter of obligations referencing the group with DN = '*string2*' for account with DN = '*string3*':_string4_.

Explanation: The validation rule provider checked if the account matches the validation filter of the obligations referencing the group. It assumes there is no match.

Action: Check the message details. Maybe you have to change the obligations.

WAR(EBR068): *string1* when searching for groups with search base = '*string2*' and filter = '*string3*' obtained from rule '*string4*': *string5*.

Explanation: The validation rule provider searches for groups matching the privilege filter(s) of the validation rule. It ignores the offended filter and continues with the next one.

Action: Check the message details and the filters.

WAR(EBR069): *string1* when casting obligationLink = '*string2*' of group with DN = '*string3*' to an obligation: *string4*.

Explanation: The validation rule provider instantiates the obligation links of the group as a SvcObligation object. It ignores the link and continues with the next one.

Action: Check the message details and the obligation link.

WAR(EBR070): *string1* when evaluating validation filter of obligation with DN = '*string2*': *string3*.

Explanation: The validation rule provider transforms the String value in the validation attribute of the obligation to a LDAP filter. It ignores the filter.

Action: Check the message details and the filter.

WAR(EBR071): Obligation link value '*string1*' of group with DN = '*string2*' is neither a String nor an array of Strings or links to Storage Objects.

Explanation: The validation rule provider tries to interpret the value as an obligation link. It ignores the link.

Action: Check the message details and the link.

WAR(EBR072): *string1* when reading obligation with DN = '*string2*' of group with DN = '*string3*': *string4*.

Explanation: The validation rule provider reads the obligation from LDAP. It ignores the link.

Action: Check the message details and the entry with the given DN.

WAR(EBR073): *string1* when reading groups where account with DN = '*string2*' is a member in state IMPORTED: *string3*.

Explanation: Before applying accept/cleanup rules the validation rule provider checks if the account is an imported member of some group. It does not apply any accept/cleanup rule on the account.

Action: Check the message details. Is the referencing in the target system correct?

INF(EBR074): Updating membership state of account '*string1*' in group '*string2*' to *string3* based on rule '*string4*'.

Explanation: By applying the accept/cleanup rule the validation rule provider changed the member state of the account from IMPORTED to IGNORE resp. DELETE.

Action: None.

WAR(EBR075): *string1* when updating membership state of account '*string2*' in group '*string3*': *string4*.

Explanation: The validation rule provider tries to change the membership state from IMPORTED to IGNORE resp. DELETE and save the group if it is necessary.

Action: Check the message details. Try to change the state manually if still necessary.

INF(EBR076): Associated user '*string2*' with account '*string1*'.

Explanation: The Account Event Controller succeeded in finding a user for the account matching the configured account attributes.

Action: None.

WAR(EBR077): *string1* when passing failed event for entry with DN '*string2*' to *string3*: *string4*.

Explanation: The event controller failed to process the change event. It tried to pass the event together with an error response to either the retry or error connector.

Action: Check the configuration for the retry or error connector. Check if the identifier section of the event is correct.

ERR(EBR078): *string1* while searching with baseDN = '*string2*' and filter = '*string3*': *string4*.

Explanation: The Storage Utilities performed a LDAP search resulting in this exception. The search internally switches to paged mode in case of sizelimit reached.

Action: Check the message details.

ERR(EBR079): *string1* when creating a session to Identity domain *string2*: *string3*.

Explanation: On start of the job the controller failed to clone an existing SvcSession or create a new one. The job is terminated.

Action: Check availability of the LDAP server, address and bind credentials.

INF(EBR080): Running consistency rule '*string1*' for Entry *string2* ...

Explanation: The Event Maintenance Controller applies the consisteny rule to the given entry.

Action: None.

ERR(EBR081): *string1* when reading LDAP entry with DN='*string2*' during link check of object with DN = '*string3*': *string4*.

Explanation: While processing a change event for the object, the controller tries to read all referenced LDAP entries and received an exception different to NO_SUCH_OBJECT.

Action: Check the LDAP entries involved.

INF(EBR082): *string1* reads provision rule with DN = *string2*.

Explanation: The Event Maintenance Controller reads the provisioning rules to be applied on changed users.

Action: None.

WAR(EBR083): *string1* received *string2* during processing of *string3* for Entry with id = '*string4*': *string5*.

Explanation: The controller cannot process the event any more and tries to put it into

the error channel.

Action: Check the message details.

INF(EBR084): *string1* puts an event for Entry with DN = '*string2*' to the retry channel after a *string3*: *string4*.

Explanation: The controller cannot process the event, but puts it into the retry channel for later retry.

Action: Check the message details.

INF(EBR085): *string1* puts an event for Entry with DN = '*string2*' to the error channel after a *string3*: *string4*.

Explanation: The controller cannot process the event and puts it into the error channel.

Action: Check the message details.

ERR(EBR086): A NullPointerException occurred with storage id=*string1*: *string2*.

Explanation: A class tried to access an object which is null. This message gives the information whether a SvcSession was available.

Action: If there was no SvcSession, repeating the operation might help. Otherwise check the source to fix the NPE.

INF(EBR087): A new LDAP session with ID=*string1* is created.

Explanation: The workflow uses this session.

Action: None.

INF(EBR089): The LDAP session with ID=*string1* is reused.

Explanation: The workflow uses this session.

Action: None.

INF(EBR090): The LDAP session with ID=*string1* is returned to the pool.

Explanation: The workflow releases this session.

Action: None.

WAR(EBR091): *string1* during close of the job controller '*string2*': *string3*.

Explanation: The close method of the parent class of the event maintenance controller returned with an exception. It is ignored.

Action: Check for more detailed log messages of the parent controller.

INF(EBR092): Failure removing privilege with DN = '*string1*' from user with DN = '*string2*' based on rule with DN = '*string3*'.

Explanation: The user has the privilege and it must be removed due to the deny rule, but this failed. Reason is not known here.

Action: Check other error messages before this one and check also the user and the privilege objects.

ERR(EBR093): Reading the object by its DN=*string1* failed: Object does not exist. A second attempt to read the object by its uid with filter=*string2* failed again: '*string3*'.

Explanation: The controller tries to read the node from the domain using the identifier from the event. Then it tries to read the object by its dxrUid.

Action: Check the message details. Does the object exist and is it of expected type? Is the class allowed to read the node? Does more than one object exist with the referenced dxrUid?

ERR(EBR094): Reading the object by its DN=*string1* failed: Object does not exist and has no dxrUid.

Explanation: The controller tries to read the node from the domain using the identifier from the event. An attempt to read it by its dxrUid was not made since the object has no dxrUid.

Action: Run the dxrUid migration to create missing uids.

ERR(EBR095): Entry '*string1*' is of type *string2*, but expected *string3*.

Explanation: The controller read the entry with the given DN from LDAP and tried to cast it to the expected class, but received a ClassCastException.

Action: Probably the event is sent to the wrong queue.

WAR(EBR096): Entry '*string1*' is locked and will currently not be processed.

Explanation: The user is locked as it is used by another application.

Action: Try later again.

INF(EBR097): No provisioning rules will be applied for user='*string1*' because no search base for provisioning rules is set in the activity of the eventmaintenace workflow.

Explanation: There is no value configured for the searchbase for provisioning rules.

Action: None.

16.5.8. Logging Messages (ELM)

This topic describes the logging messages returned by the DirX Identity Entry Lock Manager.

INF(ELM101): LockManager - The lock with the ID '*string1*' for object '*string2*' cannot be deleted because it was not found.

Explanation: The lock that should be deleted does not exist anymore.

Action: None.

INF(ELM102): LockManager - LDAP exception, could not read the specified LDAP lock entry:

'*string1*' (uid: *string2*).

Explanation: The LDAP lock entry that is tried to be read might not exist in LDAP.

Action: Check LDAP to ensure the entry exists.

INF(ELM103): LockManager - LDAP exception, could not modify the specified LDAP lock entry: '*string1*'; error: *string2*.

Explanation: The LDAP lock entry that should be modified might was already changed by some other thread/process.

Action: None. Another process probably has changed the entry and therefore that update fails due to an out of date modification.

INF(ELM104): LockManager - The entry '*string1*' is locked by someone else until *string2* (yyyyMMddHHmmss.mmmZ).

Explanation: Could not lock the specified workflow because it is already locked by someone else.

Action: Try again later.

INF(ELM105): LockManager - The user '*string1*' is locked by someone else until *string2* (yyyyMMddHHmmss.mmmZ).

Explanation: Could not lock the specified workflow because it is already locked by someone else.

Action: Try again later.

INF(ELM106): LockManager - The workflow '*string1*' is locked by someone else until *string2* (yyyyMMddHHmmss.mmmZ).

Explanation: Could not lock the specified workflow because it is already locked by someone else.

Action: Try again later.

INF(ELM127): LockManager - LDAP exception, could not create the specified LDAP lock entry: '*string1*'; error: *string2*.

Explanation: The LDAP lock entry that should be modified might was already changed by some other thread/process.

Action: None. Another process probably has changed the entry and therefore that update fails due to an out of date modification.

INF(ELM129): LockManager - LDAP exception, could not delete the specified LDAP lock entry: '*string1*'; error: *string2*.

Explanation: The LDAP lock entry that should be modified might was already changed by some other thread/process.

Action: None. Another process probably has changed the entry and therefore that update fails due to an out of date modification.

INF(ELM130): LockManager - The entry '*string1*' is locked by someone else.

Explanation: Could not lock the specified entry because it is already locked by someone else.

Action: Try again later.

INF(ELM131): LockManager - The user '*string1*' is locked by someone else.

Explanation: Could not lock the specified user because it is already locked by someone else.

Action: Try again later.

INF(ELM132): LockManager - The workflow '*string1*' is locked by someone else.

Explanation: Could not lock the specified workflow because it is already locked by someone else.

Action: Try again later.

WAR(ELM201): EntryLockManager - LDAP exception, connection could not be established using the following data: server - *string1* , port - *string2* , user - *string3*.

Explanation: Could not establish a connection to LDAP with the given credentials.

Action: Check the validity of the credentials and try again.

WAR(ELM202): EntryLockManager - Cannot return LDAP connection '*string1*' to connection pool *string2*: *string3*.

Explanation: The LDAP connection could not be returned to he connection pool.

Action: A new connection will be created.

WAR(ELM203): EntryLockManager - Cannot create/add pool specification: '*string1*'.

Explanation: The LDAP connection pool could not be created.

Action: Provide your own LDAP connection or try again later.

WAR(ELM204): EntryLockManager - LDAP exception, could not read the specified LDAP lock entry by uid: '*string1*'.

Explanation: The LDAP lock entry that is tried to be read might not exist in LDAP.

Action: Check LDAP to ensure the entry exists.

WAR(ELM205): EntryLockManager - LDAP connection is not authenticated.

Explanation: There is only an anonymous LDAP connection used.

Action: Contact the DirX Identity support.

WAR(ELM206): EntryLockManager created a new LDAP connection but that connection is not authenticated.

Explanation: There is only an anonymous LDAP connection used.

Action: Contact the DirX Identity support.

16.5.9. Logging Messages (ICF)

This topic describes the logging messages returned by the Java-based DirX Identity server framework job integration.

INF(ICF200): Starting job to *string1* collection(s).

Explanation: The import or export of collection(s) is started..

Action: None.

WAR(ICF400): Response ID=*string1* with ResultCode=SUCCESS has been written to the error channel.

Explanation: The error channel is intended to hold requests with ResultCode=FAILURE.

Action: With DirX Identity standard software, this error should not occur. Call DirX Identity support. With other software, check the piece of code that writes to the error channel.

WAR(ICF401): Response ID=*string1* with ResultCode=TEMP_ERR has been written to the error channel.

Explanation: The error channel is intended to hold requests with ResultCode=FAILURE. Requests with ResultCode=TEMP_ERR have to be written to the retry channel.

Action: With DirX Identity standard software, this error should not occur. Call DirX Identity support. With other software, check the piece of code that writes to the error channel.

WAR(ICF402): Response ID=*string1* with ResultCode=SUCCESS has been written to the retry channel.

Explanation: The retry channel is intended to hold requests with ResultCode=TEMP_ERR.

Action: With DirX Identity standard software, this error should not occur. Call DirX Identity support. With other software, check the piece of code that writes to the retry channel.

WAR(ICF403): Response ID=*string1* with ResultCode=FAILURE has been written to the retry channel.

Explanation: The retry channel is intended to hold requests with ResultCode=TEMP_ERR. Requests with ResultCode=FAILURE have to be written to the error channel.

Action: With DirX Identity standard software, this error should not occur. Call DirX Identity support. With other software, check the piece of code that writes to the retry channel.

WAR(ICF410): Job completed with temporary error: *string1*.

Explanation: The server will repeat the activity until its retry limit is reached or the task is successfully completed.

Action: Check and repair the source of the temporary error (server/network connection).

WAR(ICF411): Job has temporary errors.

Explanation: The server will repeat the activity until its retry limit is reached or the task is successfully completed.

Action: Check and repair the source of the temporary error (server/network connection).

WAR(ICF412): Job finished with errors: *string1*.

Explanation: Check the detailed messages of the job's run.

Action: Check and repair the source of the errors.

WAR(ICF413): Job finished with errors.

Explanation: Check the detailed messages of the job's run.

Action: Check and repair the source of the errors.

WAR(ICF414): Exception occurred while processing job: *string1*.

Explanation: Processing of the job has been aborted due to a severe error.

Action: Check the detailed messages of the job's run. Call DirX Identity support.

WAR(ICF415): Exception occurred while processing job.

Explanation: Processing of the job has been aborted due to a severe error.

Action: Check the detailed messages of the job's run. Call DirX Identity support.

WAR(ICF416): Filtered illegal XML character '*string1*' from audit message '*string2*'.

Explanation: An illegal xml character was detected and filtered from the message string.

Action: None.

WAR(ICF417): No crypt support available. Could not sign audit record.

Explanation: No crypt support for signing was found.

Action: Check log files. Check the XML configuration for the AuditChannelConnector class. Check existence of the PSE in data view.

WAR(ICF418): Writing to channel '*string1*' interrupted due to activity timeout: *string2*.

Explanation: Writing to channel failed due to activity timeout. Job aborted.

Action: Check previous messages. This activity has been cancelled because of timeout.

WAR(ICF419): Event channel reader terminates due to Interrupt exception (timeout).

Explanation: The thread will be aborted by server due to timeout.

Action: Increase time out for realtime workflow / activities.

ERR(ICF500): Validation errors occurred while reading event: *string1*.

Explanation: The format of the event is invalid, data are missing or may have wrong values. With DirX Identity standard software, the error should not occur.

Action: Check the detailed message. Check the source that created the event. Call DirX Identity support.

ERR(ICF501): *string1* while reading event from channel: *string2*.

Explanation: The format of the event is invalid, data are missing or may have wrong values. With DirX Identity standard software, the error should not occur.

Action: Check the detailed message. Check the source that created the event. Call DirX Identity support.

ERR(ICF502): Exception while processing event: *string1*.

Explanation: An exception occurred while processing event. With DirX Identity standard software, the error should not occur.

Action: Check the detailed message. Check the source that created the event. Call DirX Identity support.

ERR(ICF503): Error in *string1*.add(): No associated IDMRequest in context for response ID=*string2*.

Explanation: The IDMRequest to be written to the error channel does not exist. Check if connector is run in the Java-based DirX Identity server.

Action: Do not use the connector in a standalone environment. It is intended for use in the Java-based DirX Identity server. Call DirX Identity support.

ERR(ICF504): Error reading from channel *string1*: No event contained in request with UID=*string1*.

Explanation: The IDMRequest does not contain an event. With DirX Identity standard software, the error should not occur.

Action: Check the source of the request. Call DirX Identity support.

ERR(ICF505): Error reading event uid=*string1* from channel*string2*: Classloading failed, could not construct 'com.siemens.idm.framework.nodes.IDMSpmlEvent'.

Explanation: The Java class of the event could not be constructed.

Action: This error may occur in case jar files are installed in wrong places. As a workaround, run both activities being connected by the channel collocated only. Call DirX Identity support.

ERR(ICF506): Generation of XML failed for audit entry: *string1*.

Explanation: The audit entry could not be converted to XML format. With DirX Identity standard software, the error should not occur.

Action: Check the code writing to the audit connector. Call DirX Identity support.

ERR(ICF507): Validation errors occurred while reading SPML request: *string1*.

Explanation: The format of the SPML request is invalid, data are missing or may have wrong values. With DirX Identity standard software, the error should not occur.

Action: Check the detailed message. Check the source that created the request. Call DirX Identity support.

ERR(ICF508): *string1* while unmarshalling SPML request from event: *string2*.

Explanation: The format of the SPML request is invalid, data are missing or may have wrong values. With DirX Identity standard software, the error should not occur.

Action: Check the detailed message. Check the source that created the request. Call DirX Identity support.

ERR(ICF509): Exception while processing SPML request: *string1*.

Explanation: An exception occurred while processing an SPML request. With DirX Identity standard software, the error should not occur.

Action: Check the detailed message. Check the source that created the request. Call DirX Identity support.

ERR(ICF510): Controller does not implement SPMLv1 requests: *string1*.

Explanation: The reader expects a controller implementation that implements the DxmController interface and accepts SPML v1 requests.

Action: Configure an appropriate controller class.

ERR(ICF511): Exception while marshalling audit record: *string1*.

Explanation: An exception occurred while processing an audit record. With DirX Identity standard software, the error should not occur.

Action: Check the detailed message. Check the source that created the request. Call DirX Identity support.

ERR(ICF512): Exception while signing audit record: *string1*.

Explanation: An exception occurred while creating a signature for audit record.

Action: Check the detailed message. Check the source that created the request. Check the configuration of the audit channel connector. Call DirX Identity support.

ERR(ICF513): Exception while sending notification request: *string1*.

Explanation: An exception occurred while creating the e-mail object or while sending

the e-mail itself.

Action: Check the detailed message.

ERR(ICF514): EventChannelConnector: no JMS_Context found in context.

Explanation: EventChannelConnector can not find context to get a connection to a JMS provider.

Action: Call DirX Identity support.

ERR(ICF515): EventChannelConnector: no operational attribute 'topic' in SPML request: '*string1*'.

Explanation: In the event must be a topic included in the operational attributes otherwise no JMS event can be sent.

Action: Call DirX Identity support.

ERR(ICF516): JMSException while sending setPassword event in EventChannelConnector: '*string1*'.

Explanation: Event can not be sent.

Action: Call DirX Identity support.

ERR(ICF517): Cannot serialize SPML request in EventChannelConnector: request = '*string1*'.

Explanation: Event can not be sent.

Action: Call DirX Identity support.

ERR(ICF518): Cannot create publisher in EventChannelConnector: topic = '*string1*'.

Explanation: Event can not be sent.

Action: Call DirX Identity support.

ERR(ICF519): Cannot create session in EventChannelConnector: message service to connect to '*string1*'.

Explanation: Event can not be sent.

Action: Check message service.

ERR(ICF520): JMSException while creating JMS message: '*string1*'.

Explanation: Message can not be created.

Action: Call DirX Identity support.

ERR(ICF521): Cannot create session for JMS notification: message service to connect to '*string1*'.

Explanation: Message can not be sent.

Action: Check message service.

FTL(ICF900): Error while reading from channel '*string1*': Event is of wrong type '*string2*'. Expecting IDMRequest.

Explanation: An event is read from the input channel that is not an IDM event. The input channel is connected to an output channel delivering objects of wrong type.

Action: Correct Your workflow configuration.

FTL(ICF901): Reading from channel '*string1*' failed: *string2*.

Explanation: An exception occurred while reading from input channel. Processing not possible.

Action: Check detailed message and previous log messages especially for detected timeout for activity. If you find this then it is the interrupt of the thread (no action required) otherwise call DirX Identity support.

FTL(ICF902): Cannot get channel '*string1*'.

Explanation: The channel name is not available to the framework connector.

Action: Please check your workflow configuration.

FTL(ICF903): Open of channel connector failed, class=*string1*. Error in context/environment: *string2*.

Explanation: The channel reader is used in the wrong context, has to be used in Java-based DirX Identity server.

Action: Do not use the connector in a standalone environment. It is intended for use in the Java-based DirX Identity server. Call DirX Identity support.

FTL(ICF904): Error while writing to channel '*string1*': *string2*.

Explanation: Writing to channel failed. Job aborted.

Action: Check detailed message. Call DirX Identity support.

FTL(ICF905): No type property defined for audit connector *string1*.

Explanation: Type property is missing in the configuration of the audit connector.

Action: Add the type property to the configuration of the audit connector.

FTL(ICF906): NullPointerException while writing to channel '*string1*': *string2*.

Explanation: Writing to channel failed. A NullpointerException occurred. Job aborted.

Action: Check detailed message. Call DirX Identity support.

FTL(ICF940): <transport> element missing in job configuration.

Explanation: The transport element is missing in the job configuration. No transport job is started.

Action: Correct the job configuration. Create a workflow from a default transport workflow.

FTL(ICF941): <operation> element missing in job configuration.

Explanation: The operation element is missing in the job configuration. No transport job is started.

Action: Correct the job configuration. Create a workflow from a default transport workflow.

FTL(ICF942): Illegal value '*string1*' for <operation> in job configuration, expecting 'import' or 'export'.

Explanation: The operation element has a wrong value in the job configuration. No transport job is started.

Action: Correct the job configuration. Create a workflow from a default transport workflow.

FTL(ICF943): Transforming the <transport> section to a standard job configuration failed: *string1*.

Explanation: The configuration could not be transformed to the standard job configuration. No transport job is started.

Action: Correct the job configuration. Create a workflow from a default transport workflow.

16.5.10. Logging Messages (IDSJ)

This topic describes the logging messages returned by the DirX Identity Server.

WAR(IDSJ001): Couldn't initialize JMX support for '*string1*'.

Explanation: Server won't be manageable via JMX due to initialization error.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ002): Failed to get the machine's host address (cause: *string1*)

Explanation: Network support not configured properly or no network card detected.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ003): Couldn't initialize CryptSupport due to problems.

Explanation: Possible reasons: Invalid or non-existing certificate/PSEs (see LDAP configuration), wrong PINs (see install_path/ssl/password.properties)

Action: Provide suitable certificates/PSEs and PINs. If this doesn't help, inform DirX Identity technical support about the problem.

WAR(IDSJ004): CryptSupport not initialized - no PIN available!

Explanation: Private keys (PSEs) are protected by a PIN which must be provided in install_path/ssl/password.properties.

Action: Add a suitable PIN to install_path/ssl/password.properties.

ERR(IDSJ005): Decryption of '*string1*' failed.

Explanation: Failed to decrypt an encrypted placeholder.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ006): Initialization of JMX registry failed.

Explanation: The server's JMX registry doesn't work properly due to initialization error.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ007): Internal error (cause: *string1*).

Explanation: Server encountered severe difficulties.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ008): Internal error (cause: *string1*)

Explanation: Server encountered fixable difficulties.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ009): Thread *string1* terminated unexpectedly.

Explanation: Server encountered severe difficulties, a component (e.g. thread) terminated unexpectedly.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ010): Failed to setup a SocketFactory for SSL at port *string1*: *string2*.

Explanation: SSL will not work properly, since server/worker encountered severe difficulties while initializing the SocketFactory.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ011): Failed to add '*string1*' to the classloader *string2* (cause: *string3*).

Explanation: A classloader encountered serious problems while adding a jar-file to its internal repository.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ012): Failed to initialize extension '*string1*' (cause: *string2*).

Explanation: Due to an error, the mentioned server extension couldn't be initialized. As a consequence, the respective functionality won't be available.

Action: Check your installation, inform DirX Identity technical support about the problem.

INF(IDSJ013): Embedded Tomcat deploys extension *string1*:_string2_

Explanation: Tomcat is currently loading a webapp provided by a server extension.

Action: None.

WAR(IDSJ014): Initialization extension '*string1*' failed (cause: *string2*).

Explanation: Extension folder exists (see extensions/), but presumably doesn't contain jar files.

Action: Check your installation, inform DirX Identity technical support about the problem.

INF(IDSJ015): Configuration of extension '*string1*' failed.

Explanation: Extension reported that it couldn't initialize properly.

Action: None.

ERR(IDSJ016): Failed to load locally stored settings (cause: *string1*).

Explanation: Server encountered problems while loading local settings from directory install_path/ids-j-<domain>-S<number>/confdb/server/local

Action: Check access rights.

INF(IDSJ017): *string1* worker threads started.

Explanation: This message indicates how many worker threads have been started.

Action: None.

INF(IDSJ018): Starting IDMServer.

Explanation: Server is currently initializing.

Action: None.

INF(IDSJ019): Stopping IDM service now.

Explanation: Server is currently going down.

Action: None.

INF(IDSJ020): Loading configuration from *string1*.

Explanation: Server is currently loading a server extension.

Action: None.

INF(IDSJ021): Starting RMI registry at port *string1*.

Explanation: Server is currently starting RMI registry in order to communicate with remote worker containers.

Action: None.

INF(IDSJ022): Lookup RMI registry at *string1*:_string2_.

Explanation: Remote worker container is currently trying to connect to RMI registry.

Action: None.

INF(IDSJ023): starting MX4J http adaptor at *string1*:_string2_.

Explanation: JMX implementation 'MX4J http adaptor' is being started.

Action: None.

WAR(IDSJ024): Couldn't initialize AWT. Servlets delivering graphical content might not work correctly (cause *string1*).

Explanation: If running on a Linux platform, server needs a graphical subsystem to deliver graphical content.

Action: Install graphical subsystem.

INF(IDSJ025): Server state will be written every *string1* seconds to file *string2*.

Explanation: Indicates when the server state will be written to a file.

Action: None.

INF(IDSJ026): Registered UserMapping *string1*.

Explanation: Server deployed a UserMapping (identifier -> DN).

Action: None.

ERR(IDSJ027): Failed to register UserMapping *string1* (cause: *string2*).

Explanation: UserMapping inoperable. SOAP services will experience problems when authenticating.

Action: If this is a 3rd party UserMapper, contact provider. Otherwise inform DirX Identity technical support about the problem.

WAR(IDSJ028): Problems while initializing LDAP realm for authentication (cause: *string1*).

Explanation: Users of the server's Web Admin might experience problems when logging in.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ029): Cannot add workflow schedules to space: *string1*.

Explanation: Probably memory or other internal problem.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ030): Failed to export repositories (cause *string1*).

Explanation: Export of repositories (triggered by switch exportOnShutdown=true, see install_path/ids-j-<domain>-S<number>/confdb/GlobalContext.properties).

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ031): Failed to import repositories (cause *string1*).

Explanation: Import of formerly exported repositories failed.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ032): Failed to transform repository *string1* (cause *string2*).

Explanation: Repository cannot be transformed to new format.

Action: Unsuitable repository format, inform DirX Identity technical support about the problem.

INF(IDSJ033): Export operation processed *string1* repositories.

Explanation: Number of exported adaptor-repositories.

Action: None.

INF(IDSJ034): Import operation processed *string1* repositories.

Explanation: Number of imported adaptor-repositories.

Action: None.

INF(IDSJ035): Starting embedded Tomcat.

Explanation: Server is currently initializing.

Action: None.

INF(IDSJ036): Starting to read workflow definitions (Connectivity).

Explanation: Server is currently initializing.

Action: None.

INF(IDSJ037): Starting to read workflow definitions (Provisioning).

Explanation: Server is currently initializing.

Action: None.

INF(IDSJ038): Starting JMX HTTP adaptor.

Explanation: Server is currently initializing.

Action: None.

INF(IDSJ039): Starting component *string1*.

Explanation: Server is currently initializing.

Action: None.

INF(IDSJ040): No transformation from version *string1* to version *string2* provided. File *string3* won't be transformed (reason *string4*).

Explanation: If repositories are incompatible between versions, a transformer should have been provided.

Action: None.

WAR(IDSJ041): No LDAP reader found for reading nationalization strings.

Explanation: LDAP reader for 'domain' access is not available.

Action: Check the XML server configuration.

WAR(IDSJ042): Creation of properties file '*string1*' (or superior directory) failed; error: *string2*.

Explanation: Creation of nationalized properties file failed.

Action: Check the error reason.

INF(IDSJ043): Extension '*string1*' started.

Explanation: Server is currently initializing.

Action: None.

INF(IDSJ044): Reading workflow definitions in folder '*string1*' failed: *string2*.

Explanation: At least one workflow definition couldn't be read.

Action: Check the given reason.

INF(IDSJ045): Starting worker *string1*.

Explanation: This message indicates which worker thread gets started.

Action: None.

WAR(IDSJ046): Cannot authenticate user '*string1*' (password length *string2*) against *string3*:_string4_ (cause: *string5*).

Explanation: A request cannot processed because the associated user cannot be authenticated.

Action: The caller is either directly a user using the WebCenter or a technical user internally used by the Web Center.

WAR(IDSJ047): Cannot read the domain configuration of '*string1*'.

Explanation: See the detailed message. Internally the dxrTopicIncludesDomain cannot be evaluated. The server assumes it is set to FALSE.

Action: Check the detailed message and act accordingly. If the attribute is really TRUE restart the IdS-J after fixing the access.

WAR(IDSJ048): Cannot create or access configured log path for audit listener '*string1*' cause: *string2*. Using default path.

Explanation: The given log file path '*string1*' can not be created or accessed. Instead, the server uses a default folder AuditTrailDB (for file-based auditing) or logs/AuditTrail (for JMS-based auditing), respectively. In this case, said default folder location is relative to install_path/ids-j-<domain>-S<number>.

Action: Correct the configuration or the access rights.

WAR(IDSJ049): Cannot find the attribute '*string1*' in the server configuration entry.

Explanation: The given attribute is missing. Some features which are dependable may be not work correctly.

Action: Correct the attribute in the IDS-J server entry.

WAR(IDSJ050): Cannot create script home directory: *string1*.

Explanation: The server could not create the script home directory underneath the repository directory.

Action: Check the error message.

ERR(IDSJ051): Execution of 'isRunnable' implementation failed.

Explanation: Erroneous Java script or java-class, invalid definition (e.g. missing class name).

Action: Check workflow definition.

ERR(IDSJ052): Execution of 'notRunnable' implementation failed.

Explanation: Erroneous Java script or java-class, invalid definition (e.g. missing class name).

Action: Check workflow definition.

ERR(IDSJ053): Activity '*string1*': Error in channel mapping, *string2* doesn't exist.

Explanation: Channel mapping refers a non-existing channel.

Action: Check workflow definition.

ERR(IDSJ054): Inconsistent channel mapping, no activity '*string1*' found!

Explanation: Channel mapping refers a non-existing activity.

Action: Check workflow definition.

WAR(IDSJ055): FinalizeChannels: Channel '*string1*' in activity '*string2*' of workflow '*string3*' doesn't exist!

Explanation: A channel which must be closed doesn't exist (any more).

Action: None.

WAR(IDSJ056): Activity '*string1*' will not be executed since workflow is expired! Running time of this workflow: *string2*, workflow expired since: *string3* (both sec).

Explanation: Workflow has expired. No more activities will be started.

Action: None.

INF(IDSJ057): Aborting activity '*string1*'.

Explanation: Worker container received an abort request or timeout occurred. Activity is being aborted.

Action: None.

WAR(IDSJ058): Detected timeout for activity '*string1*', trying to abort it.

Explanation: Currently running activity will be aborted.

Action: Increase timeout values of workflow/activity definition.

WAR(IDSJ059): Prepare for activity '*string1*' failed, set state to *string2*, applicationState to *string3* (cause: *string4*).

Explanation: Failed to set state and applicationState.

Action: Check workflow definition.

WAR(IDSJ060): For the resource family '*string1*' of the activity '*string2*' (workflow '*string3*') no worker thread is configured. The task is written to Java Space anyway.

Explanation: The resource family indirectly determines on which servers or worker containers an activity may be run. Make sure that for each activity there is at least one system associated with the same resource family.

Action: In the server's Web Admin interface you can add worker threads for a resource family. Or you can add resource families in the Identity Manager. The later requires a restart of the IdS-J server.

ERR(IDSJ061): Cannot read activity '*string1*' definition of workflow '*string2*'.

Explanation: Activity task cannot read the activity definition.

Action: Check workflow definition and/or your LDAP server.

WAR(IDSJ062): *string1* PSE '*string2*' not usable.

Explanation: PSEs and PINs are not matching. See the detailed message.

Action: Correct PSE and PIN configuration.

WAR(IDSJ063): The configured *string1* PSE doesn't work properly.

Explanation: The server tried an encrypt/decrypt cycle with the configured PSE. This failed. See the detailed message.

Action: Correct PSE and PIN configuration.

WAR(IDSJ064): The configured *string1* PSE doesn't work properly. Used certificate: *string2*.

Explanation: The server tried an encrypt/decrypt cycle with the configured PSE. This failed. See the detailed message.

Action: Correct PSE and PIN configuration.

WAR(IDSJ065): During startup several warnings occurred during initializing PSE/cryptographic support. See 'start.log' for more details.

Explanation: Configuring PSE environment failed.

Action: See start.log for reason and actions.

WAR(IDSJ066): Cleanup of timer tasks failed (cause: *string1*).

Explanation: Periodically cancelled timer task are cleaned up. Encountered a problem while cleaning up, see error cause for further information.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ067): Space usage checking for partition named by this file path '*string1*' failed.

Explanation: Observing the usable space in the partition named by the given file path cannot be done.

Action: Check the given exception.

ERR(IDSJ068): Very low space (*string2*% used) in the partition named by this file path '*string1*'. Blocking all JMS message-based adaptors.

Explanation: Used space is above the high water mark for the given partition. No events are accepted anymore.

Action: Clean up the partition.

INF(IDSJ069): Enough space (*string2*% used) now in the partition named by this file path '*string1*'. JMS adaptors are unblocked now.

Explanation: Used space fall under the high water mark. JMS adaptors are unblocked now.

Action: Check the partition if you still need to clean up.

WAR(IDSJ070): Low space (*string2*% used) in the partition named by this file path '*string1*'.

Explanation: Used space is above the low water mark for the given partition.

Action: Clean up the partition.

WAR(IDSJ071): Detected timeout for activity '*string1*'. Set cancellation flag for graceful abort.

Explanation: Worker thread received an abort request or timeout occurred. Activity is flagged to aborted.

Action: Eventually increase activity timeout value.

INF(IDSJ072): Worker threads all stopped.

Explanation: All the worker threads for the resource families have been stopped.

Action: None.

ERR(IDSJ073): Failed to process WebCenter authorization data (auth. type: *string1*, subject DN: *string2*, remote host: *string3*, remote user *string4*).

Explanation: An HTTP request could not be authorized.

Action: Check the given subject DN if given. Check error message regarding encryption during startup in 'start.log'.

WAR(IDSJ074): Could not modify dxmState attribute for this server: '*string1*'.

Explanation: Changing the state attribute failed.

Action: Check the exception message.

INF(IDSJ075): Message broker '*string1*' successfully started as master on port '*string2*'.

Explanation: A message broker has been started in master mode.

Action: None.

INF(IDSJ076): Message broker '*string1*' started without slave, no high availability is required.

Explanation: The master broker is started as single embedded broker, no slave is required.

Action: None.

INF(IDSJ077): Message broker '*string1*' has been started as slave to local master using repository '*string2*'.

Explanation: A message broker has been started in slave mode to be attached to a master broker.

Action: None.

INF(IDSJ078): Message broker '*string1*' being stopped.

Explanation: A message broker is stopped, regardless whether master or slave.

Action: None.

ERR(IDSJ079): Failure while trying to start the master broker '*string1*', exception with message '*string2*' occurred.

Explanation: The attempt to start the master broker failed.

Action: Examine failure reason, if need be add missing configuration items.

ERR(IDSJ080): Failure while trying to start the slave broker '*string1*', exception with message '*string2*' occurred.

Explanation: The attempt to start the slave broker failed, master broker starts without slave.

Action: Examine failure reason, if need be add missing configuration items.

WAR(IDSJ081): Failure while trying to stop the message broker '*string1*', exception with message '*string2*' occurred.

Explanation: The attempt to stop the broker failed.

Action: Check active MQ processes running after server shut down and close them.

ERR(IDSJ082): Failure while trying to connect the message broker '*string1*' with URI '*string2*', exception with message '*string3*' occurred.

Explanation: The attempt to connect a master broker to another master broker for message forwarding failed, master broker starts unconnected.

Action: Examine failure reason, if need be add missing configuration items.

INF(IDSJ083): Network connection between message broker '*string1*' and remote broker '*string2*' started.

Explanation: Two master brokers are connected for the forwarding of messages.

Action: None.

INF(IDSJ084): Message recovery failed, exception '*string1*' occurred.

Explanation: Exception occurred during attempt to retrieve messages from a remote datastore of a failed second server.

Action: Investigate failure reason, retry.

INF(IDSJ085): Network connection between (slave) broker '*string1*' and message broker '*string2*' started.

Explanation: Two master brokers are connected for the forwarding of messages.

Action: None.

INF(IDSJ086): Starting message broker '*string1*' as master on port '*string2*'.

Explanation: A message broker is being started in master mode.

Action: None.

INF(IDSJ087): Starting (slave) message broker '*string1*'.

Explanation: A message broker is started in slave mode to be attached to a master broker.

Action: None.

INF(IDSJ088): Starting network connection for message broker '*string1*'.

Explanation: Two master brokers are connected for the forwarding of messages.

Action: None.

INF(IDSJ089): Starting network connection for (slave) message broker '*string1*'.

Explanation: Two master brokers are connected for the forwarding of messages.

Action: None.

INF(IDSJ090): At least queue '*string1*' is not empty (size = *string2*). Sleeping 10 seconds

before checking queue sizes again.

Explanation: All queues need to be empty before slave broker can terminate.

Action: None.

INF(IDSJ091): Checking queue size for queue '*string1*': size = *string2*).

Explanation: All queues need to be empty before slave broker can terminate.

Action: None.

INF(IDSJ092): Forwarding of messages (by slave broker) terminated; number of recovered messages: '*string1*'; number of messages that were not forwarded = *string2*).

Explanation: Message recovery (done by slave broker) finishes and logs the messages that have been handled / not handled.

Action: None.

INF(IDSJ093): Reading workflow definitions finished (Connectivity total: *string1*; Provisioning total: *string2*).

Explanation: Server is currently initializing.

Action: None.

ERR(IDSJ101): Error while getting private keys from Resource Manager '*string1*'.

Explanation: Server (or worker respectively) failed to retrieve private keys. Encryption won't work properly.

Action: Check configuration, inform DirX Identity technical support about the problem.

ERR(IDSJ102): Error while getting certificate from Resource Manager '*string1*'

Explanation: Server (or worker respectively) failed to retrieve certificates. Encryption won't work properly.

Action: Check configuration, inform DirX Identity technical support about the problem.

WAR(IDSJ103): Ignored filter instruction *string1*.

Explanation: Configuration problem (probably install_path/ids-j-<domain>-S<number>/bin/resmgr.xml), invalid filter.

Action: Check configuration (see install_path/ids-j-<domain>-S<number>/bin/resmgr.xml), inform DirX Identity technical support about the problem.

WAR(IDSJ104): Resource *string1* is both flat and a directory.

Explanation: A resource exists in at least two resources - as a flat file and as a directory.

Action: Check configuration.

INF(IDSJ105): Loading Resource Manager *string1*.

Explanation: Server is currently loading a Resource Manager.

Action: None.

WAR(IDSJ106): *string1*Ignored duplicate resource *string2* from *string3*.

Explanation: A resource exists in at least two resources. Only first occurrence will be used.

Action: Check configuration.

WAR(IDSJ107): *string1* caught *string2* while reading entry with DN = *string3*: *string4*.

Explanation: The LDAP reader cannot find the requested LDAP entry.

Action: Check bind credentials and access rights.

WAR(IDSJ108): *string1* caught *string2* while searching with base = *string3* and filter = '*string4*': *string5*.

Explanation: The LDAP reader cannot search the requested LDAP entries.

Action: Check LDAP filter, bind credentials and access rights.

ERR(IDSJ109): Cannot load LdapReader with name = *string1* and class = *string2*: *string3*.

Explanation: Configuration Manager tried to load the configured LDAP readers.

Action: Check the elements <ldapreader> in the file resmgr.xml for correct class names and sub-elements.

WAR(IDSJ110): *string1* received Search Result Entry of type *string2* while searching with base = '*string3*' and filter = '*string4*'.

Explanation: The LDAP reader ignored the entry.

Action: Maybe size limit was exceeded. Change the search filter, if possible.

WAR(IDSJ111): JMX access to attribute *string1* of object *string2* failed: *string3*.

Explanation: Generating the server dump file content for given object failed. Object omitted.

Action: Check configuration, inform DirX Identity technical support about the problem.

ERR(IDSJ151): Task *string1* terminated unexpectedly.

Explanation: Internal error, a task (e.g. Activity) terminated unexpectedly.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ153): Severe error in WorkflowEngine (cause: *string1*) while executing workflow *string2*.

Explanation: Probably internal error, for further information see detailed message.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ154): Tried to abort activity-task *string1* due to a timeout, but encountered problems (cause: *string2*)

Explanation: Tried to terminate an expired activity, but failed to accomplish this.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ155): Encountered problems while getting applicable workflows for request '*string1*' from the Configuration Manager (cause: *string2*).

Explanation: Probably a configuration problem.

Action: Check configuration, inform DirX Identity technical support about the problem.

WAR(IDSJ156): Invalid workflow result - has no associated requests! Entry is *string1*.

Explanation: Workflow instance processed entries, but WorkflowEngine cannot associate responses with the respective requests.

Action: Inform DirX Identity technical support about the problem.

INF(IDSJ157): Response *string1* has no associated pending request!

Explanation: Response doesn't have a matching request (i.e. incoming event).

Action: None.

WAR(IDSJ158): Couldn't process invalid response *string1*.

Explanation: Internal error, responses SHOULD be of type IDMResponse (which is not the case).

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ159): Couldn't find workflow for *string1* event(s). Topics: *string2*.

Explanation: WorkflowEngine couldn't associate some events to appropriate workflows.

Action: Review workflow configuration, make sure that workflows are active (see active-flag).

WAR(IDSJ160): No workflows found for eventType=*string1*

Explanation: WorkflowEngine couldn't associate some events to appropriate workflows.

Action: Review workflow configuration, make sure that workflows are active (see active-flag).

WAR(IDSJ161): Event has no type: *string1*

Explanation: Probably internal problem (adaptors), all events SHOULD have a type.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ162): Cannot process event *string1*, should be instance of IDMRequest.

Explanation: Probably internal problem (adaptors), requests SHOULD be of type

IDMRequest

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ163): ResultListener encountered problems, going to sleep *string1*ms before retry.

Explanation: Internal error.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ164): SchedulerTask failed to process *string1*, invalid operation '*string2*'.

Explanation: Internal error, the SchedulerTask received an invalid operation.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ165): SchedulerTask failed to process *string1*, schedule doesn't exist.

Explanation: Failed to process operation (e.g. delete schedule), since schedule doesn't exist.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ166): Duplicate event *string1*.

Explanation: Internal error. Adaptor sent an event twice.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ167): Didn't start *string1*, since task has been removed from the scheduler table!

Explanation: Prohibits execution of previously deleted schedules.

Action: None.

INF(IDSJ168): Cannot find schedule '*string1*' in schedules table: ignore it.

Explanation: Probably the schedule is out-dated and not active any more.

Action: Check schedule configuration, if schedule with this UID exists and is active. If so, inform DirX Identity technical support about the problem.

INF(IDSJ169): Scheduler skipped task '*string1*' because there is already a running instance.

Explanation: Prohibits multiple executions of task which may run only once for a given time.

Action: None.

ERR(IDSJ170): Task '*string1*' failed. Transaction: *string2*, exception: *string3*.

Explanation: A task (e.g. Activity) encountered a problem. Its enclosing transaction will be aborted.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ171): Task '*string1*' failed due to a ClassCastException: local classloader: *string2*

classloader: *string3*

Explanation: A task (e.g. Activity) encountered a problem. Its enclosing transaction will be aborted.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ172): Operation *string3* to/from javaspace for Resource family *string1* for worker *string2* failed.

Explanation: The resource family could not be added to or removed from Javaspace. Some tasks might not run.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ173): Couldn't find workflow for event no: *string1* topic: *string2* event: *string3*.

Explanation: WorkflowEngine couldn't associate event to appropriate workflows.

Action: Review workflow configuration, make sure that workflows are active (see active-flag).

WAR(IDSJ174): Can't handle the response of a request (adaptor: *string1*, uid of request: *string2*).

Explanation: The adaptor has been moved and the response handling fails (statistics).

Action: If you moved the adaptor then just the internal bookkeeping can be finished.

INF(IDSJ175): Starting workflow '*string1*' with *string2* event(s).

Explanation: A suitable workflow for the given number of events (requests) will be started.

Action: None.

WAR(IDSJ176): Found *string1* applicable workflows for event (request ID *string2*). In current DXI installation this is not recommended.

Explanation: Several suitable workflows were found for a given event. In current DXI installation this is not recommended.

Action: None.

WAR(IDSJ177): ResultListener couldn't add *string1* event(s) to the Dead letter queue (timeout: *string2* sec). Exception is '*string3*'.

Explanation: ResultListener couldn't add events to the Dead letter queue.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ178): SAXParseException occurred for event with no corresponding workflow: adaptor: *string1* uid: *string2* createTime: *string3*.

Explanation: WorkflowEngine couldn't associate some events to appropriate workflows. Event contains invalid XML.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ179): ResultListener thread will be restarted in *string1* milliseconds.

Explanation: ResultListener thread terminated unexpectedly. Will be restarted.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ201): AuditListener terminated unexpectedly: *string1*. Server is terminating.

Explanation: Internal error, all future Audit information sent from workflows will be lost. Server is terminating.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ202): AuditListener encountered problems, sleeping *string1* ms before retry.

Explanation: Internal error, AuditListener tries to recover.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ203): Exception *string1* in AuditListener while publishing Audit Record to handler *string2*: *string3*. Audit Record was: *string4*.

Explanation: Error in Audit Handler, Audit message is lost.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ204): AuditListener ignored '*string1*' because it isn't a LogRecord.

Explanation: AuditListener received an invalid audit message.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ251): GlobalContext failed to stop life cycle of *string1*.

Explanation: Internal error.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ252): GlobalContext couldn't provide the value of variable *string1*.

Explanation: Server tried to replace a placeholder which isn't defined.

Action: Check configuration, Inform DirX Identity technical support about the problem.

ERR(IDSJ253): The GlobalContext failed to retrieve the directory content of *string1*.

Explanation: A worker failed to read a directory via HTTP.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ254): GlobalContext failed to provide a classloader for '*string1* ' (cause: *string2*).

Explanation: A worker failed to read a directory via HTTP.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ255): GlobalContext failed to reload the Configuration Manager (cause: *string1*).

Explanation: Internal error, maybe due to time dependencies, see detailed message for further information.

Action: Start reload of configuration again, if the issue still exists inform DirX Identity technical support about the problem.

ERR(IDSJ256): GlobalContext failed to start life cycle of *string1*.

Explanation: Internal error.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ257): Couldn't load *string1* (annotation=*string2*) using classloader *string3* (cause: *string4*).

Explanation: Server failed to load a java-class. See detailed message for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ258): GlobalContext failed to reload the properties *string2* (cause: *string3*).

Explanation: Server failed to load properties from install_path/ids-j-<domain>-S<number>/confdb/GlobalContext.properties.

Action: Check access rights of file install_path/ids-j-<domain>-S<number>/confdb/GlobalContext.properties.

INF(IDSJ259): UniverseContext has changed: *string1*.

Explanation: In most cases the result of a 'reload configuration' request.

Action: None.

INF(IDSJ260): Reporting 'serverstate' has been interrupted. Report not finished.

Explanation: In most cases the next cycle will produce a whole report.

Action: None.

ERR(IDSJ300): Encountered problems while processing 'fireEvent' SOAP-request. Message is *string1*

Explanation: Failed to process event. See detailed message for further information.

Action: Inform DirX Identity technical support about the problem.

INF(IDSJ320): AdminService: Processing 'shutdown server' request, exiting in *string1* seconds.

Explanation: SOAP-Request 'shutdown server' is being executed.

Action: None.

INF(IDSJ321): AdminService: suspend server.

Explanation: SOAP-Request 'suspend server' is being executed.

Action: None.

INF(IDSJ322): AdminService: resume server.

Explanation: SOAP-Request 'resume server' is being executed.

Action: None.

INF(IDSJ323): AdminService: reload configuration.

Explanation: SOAP-Request 'reload configuration' is being executed.

Action: None.

ERR(IDSJ350): Tried to run WorkflowContainer request *string1*, but encountered problems (cause: *string2*).

Explanation: Internal error. See detailed message for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ351): WorkflowContainerListener encountered problems, going to sleep *string1* ms before retry.

Explanation: Internal error, component 'WorkflowContainerListener' will try to recover.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ352): Couldn’t read from console, sleeping now (cause: *string1*).

Explanation: Internal error. See detailed message for further information.

Action: Inform DirX Identity technical support about the problem.

INF(IDSJ353): Trying to rebind to Java server (*string1* of *string2*). Note: Not all tries to rebind are logged.

Explanation: Remote worker is currently rebinding to the server.

Action: None.

INF(IDSJ354): '*string1*' successfully bound.

Explanation: Connection to either JavaSpace, Repository or TransactionManager successfully established.

Action: None.

ERR(IDSJ355): Rebind to Java server failed: *string1*, sleeping *string2* ms. Note: Not all tries to rebind are logged.

Explanation: Worker failed to bind to server. Server is probably down.

Action: Check idmsvc.ini in worker directory, assure that server is started.

INF(IDSJ356): Stopping WorkerContainer now.

Explanation: Worker container is going down now.

Action: None.

INF(IDSJ357): Worker *string1* successfully bound.

Explanation: Worker - Server connection (RMI) established.

Action: None.

WAR(IDSJ358): Worker *string1* failed to process task *string2* (cause *string3*)

Explanation: Internal error. Worker encountered an unexpected error.

Action: Inform DirX Identity technical support about the problem.

INF(IDSJ359): Worker *string1*: Try to rebind due to : *string2*

Explanation: Worker detected a serious problem (e.g. server went down). Trying to rebind.

Action: None.

WAR(IDSJ360): Worker *string1*: aborting infrastructure transaction *string2*.

Explanation: Due to an error, worker aborts a transaction.

Action: None.

WAR(IDSJ361): Timeout! Couldn't read WorkerTable within *string1* ms, worker *string2* won't be added to the table.

Explanation: Worker doesn't respond to a server request in appropriate time.

Action: Might happen under heavy load. If not, inform DirX Identity technical support about the problem.

ERR(IDSJ362): No parameter 'com.siemens.idm.baseurl' set in configuration file idmsvc.ini.

Explanation: The parameter is mandatory and addresses the Java server.

Action: Did you run the Configuration Wizard after the installation or after a loading of an backup? Either run the Configuration Wizard or change the parameter in the idmsvc.ini file.

ERR(IDSJ363): No hostname in parameter 'com.siemens.idm.baseurl' in configuration file idmsvc.ini.

Explanation: The parameter is mandatory and addresses the Java server.

Action: Did you run the Configuration Wizard after the installation or after a loading of an backup? Either run the Configuration Wizard or change the parameter in the idmsvc.ini file.

ERR(IDSJ364): Cannot connect to Java server: *string1*; wait *string2* ms before retry *string3* of

string4

Explanation: Failure to connect to Java server due to the given exception message. The server is retrying.

Action: Check the given exception message. Is the Java server running? Are the connection parameter correct (SSL etc.)?

ERR(IDSJ365): Error overriding standard error/output to file: *string1*

Explanation: Failure to redirect standard error/output streams.

Action: Check the given exception message.

ERR(IDSJ366): Cannot find main Java class in the jar file.

Explanation: The parameter 'mainclass' is mandatory.

Action: Did you run the Configuration Wizard after the installation or after a loading of an backup? Is the idmsvc.ini file corrupted? Either run the Configuration Wizard or change the parameter in the idmsvc.ini file.

ERR(IDSJ367): Cannot find the public method in the jar file.

Explanation: The parameter 'mainclass' is mandatory.

Action: Did you run the Configuration Wizard after the installation or after a loading of an backup? Is the idmsvc.ini file corrupted? Either run the Configuration Wizard or change the parameter in the idmsvc.ini file.

ERR(IDSJ368): Cannot create JVM.

Explanation: The parameter 'vm' is mandatory. Fatal error in loading JVM.

Action: Did you run the Configuration Wizard after the installation or after a loading of an backup? Is the idmsvc.ini file corrupted? Either run the Configuration Wizard or change the parameter in the idmsvc.ini file.

ERR(IDSJ369): Out of memory.

Explanation: Fatal error in loading JVM.

Action: Did you run the Configuration Wizard after the installation or after a loading of an backup? Is the idmsvc.ini file corrupted? Either run the Configuration Wizard or change the parameter in the idmsvc.ini file.

ERR(IDSJ370): Cannot load VM library.

Explanation: Fatal error in loading JVM.

Action: Did you run the Configuration Wizard after the installation or after a loading of an backup? Is the idmsvc.ini file corrupted? Either run the Configuration Wizard or change the parameter in the idmsvc.ini file.

ERR(IDSJ371): Unexpected error occurred while starting. See 'idmsvc-start.log' file for more details.

Explanation: Fatal error in initializing the server.

Action: Check the 'idmsvc-start.log' file.

ERR(IDSJ372): Starting service failed with error: *string1* (*string2*).

Explanation: Fatal error in loading Java server.

Action: Check the given detailed message.

ERR(IDSJ374): Failure in initializing the Java server: *string1*

Explanation: The reason is given in the exception message. See also the file 'logs/idmsvc-start.log'.

Action: Check the given exception message. Are the connection parameter correct (SSL etc.)? Is the configuration correct?

ERR(IDSJ375): Failure in shutting down the Java server: *string1*

Explanation: The reason is given in the exception message. See also the file 'logs/idmsvc-shutdown.log'.

Action: Check the given exception message. Are the connection parameter correct (SSL etc.)? Is the configuration correct?

ERR(IDSJ376): Unexpected error occurred while shutting down. See 'idmsvc-stop.log' file for more details.

Explanation: Fatal error in shutting down the server.

Action: Check the 'idmsvc-start.log' file.

WAR(IDSJ377): Writing the diagnosis support info file '*string1*' failed: *string2*.

Explanation: Could not write partly or at all the diagnosis file. See given exception for more information.

Action: Check the stderr.<n>.log file.

ERR(IDSJ378): Starting service failed when calling JNI_CreateJavaVM. Most probably JMX port is in use. Check with OS tools like netstat for used ports.

Explanation: Fatal error in creating Java VM.

Action: Check the given detailed message.

WAR(IDSJ379): Loading the library '*string1*' failed; using path: *string2*.

Explanation: Diagnosis support info file could only partly written.

Action: Check the given detailed message. Inform DirX Identity technical support about the problem.

ERR(IDSJ400): Cannot load adaptor '*string1*', missing class definition.

Explanation: Incomplete or erroneous adaptor configuration.

Action: Check adaptor configuration, inform DirX Identity technical support about the problem.

ERR(IDSJ401): Error while instantiating adaptor '*string1*' (*string2*)

Explanation: Possible reason: missing Java classes.

Action: Check adaptor configuration and Java classes, inform DirX Identity technical support about the problem.

INF(IDSJ402): Adaptor *string1* started.

Explanation: Server has started an adaptor.

Action: None.

INF(IDSJ403): Terminating adaptor *string1*.

Explanation: Server has terminated an adaptor.

Action: None.

WAR(IDSJ404): adaptor thread for *string1* is still alive!

Explanation: Soft termination of adaptor failed while server is going down.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ405): Encountered problems while terminating adaptor *string1* (cause *string2*).

Explanation: Adaptor reported an error while shutting down.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ406): Writing XML record '*string1*' failed: *string2*.

Explanation: Writing XML record to the repository failed.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ407): The adaptor '*string1*' could not be started: *string2*.

Explanation: Adaptor reported an error while being started. Operation was called via JMX.

Action: Check the exception message.

WAR(IDSJ408): The adaptor '*string1*' could not be stopped: *string2*.

Explanation: Adaptor reported an error while being stopped. Operation was called via JMX.

Action: Check the exception message.

WAR(IDSJ409): The worker for the Resource Family '*string1*' could not be started: *string2*.

Explanation: Worker thread reported an error while being started. Operation was called

via JMX.

Action: Check the exception message.

WAR(IDSJ410): The worker for the Resource Family '*string1*' could not be stopped: *string2*.

Explanation: Worker thread reported an error while being stopped. Operation was called via JMX.

Action: Check the exception message.

WAR(IDSJ411): Reload adaptors is not allowed if 'backup' flag is enabled.

Explanation: Cannot reload adaptors when 'backup' flag is enabled. Both features do not work together.

Action: None.

WAR(IDSJ412): Reading XML record '*string1*' failed: *string2*.

Explanation: Reading XML record from the repository failed.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ500): Failed to produce image for workflow *string1* (cause: *string2*).

Explanation: Typically happens on Unix platforms, if graphical subsystem is not installed.

Action: Check if graphical subsystem is installed.

ERR(IDSJ600): Exception adding workflow *string1* to cache: *string2*.

Explanation: Scheduler could not add new workflow instance to internal Cache. This may be a memory problem.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ601): Exception reading workflow *string1* from cache: *string2*.

Explanation: Scheduler could not read workflow instance to internal Cache. Either, the workflow has not yet instantiated or this is an internal problem.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ602): Exception starting workflow *string1*: *string2*.

Explanation: Scheduler could not start workflow.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ603): Exception waiting for result of workflow *string1*: *string2*.

Explanation: Scheduler receives exception when waiting for workflow termination.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ604): Cannot convert schedule time string *string1* to Date: *string2*.

Explanation: Scheduler could not transform the start or expiration time string to a Java Date object.

Action: Check the start time string. It must conform to GMT format: YYYYMMDDHHMMSSZ.

ERR(IDSJ605): LDAP Entry *string1* has no dxmDisplayName.

Explanation: All Entries beneath the connectivity schedules root MUST have a display Name attribute.

Action: Check the schedule container and entries in the connectivity database.

ERR(IDSJ606): Searching children of schedule container *string1* failed: *string2*.

Explanation: Exception when recursively searching for schedule configuration in LDAP connectivity database.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ607): Missing attribute(s) in LDAP schedule entry *string1*: dxmDisplayName = *string2*, dxmWorkflow-DN = *string3*, dxmStartTime = *string4*.

Explanation: A schedule configuration entry needs at least a start time, a workflow reference and a display name.

Action: Add the missing attributes.

ERR(IDSJ608): Exception when reading ranges of schedule *string1*: *string2*.

Explanation: The schedule range configuration seems to be wrong.

Action: Check correct schedule range configuration.

WAR(IDSJ609): Scheduler LDAP bind to *string1*:_string2_ as *string3* failed after *string4* retries: *string5*.

Explanation: Schedule Resource Manager tried to read schedule configuration from LDAP connectivity database.

Action: Check server address and bind credentials for connectivity database.

WAR(IDSJ610): Cannot read schedule with DN *string1*:_string2_.

Explanation: Schedule Resource Manager tried to read schedule with a DN constructed from an identifier based on display names.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ611): Exception reading LDAP entry with private key *string1*:_string2_.

Explanation: Resource Manager tried to read LDAP entry holding the private key(s).

Action: Check LDAP access control.

WAR(IDSJ612): Exception reading LDAP entry with certificate *string1*:_string2_.

Explanation: Resource Manager tried to read LDAP entry holding the certificate(s).

Action: Check LDAP access control.

INF(IDSJ613): Restarting scheduled workflow '*string1*' at *string2*.

Explanation: Scheduler sets a timer for restarting the workflow according schedule.

Action: None.

INF(IDSJ614): Starting workflow '*string1*' according to schedule '*string2*' ...

Explanation: Scheduler is starting the workflow according to schedule.

Action: None.

ERR(IDSJ615): Cannot start workflow '*string1*' according to schedule '*string2*': *string3*.

Explanation: Scheduler cannot start the workflow according to schedule.

Action: None.

ERR(IDSJ616): *string1* should be a folder.

Explanation: Resource Manager tries to read schedules beneath an entry, which is no folder.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ617): *string1* when reading schedule with ID='*string2*': *string3*.

Explanation: Resource Manager received an exception when reading a schedule from the LDAP entry.

Action: Check correct schedule configuration.

ERR(IDSJ618): Cannot read schedule table: *string1*.

Explanation: Workflow scheduler cannot read the central schedule task table from the Space.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ619): Cannot take schedule table: *string1*.

Explanation: Workflow scheduler cannot take (read and remove) the central schedule task table from the Space.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ620): Cannot write schedule table: *string1*.

Explanation: Workflow scheduler cannot write the central schedule task table to the Space.

Action: Inform DirX Identity technical support about the problem.

INF(IDSJ621): Add schedule '*string1*' for workflow '*string2*'.

Explanation: A schedule for a workflow has been added.

Action: None.

ERR(IDSJ622): Cannot write schedule task '*string1*' to Space: *string2*.

Explanation: Workflow scheduler cannot add a new schedule task to the central Space.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ623): Cannot write schedule task '*string1*' to Space: *string2*.

Explanation: Workflow scheduler cannot add a new schedule task to the central Space.

Action: Inform DirX Identity technical support about the problem.

INF(IDSJ624): Starting workflow schedules.

Explanation: Server is starting schedules according to the configuration.

Action: None.

ERR(IDSJ625): Cannot start all schedules: *string1*.

Explanation: An exception occurred when Scheduler tried to start all configured schedules.

Action: Inform DirX Identity technical support about the problem.

INF(IDSJ626): Removing workflow schedules.

Explanation: Server is removing previously loaded schedules.

Action: None.

ERR(IDSJ627): Cannot remove schedules: *string1*.

Explanation: An exception occurred when Scheduler tried to remove all schedules.

Action: Inform DirX Identity technical support about the problem.

INF(IDSJ628): Reloading workflow schedules.

Explanation: In most cases the result of a 'reload configuration' request.

Action: None.

ERR(IDSJ629): Cannot reload schedules: *string1*.

Explanation: An exception occurred when Scheduler tried to reload schedules (remove old and start new ones).

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJ630): Cannot initialize server: *string1*.

Explanation: An exception occurred during initialization of the server.

Action: See detailed exception message.

INF(IDSJ631): Bind to LDAP server *string1* failed (*string2* of *string3*): '*string4*'. Trying to connect...

Explanation: Bind failure is seen as temporary. Retrying is done.

Action: None.

ERR(IDSJ632): Bind to LDAP server *string1* failed after *string2* retries: '*string3*'.

Explanation: Cannot bind to LDAP server. Server is shutting down.

Action: Check your LDAP server connection.

INF(IDSJ633): Bind to LDAP server *string1* successful.

Explanation: Successfully bound to the named LDAP server.

Action: None.

INF(IDSJ634): Starting IdS-J server.

Explanation: The IdS-J server is starting (phase 2 after reading the configuration from LDAP).

Action: None.

INF(IDSJ635): Redirected standard output and error to files.

Explanation: Server is redirecting stdout and stderr to files for better traceability.

Action: None.

WAR(IDSJ636): Redirecting standard output and error to files failed *string1*

Explanation: See the error message.

Action: Depends on the error message.

INF(IDSJ637): Read workflow schedule '*string1*' (ID: '*string2*').

Explanation: Scheduler has read the schedule from connectivity database and stored it in its schedule table.

Action: None.

INF(IDSJ638): Bind of Schedule Resource Manager to LDAP server *string1*:_string2_ failed: *string3*.

Explanation: The resource manager cannot bind to connectivity database and read the workflow schedules.

Action: Check your LDAP server connection.

INF(IDSJ639): Schedule Resource Manager cannot read bind credentials: *string1*.

Explanation: The resource manager cannot find the configuration data for the LDAP bind.

Action: Check the configuration data.

INF(IDSJ640): Searching resource with display name = '*string1*' underneath folder with DN = '*string2*' failed: *string3*.

Explanation: The resource manager searches a schedule or a schedule folder with the given name, but caught an exception or found none or more than one result entries.

Action: Check the configuration data for the scheduler and the connectivity database underneath the schedules folder.

INF(IDSJ641): Finished Workflow '*string1*' for target system '*string2*' started by schedule '*string3*'.

Explanation: Scheduler was informed that the previously started workflow has terminated. It tries to start the same workflow for the next waiting target systems.

Action: None.

INF(IDSJ642): Starting workflow '*string1*' for target system '*string2*' according to schedule '*string3*' ...

Explanation: Scheduler is regularly starting the workflow for the given target system according schedule.

Action: None.

INF(IDSJ643): Cannot start workflow with DN = '*string1*': number of currently running instances reached the limit *string2*.

Explanation: the configured limit of parallel running scheduled instances of the same workflow is reached. Scheduler did not start the next one although schedule start time is reached.

Action: Maybe you want to raise the workflow limit or the time interval.

INF(IDSJ644): Discard schedule '*string3*' (DN= '*string1*') because of inactive or not applicable for the active adaptors of this server instance or not existing workflow with DN = '*string2*'.

Explanation: The schedule references a workflow, which is either not existing or not active or not applicable for this server (depending on the active adaptors). Schedule is ignored.

Action: Correct the workflow DN in the schedule or set the workflow active.

ERR(IDSJ645): Cannot initialize server: *string1*.

Explanation: An exception occurred during initialization (2nd phase) of the server.

Action: See detailed exception message.

INF(IDSJ646): Starting *string1* server --- version *string2*

Explanation: Displays the current version.

Action: None.

INF(IDSJ647): Key/Truststore '*string1*' contains *string2* entries.

Explanation: Displays the aliases in the named key/truststore

Action: None.

INF(IDSJ648): Aliasname '*string1*'.

Explanation: Displays the alias name in the named key/truststore

Action: None.

INF(IDSJ649): ConstraintViolation for activity *string1* in workflow with ID '*string2*'. Set state to *string3*, applicationState to *string4*. Violation details: *string5*).

Explanation: The participants constraints implementation for a people activity threw a violation exception to indicate that the number of participants is not sufficient.

Action: None.

INF(IDSJ650): State of activity '*string1*' of workflow with ID '*string2*' (*string3*) changes from '*string4*' to '*string5*'.

Explanation: The state of the activity changed.

Action: None.

INF(IDSJ651): Workflow with ID '*string1*' (*string2*) terminated with '*string3*': *string4*.

Explanation: Workflow engine terminated with error.

Action: Check the error message.

INF(IDSJ652): Activity '*string1*' is in retry mode and is scheduled as RUNNABLE in *string2* seconds. Current retry count: *string3*, retry limit: *string4*.

Explanation: Activity state is set to RUNNABLE.

Action: None.

INF(IDSJ653): State of workflow with ID '*string1*' (*string2*) changes from '*string3*' to '*string4*'.

Explanation: The state of the workflow changed.

Action: None.

ERR(IDSJ654): Activity '*string1*' (*string2*) of workflow with ID '*string3*' terminated with '*string4*': *string5*.

Explanation: Activity task terminated with error.

Action: Check the error message.

ERR(IDSJ655): An activity task of workflow with ID '*string1*' terminated with '*string2*': *string3*.

Explanation: Activity task terminated with error.

Action: Check the error message.

INF(IDSJ656): Exception handling of activity task '*string1*' of workflow with ID '*string2*' terminated with '*string3*': *string4*.

Explanation: Exception handling of activity task terminated with error.

Action: Check the error message.

INF(IDSJ657): State of activity '*string1*' of workflow with ID '*string2*' (*string3*) changes from '*string4*' to '*string5*'. Workflow put to Java space.

Explanation: The state of the workflow and/or its activities changed.

Action: None.

INF(IDSJ658): State of activity '*string1*' of workflow with ID '*string2*' changes to '*string3*'. Workflow put to Java space.

Explanation: The state of the workflow and/or its activities changed.

Action: None.

INF(IDSJ659): '*string1*' did not get an LDAP reader, scheduler cannot run any workflow.

Explanation: Scheduler could not get an LDAP reader used for reading schedules.

Action: None.

INF(IDSJ660): Workflow with uid = *string1* has finished. Cleanup started.

Explanation: Workflow has terminated (either successfully or with errors).

Action: None.

WAR(IDSJ661): Cannot retrieve parent workflow ID from parent uid=*string1* for child workflow uid=*string2*, displayName=*string1* from repository.

Explanation: The repository does not contain the parent workflow whose uid is stored in a child workflow. No child info is passed to the parent workflow.

Action: Check if parent has expired or is deleted.

ERR(IDSJ662): Exception *string1* occurred while retrieving parent workflow ID from parent uid=*string2* for child workflow uid=*string3*, displayName=*string4* from repository: *string5*.

Explanation: An exception occurred during read of the parent workflow ID from the repository. No child info is passed to the parent workflow.

Action: Check the details of the message.

ERR(IDSJ663): Exception *string1* occurred while reading parent workflow ID=*string2* for child workflow uid=*string3*, displayName=*string4* from repository: *string5*.

Explanation: An exception occurred during read of the parent workflow from the repository. No child info is passed to the parent workflow.

Action: Check the details of the message.

ERR(IDSJ664): Exception *string1* occurred in child=*string2* while processing workflow info for parent id=*string3*, startedRequestWorkflows=*string4*: *string5*.

Explanation: An exception occurred while constructing the workflow info from the startedRequestWorkflows context attribute. No child info is passed to the parent workflow.

Action: Check the details of the message.

ERR(IDSJ665): Exception *string1* occurred in child=*string2* while processing workflow change tracking info from parent id=*string3*, accountGroupChanges=*string4*: *string5*.

Explanation: An exception occurred while constructing the workflow change tracking info from the parent's accountGroupChanges context attribute. No child info is passed to the parent workflow.

Action: Check the details of the message.

ERR(IDSJ666): Exception *string1* occurred in child=*string2* while processing own workflow change tracking info for parent id=*string3*, accountGroupChanges=*string4*: *string5*.

Explanation: An exception occurred while constructing the workflow change tracking info from the child's accountGroupChanges context attribute. No child info is passed to the parent workflow.

Action: Check the details of the message.

ERR(IDSJ667): Exception *string1* occurred in child=*string2* during context update of parent id=*string3*: *string4*.

Explanation: An exception occurred while updating the parent context. No child info is passed to the parent workflow.

Action: Check the details of the message.

ERR(IDSJ668): Exception *string1* occurred in child=*string2* while writing checker task for parent id=*string3* to JavaSpace: *string4*.

Explanation: An exception occurred while creating the checker task for finished child workflows. The parent workflow may stick at the SyncChildWorkflows activity.

Action: Check the details of the message.

INF(IDSJ669): Notifying parent workflow *string1* from child *string2*: state=*string3*, applicationState=*string4*, accountGroupChanges=*string5*.

Explanation: The parent workflow is notified by the child. This message includes the account/group changes and is reported only with loglevel FINEST.

Action: None.

INF(IDSJ670): Notifying parent workflow *string1* from child *string2*: state=*string3*, applicationState=*string4*.

Explanation: The parent workflow is notified by the child. This message is reported with loglevel FINE.

Action: None.

INF(IDSJ671): Checking state of the synchronization activity failed for workflow *string1*: Name of the synchronization activity is not present in the workflow context.

Explanation: The name may be written to context later.

Action: None.

ERR(IDSJ672): Checking state of the synchronization activity failed for workflow *string1*: Synchronization activity '*string2*' instance not found.

Explanation: The synchronization activity whose name is stored in the workflow context does not exist for the workflow instance.

Action: Check workflow structure and maybe other error messages (if exist).

INF(IDSJ673): Checking state of the synchronization activity for workflow *string1*: Found activity '*string2*'.

Explanation: The synchronization activity whose name is stored in the workflow context is identified.

Action: None.

INF(IDSJ674): Synchronization activity '*string1*' for workflow *string2* has finished with state=*string3*, applicationState=*string4*.

Explanation: The synchronization activity has finished with the reported state / applicationState.

Action: None.

ERR(IDSJ675): Exception *string1* occurred while evaluating synchronization activity *string2* of workflow *string3*: *string4*.

Explanation: An unexpected error occurred during evaluation of the synchronization activity.

Action: Check workflow structure. Check for class loading problems, especially with customer provided jar files. Check the detailed message.

INF(IDSJ676): Running Workflow *string1*.

Explanation: The Workflow was started.

Action: None.

INF(IDSJ677): Re-processing Workflow *string1*.

Explanation: The Workflow was processed.

Action: None.

WAR(IDSJ678): Reading workflow with ID '*string1*' failed: *string2*.

Explanation: Workflow engine could not read workflow.

Action: Check the error message.

ERR(IDSJ679): Ignore schedule with UID='*string1*' and display name = '*string2*': time interval too low: *string3* sec.

Explanation: The time interval in a schedule should be at least 1 minute.

Action: Set the time interval > 1 minute.

ERR(IDSJ680): Time Interval of schedule '*string1*' lower than deviation. Set it to deviation + 1. Configured value: '*string2*'.

Explanation: The time interval of a schedule should be at least the deviation.

Action: Set the time interval > deviation.

INF(IDSJ681): Workflow with ID '*string1*' (*string2*) terminated with '*string3*'.

Explanation: Workflow engine terminated due to interruption (workflow timeout or server shutdown).

Action: None.

INF(IDSJ682): Reading schedules beneath folder '*string1*'.

Explanation: Reading schedules beneath given folder DN.

Action: None.

INF(IDSJ683): Reloading adaptors, resource family and schedule configuration started (Timeout: *string1*).

Explanation: Reloading adaptors, resource families and schedules.

Action: None.

INF(IDSJ684): Reloading adaptors is currently running (start time: *string1*).

Explanation: Reloading adaptors operation is currently running.

Action: None.

INF(IDSJ685): Reloading adaptors, resource family and schedule configuration finished (start time: *string1*).

Explanation: Reloading adaptors, resource families and schedules is finished.

Action: None.

INF(IDSJ686): Pending request for reloading adaptors, resource family and schedule configuration started.

Explanation: Reloading adaptors, resource families and schedules is started again.

Action: None.

INF(IDSJ687): Scheduler is not running in this IDS-J server.

Explanation: The scheduler task is not assigned to this server.

Action: None.

WAR(IDSJ688): Cannot start workflow '*string1*' according schedule '*string2*': No target systems found; the schedule is ignored.

Explanation: Using the target system cluster definition no target system could be found.

Action: Check the search base and filter in your target system selection.

ERR(IDSJ689): Could not evaluate a queue name from the workflow's 'resolutionVariables' (namely 'topicSet') for schedule *string1*.

Explanation: The workflow could not be started by schedule because the queue name could not be evaluated successfully.

Action: Check the 'resolutionVariables' of the workflow. The queue name for starting a workflow is unknown.

WAR(IDSJ690): 'reload Configuration' or 'synchronize Schedulers' were called in the startup phase of the server where no JavaSpace is yet available. Repeat operation later again.

Explanation: 'reload Configuration' or 'synchronize Schedulers' were called in the startup phase of the server where no JavaSpace is yet available. Repeat operation later again.

Action: Call 'reloadConfiguration' or later again.

INF(IDSJ691): Scheduler task created for workflow '*string1*' (as workflow is currently locked); repetition count: *string2*; scheduled for *string3*.

Explanation: The workflow is currently locked by some other workflow task; therefore the execution of current workflow task is repeated later on (started as a scheduler task).

Action: None.

WAR(IDSJ692): No scheduler task created for workflow '*string1*' (as workflow is currently locked but maximum number of repetitions has been reached); repetition count: *string2*.

Explanation: The workflow is currently locked by some other workflow task; therefore the execution of current workflow task needs to be repeated later on (started as a

scheduler task) but the maximmum number of repetitions has been hit.

Action: Check why so many activities are running in parallel.

WAR(IDSJ693): Ignoring workflow '*string1*' since workflow was not read successfully.

Explanation: The workflow task couldn't be processed as the workflow couldn't be read successfully from LDAP.

Action: Check the error while reading the workflow.

ERR(IDSJ694): '*string1*' while reading schedules beneath folder '*string2*'; error: *string3*.

Explanation: Reading schedules beneath given folder DN failed with an exception.

Action: Check the error.

INF(IDSJ695): '*string1*' activity does not exist for workflow '*string2*'. Property list will simply not be applied.

Explanation: For combined Java workflows, there is no initial 'join' activity and therefore property list will be ignored.

Action: None.

ERR(IDSJ696): Encountered problems while running activity '*string1*' (cause *string2*).

Explanation: Activity failed. For further information, see detailed message.

Action: Check workflow definition.

INF(IDSJ697): Aborting workflow via Web Admin action '*string1*'.

Explanation: Worker container received an abort request. Workflow is being aborted.

Action: None.

WAR(IDSJ698): Workflow with ID '*string1*' (*string2*) terminated with '*string3*': *string4*.

Explanation: Workflow engine terminated with error.

Action: Check the error message.

WAR(IDSJ699): The last time out scheduler task differs from the expected one; current task: '*string1*'; expected task: '*string2*'; run time instance: cn=*string3*; worflow dn:_string4_.

Explanation: The last timeout task is not the one that was expected.

Action: Check the error message.

INF(IDSJ700): Importing file *string1*. Different versions found, trying to convert from '*string2*' to '*string3*'

Explanation: Server found previously exported repository file during startup.

Action: None.

INF(IDSJ701): *string1* entries successfully imported from file *string2* for repository *string3*.

Explanation: Import was successful.

Action: None.

INF(IDSJ702): *string1* repositories imported.

Explanation: During startup the given number of repositories have been imported.

Action: If all exported repositories have now been imported you can change the setting in GlobalContext.properties file.

ERR(IDSJ800): Load of server configuration failed '*string1*'.

Explanation: During startup the load of the configuration from Connectivity store failed. See exception message for more information.

Action: Check the message.

INF(IDSJ801): Recovering is already running since *string1*. Request is ignored.

Explanation: Recovering is already started.

Action: None.

WAR(IDSJ802): Recovering messages from Active MQ and/or BackupSlaveListener failed: '*string1*'.

Explanation: The recovering from the shared database of another Java server or the recovering from the backup repository failed.

Action: Check the exception message.

INF(IDSJ803): Message recovery failed, exception '*string1*' occurred.

Explanation: Exception occurred during attempt to retrieve messages from a remote datastore of a failed second server.

Action: Investigate failure reason, retry.

INF(IDSJ804): Recovering messages from Active MQ and/or BackupSlaveListener started.

Explanation: Recovering is starting.

Action: None.

INF(IDSJ805): Recovering messages from Active MQ and/or BackupSlaveListener finished.

Explanation: Recovering finished successfully.

Action: None.

INF(IDSJ806): Initializing embedded Tomcat.

Explanation: Server is currently initializing the embedded Tomcat.

Action: None.

INF(IDSJ807): Stopping embedded Tomcat.

Explanation: Server is currently terminating.

Action: None.

WAR(IDSJ809): The last time out scheduler is not set; current task: '*string1*'; run time instance: cn=*string3*; workflow dn:_string4_.

Explanation: The last timeout task is not available in this workflow scheduler task.

Action: Check the error message.

INF(IDSJ810): Server is not configure to run workflow'*string1*'. Domain '*string2*' does not match.

Explanation: This server is not configured to run the given workflow.

Action: Check the error message.

INF(IDSJ811): Canceling scheduled workflow '*string1*'.

Explanation: Time task for scheduler workflow has been canceled as schedules need to be reloaded.

Action: None.

ERR(IDSJ812): Cannot remove schedule task '*string1*' from JavaSpace: *string2*.

Explanation: Workflow scheduler cannot delete a schedule task from the central Space.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJ813): Scheduler ID '*string1*' (Schedule name: *string2*) exists already and will no longer be active.

Explanation: There is another schedule with the same scheduler ID. But the schedule ID must be unique. Only the last definition for that scheduler ID will be active.

Action: Check the schedules and make sure that the schedule ID (attribute cn in LDAP) is unique.

INF(IDSJ814): *string1*: Bind to LDAP server *string2* failed (*string3* of *string4*): '*string5*'. Trying to connect...

Explanation: Bind failure is seen as temporary. Retrying is done.

Action: None.

ERR(IDSJ815): *string1*: Bind to LDAP server *string2* failed after *string3* retries: *string4*.

Explanation: Cannot bind to LDAP server.

Action: Check your LDAP server connection.

WAR(IDSJ816): Parameter '*string1*' with value '*string2*' for Tomcat connector configuration not known. Parameter is ignored.

Explanation: The given parameter cannot beset for Tomcat connector configuration.

Action: Inform DirX Identity technical support about the problem.

INF(IDSJ817): Run of 'error' activity postponed as 'join' activity is in retry mode and currently set to state '*string1*'.

Explanation: Error activity will start at the end of 'join' activity.

Action: None.

INF(IDSJ900): Out dated workflow changes for workflow '*string1*' are ignored. These ones must have already been processed by another workflow task.

Explanation: Due to heavy load FullCheck eventually sends the same task more than once. Only the first task is processed, the others are ignored.

Action: None.

INF(IDSJ901): Ignoring workflow changes for workflow '*string1*' since workflow is already finished.

Explanation: The workflow task is ignored as the workflow is already finished. This may happen if there is a heavy load in the server and the workflow task arrived too late..

Action: None.

INF(IDSJ902): Ignoring activity changes of activity '*string1*' since workflow is already finished.

Explanation: The workflow task is ignored as the workflow is already finished. This may happen if there is a heavy load in the server and the activity task arrived too late..

Action: None.

INF(IDSJ903): Ignoring activity task for activity '*string1*' since activity is already finished.

Explanation: The activity task is ignored as the activity is already finished. This may happen if there is a heavy load in the server and the activity task arrived too late..

Action: None.

INF(IDSJ904): Running Activity *string1*.

Explanation: The activity was started.

Action: None.

WAR(IDSJ905): Cannot start workflow '*string1*' according to schedule '*string2*' - probably outside of ranges.

Explanation: Scheduler cannot start the workflow according to schedule. Probably outside of ranges for 1 year period.

Action: None.

16.5.11. Logging Messages (IDSJADP)

This topic describes the logging messages returned by the DirX Identity Server Adaptors.

WAR(IDSJADP000): Couldn't initialize JMX support for adaptor '*string1*'.

Explanation: Server will not be manageable via JMX due to initialization error.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJADP001): Adaptor '*string1*' couldn't process message '*string2*'.

Explanation: The adaptor could not process the given message.

Action: Read the text of the given error message and exception and check for the reason.

ERR(IDSJADP002): Adaptor '*string1*' failed to unbind.

Explanation: The adaptor could not unbind the connection.

Action: Read the text of the given error message and exception and check for the reason.

ERR(IDSJADP003): Adaptor '*string1*' out of order, since the channel '*string2*' has been closed.

Explanation: The adaptor can not process any more because the channel is closed.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJADP004): Adaptor '*string1*' stopped unexpectedly.

Explanation: Server encountered severe difficulties.

Action: Inform DirX Identity technical support about the problem.

INF(IDSJADP005): Adaptor '*string1*' attempted to delete message (identified by *string2*) from the repository but failed.

Explanation: Server could not delete the event from repository.

Action: None.

INF(IDSJADP006): Adaptor '*string1*' encountered some problems (*string2*) - is in suspended mode now.

Explanation: Adaptor is waiting that the administrator is fixing the problem.

Action: None.

ERR(IDSJADP007): The JMS messaging system of adaptor '*string1*' reported the exception *string2* (root cause is *string3*).

Explanation: The adaptor is retrying a rebind. Most commonly the cause is on the messaging server side or the network.

Action: Check for the reason, in particular the messaging server or the network. If this error situation persists inform DirX Identity technical support about the problem.

INF(IDSJADP008): Adaptor '*string1*' attempted to delete message (identified by *string2*) from the repository but failed. Stacktrace: *string3*.

Explanation: Server could not delete the event from repository.

Action: None.

ERR(IDSJADP009): The message dispatcher of adaptor '*string1*' detected a serious problem, failed to process message '*string1*'.

Explanation: Adaptor is waiting that the administrator is fixing the problem.

Action: Check for the reason. If this error situation persists inform DirX Identity technical support about the problem.

ERR(IDSJADP010): Adaptor '*string1*' failed to execute the Java script '*string2*'.

Explanation: The Java script failed to execute.

Action: Check the given exception and correct the Java script.

ERR(IDSJADP011): Adaptor '*string1*': Directory *string2* doesn't exist.

Explanation: The configuration for the LDAP adaptor is wrong. Adaptor is aborting.

Action: Check the configuration.

ERR(IDSJADP012): Adaptor '*string1*': Missing mandatory parameter '*string2*'.

Explanation: The configuration for the LDAP adaptor is wrong. Adaptor is aborting.

Action: Check the configuration.

INF(IDSJADP013): Adaptor '*string1*': *string2* entries recovered (total: *string3*).

Explanation: During recovery the adaptor has restored the given number of entries.

Action: None.

ERR(IDSJADP016): Internal error (cause: *string1*).

Explanation: During termination of adaptor or waiting for user action the given error occurred.

Action: Depending on the given exception this might be tolerable or you should inform DirX Identity technical support about the problem.

ERR(IDSJADP017): Recovering and re-opening of the repository *string1* failed: *string2*. Shutting down the server.

Explanation: Closing, backing up (saving) and open a fresh repository failed. Events and

messages can not be stored permanently any more.

Action: Check the file system. Depending on the given error messages inform DirX Identity technical support.

ERR(IDSJADP018): Received a not supported message with messageID '*string1*' topic '*string2*'.

Explanation: A not supported message was received with an empty body and without header property 'DXMUserData'.

Action: Check the client which produces such a message.

ERR(IDSJADP019): Received a not supported message with messageID '*string1*' topic '*string2*' JMStype '*string3*'.

Explanation: A not supported message was received which is not a JMSBytesMessage nor does it have a header property 'DXMUserData'.

Action: Check the client which produces such a message.

ERR(IDSJADP020): Exception while loading operation from XML (for Request workflows): XML data: '*string1*'; error: '*string2*'.

Explanation: There are XML problems while loading the given data string.

Action: Check the error message.

INF(IDSJADP021): Adaptor '*string1*' terminates (nothing to do).

Explanation: Obsolete adaptor terminates because no events are found in the repository.

Action: none.

INF(IDSJADP022): Ignoring received message message with messageID '*string1*' topic '*string2*' JMStype '*string3*'.

Explanation: Message of given type is ignored by the Mail adaptor.

Action: none.

ERR(IDSJADP023): Exception in *string1* when initializing: '*string2*'.

Explanation: There are initializing problems in the adaptor.

Action: Depending on the given exception this might be tolerable or you should inform DirX Identity technical support about the problem.

ERR(IDSJADP024): Exception when notifying Workflow Engine of changes of workflow with uid '*string1*' and path '*string2*': *string3*.

Explanation: There are initializing problems in the adaptor.

Action: Depending on the given exception this might be tolerable or you should inform DirX Identity technical support about the problem.

ERR(IDSJADP025): Exception when notifying Workflow Engine of changes of activity with name '*string1*' of Workflow with uid '*string2*' and path '*string3*': *string4*.

Explanation: There are initializing problems in the adaptor.

Action: Depending on the given exception this might be tolerable or you should inform DirX Identity technical support about the problem.

WAR(IDSJADP026): Adaptor '*string1*' couldn't process message. Trying rebind. Received JMSException '*string2*'.

Explanation: There are messaging related problems in the adaptor.

Action: Depending on the given exception this might be tolerable or you should inform DirX Identity technical support about the problem.

WAR(IDSJADP027): For the resource family '*string1*' of the activity '*string2*' (workflow '*string3*') no worker thread is configured. The task is written to Java Space anyway.

Explanation: The resource family indirectly determines on which servers or worker containers an activity may be run. Make sure that for each activity there is at least one system associated with the same resource family.

Action: In the server's Web Admin interface you can add worker threads for a resource family. Or you can add resource families in the Identity Manager. The later requires a restart of the IdS-J server.

INF(IDSJADP028): Adaptor '*string1*' suspended.

Explanation: The named adaptor has been suspended (via Web Admin / JMX call).

Action: none.

INF(IDSJADP029): Suspended adaptor '*string1*' resumed.

Explanation: The named adaptor has been resumed (via Web Admin / JMX call).

Action: none.

WAR(IDSJADP030): DeadLetterQueue operational error '*string1*'.

Explanation: Some operations on the DeadLetterQueue couldn't be done.

Action: Depending on the given exception this might be tolerable or you should inform DirX Identity technical support about the problem.

WAR(IDSJADP031): Workflow '*string1*' not started as it is not the right domain '*string2*' or other criteria are not fulfilled.

Explanation: Workflow can not be started as the criteria regarding domain - server runs EntryChangeListener etc. are fulfilled.

Action: Check with message why this workflow is not started.

WAR(IDSJADP032): Adaptor *string1*: Incoming event contained invalid XML characters. *string2* characters were removed. Incoming event: '*string3*'.

Explanation: Certain invalid XML characters were removed from the incoming event.

Action: Check the origin of this message and why it contained invalid data for events in XML format.

WAR(IDSJADP050): Adaptor '*string1*': Failed to broadcast certificate (cause: *string2*).

Explanation: Broadcasting the changed certificate to other global contexts failed.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJADP051): Adaptor '*string1*': Failed to broadcast 'reload CryptSupport' request to worker containers (cause: *string2*).

Explanation: The given request could not be broadcast to worker containers.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJADP052): Adaptor '*string1*': Cannot publish certificate to topic *string2*, JMS messaging system isn't initialized properly.

Explanation: The JMS session is not set up.

Action: Check previous error messages and change configuration.

INF(IDSJADP053): Adaptor '*string1*': published certificate to *string2*, JMSType=*string3*: cert=*string4*.

Explanation: Adaptor successfully published the certificate.

Action: None.

INF(IDSJADP054): processing configuration request '*string1*'.

Explanation: Adaptor starts to process the configuration request.

Action: None.

INF(IDSJADP055): Adaptor '*string1*': ignored event with JMSType='*string2*'.

Explanation: Adaptor ignored the event with the given JMSType.

Action: If this happens constantly inform DirX Identity technical support about the problem.

WAR(IDSJADP056): Adaptor '*string1*': ignored request certificate event. No certificate available.

Explanation: The user_admin has no certificate. Request is ignored.

Action: The user server_admin needs a PSE.

WAR(IDSJADP057): Adaptor '*string1*': Certificate for user cn=server_admin,dxmC=DirXmetahub has changed (sender: '*string2*'). Provide new PIN and restart server.

Explanation: The server has received a message that the certificate of given user has

changed (from another Java server).

Action: Change PIN for server instance and restart it.

INF(IDSJADP058): Adaptor '*string1*': published list of message services to *string2*, JMSType=*string3*, list size=*string4*.

Explanation: Adaptor successfully published the list of actual configured message services.

Action: None.

INF(IDSJADP059): Broadcast for PasswordListener for message service list will be done every *string1* minutes.

Explanation: The list of all configured message services in the Connectivity will be send to all listening PasswordListener.

Action: None.

WAR(IDSJADP060): Cannot parse interval time '*string2*' for message services list broadcast: '*string1*'.

Explanation: Interval has to be given in the format hh:mm.

Action: Change the interval.

WAR(IDSJADP061): Adaptor '*string1*': Failed to broadcast message services list (cause: *string2*).

Explanation: Broadcasting the message services list to PasswordListeners failed.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJADP062): Adaptor '*string1*': Error reading current message services list from Connectivity: '*string2*'.

Explanation: Cannot read current message services list from the Connectivity.

Action: If this is a momentary issue with the LDAP connection then do nothing otherwise read the exception details and inform DirX Identity technical support about the problem.

WAR(IDSJADP100): Adaptor '*string1*': Failed to process message '*string2*' while recovering.

Explanation: Adaptor could not recover properly.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJADP101): Adaptor '*string1*': Ignored *string2*, must be IDMRequest or IDMEvent.

Explanation: Adaptor only processes IDMRequests or IDMEvents.

Action: Check the configuration of the sender.

WAR(IDSJADP150): Adaptor '*string1*': Ignored *string2*, must be a IDMRequest.

Explanation: Adaptor only processes IDMRequests.

Action: Check the configuration of the sender.

INF(IDSJADP151): Adaptor '*string1*': processing requests, topic='*string2*'.

Explanation: Adaptor starts to process the requests in the dead letter queue.

Action: None.

WAR(IDSJADP152): Adaptor '*string1*': failed to add request to BatchQueue (cause: *string2*).

Explanation: Adaptor could not add the requests.

Action: Check the given exception or inform DirX Identity technical support about the problem.

INF(IDSJADP153): Adaptor '*string1*': *string2* entries processed.

Explanation: Adaptor finished processing.

Action: None.

INF(IDSJADP154): Adaptor '*string1*': clearing repository, topic='*string2*'.

Explanation: Adaptor is removing all requests that match the topic from the dead letter queue.

Action: None.

INF(IDSJADP155): Adaptor '*string1*': *string2* entries deleted.

Explanation: Adaptor removed the given number of entries.

Action: None.

INF(IDSJADP156): Adaptor '*string1*': searching elements, query=*string2*

Explanation: Adaptor searches for entries in the dead letter queue.

Action: None.

INF(IDSJADP157): Adaptor '*string1*': *string2* entries found.

Explanation: Adaptor found the given number of entries in the dead letter queue.

Action: None.

INF(IDSJADP200): Adaptor '*string1*': binding to message service.

Explanation: Adaptor start connecting to the message service.

Action: None.

INF(IDSJADP201): Adaptor '*string1*': successfully bound to message service on '*string2*' with subscription ID '*string3*'.

Explanation: Adaptor successfully connected to the message service.

Action: None.

WAR(IDSJADP202): Adaptor '*string1*': Retry to connect to messaging service in *string2* seconds (cause: *string3*).

Explanation: Adaptor is retrying to connect.

Action: None.

INF(IDSJADP203): Adaptor '*string1*': Requesting a full rebind due to a JMS error.

Explanation: Adaptor will try to reconnect.

Action: None.

WAR(IDSJADP204): Adaptor '*string1*': Failed to unbind from message service (cause: *string2*).

Explanation: Adaptor failed to disconnect from the message service.

Action: You can ignore this error, but if it happens often then inform DirX Identity technical support about the problem.

ERR(IDSJADP250): Adaptor '*string1*' couldn’t setup recorder: *string2* (cause: *string3*).

Explanation: Adaptor could not set up recorder.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJADP251): Adaptor '*string1*' failed to process message '*string2*' while recovering (cause: *string3*).

Explanation: Adaptor failed to process the message.

Action: Backup the named repository folder. Inform DirX Identity technical support about the problem.

ERR(IDSJADP300): Adaptor '*string1*' failed to play previously recorded messages (cause *string2*).

Explanation: Adaptor failed to process the message.

Action: Check the reason.

INF(IDSJADP301): Adaptor '*string1*' is shutting down the server.

Explanation: Adaptor starts to shut down the server.

Action: None.

INF(IDSJADP302): Adaptor '*string1*': *string2* script(s) executed, *string3* error(s). Terminating server in *string4*s, exitcode=*string5*.

Explanation: Statistics about running playback scripts before shutting down the server.

Action: None.

ERR(IDSJADP350): Adaptor '*string1*' failed to process messages in directory *string2* (cause: *string3*).

Explanation: Adaptor could not process LDIF changes.

Action: Inform DirX Identity technical support about the problem.

INF(IDSJADP351): Adaptor '*string1*': processing directory *string2*.

Explanation: Adaptor starts to process LDIF changes.

Action: None.

ERR(IDSJADP400): Adaptor '*string1*' received a not supported message with operation code '*string2*'.

Explanation: Adaptor can not process the message.

Action: Check the client which produces such a message.

ERR(IDSJADP401): Adaptor '*string1*' received a not supported message with operation code '*string2*'.

Explanation: A not supported message was received which is not a JMSBytesMessage nor does it have a header property 'DXMUserData'.

Action: Check the given exception and client which produces such a message.

WAR(IDSJADP402): Can not initialize DB repository. FS repository will be used. *string1*.

Explanation: Can not initialize DB repository. Switching to FS repository.

Action: Check presence of JDBC driver for HSQLDB.

WAR(IDSJADP403): Adaptor '*string1*': A request to reload schedules does not affect this server as the configured domain '*string2*' does not match the domain in the request: '*string3*'.

Explanation: The scheduler only reloads schedules for its domain.

Action: Use correct domain-specific subfolder for your schedules.

16.5.12. Logging Messages (IDSJAPI)

This topic describes the logging messages returned by the DirX Identity Server API.

WAR(IDSJAPI000): Failed to convert number to state. Format error while parsing '*string1*' (cause *string2*)

Explanation: Cannot convert number to state. Invalid number.

Action: Change the setting of states in your activity.

WAR(IDSJAPI001): Workflow status writer didn't get workflow logging info from thread context. No monitor entries are created.

Explanation: Internal error while providing information using the thread context.

Action: Inform your DirX Identity support.

WAR(IDSJAPI002): Workflow status writer couldn’t establish an LDAP connection. No monitor entries are created.

Explanation: Internal error while creating an LDAP connection.

Action: Inform your DirX Identity support.

WAR(IDSJAPI003): Workflow status writer couldn’t read the attribute 'dxrOptions' at the domain. No monitor entries are created. Error: *string1*.

Explanation: Internal error while creating an LDAP connection.

Action: Inform your DirX Identity support.

WAR(IDSJAPI004): Workflow status writer couldn’t reconnect to LDAP (for adding/updating the entry '*string1*'). Trying to reconnect. Error: *string2*.

Explanation: Error while creating an LDAP connection. In case of temporary LDAP exceptions the reconnect is tried again.

Action: Check the LDAP error message.

INF(IDSJAPI005): Workflow status writer created monitor entry: '*string1*'.

Explanation: Monitor entry has been created.

Action: None.

INF(IDSJAPI006): Workflow status writer: Creation of monitor entry: '*string1*' failed: Error: *string2*.

Explanation: Creation of monitor entry failed.

Action: Check the LDAP error message.

INF(IDSJAPI007): Workflow status writer modified monitor entry: '*string1*'.

Explanation: Monitor entry has been created.

Action: None.

INF(IDSJAPI008): Workflow status writer: Modification of monitor entry: '*string1*' failed: Error: *string2*.

Explanation: Modification of monitor entry failed.

Action: Check the LDAP error message.

INF(IDSJAPI009): Workflow status writer read monitor entry: '*string1*'.

Explanation: Monitor entry has been created.

Action: None.

INF(IDSJAPI010): Workflow status writer: Retrieval of monitor entry: '*string1*' failed: Error: *string2*.

Explanation: Retrieval of monitor entry failed.

Action: Check the LDAP error message.

INF(IDSJAPI011): Workflow status writer: Superior monitor entry already exists: '*string1*'.

Explanation: LDAP entry already exists.

Action: None.

INF(IDSJAPI012): Workflow status writer: Merging duplicate value '*string1*' of attribute '*string2*' of monitor entry '*string3*'.

Explanation: Merging attributes is required.

Action: None.

INF(IDSJAPI013): Workflow status writer: Retrying to add entry with merged attributes, DN = '*string1*'

Explanation: Merging attributes is required.

Action: None.

WAR(IDSJAPI014): *string1* while setting a value; Class name of object: *string2*, method name: *string3*, value (class name: *string4*): '*string5*'.

Explanation: The setter method for changing the value of a request workflow or a request activity can't be set correctly due to unexpected exception.

Action: Check the parameters listed in the message.

WAR(IDSJAPI015): Workflow status writer couldn't reconnect to LDAP (for entry '*string1*'). No monitor entries are created. Retry limit reached. Error: *string2*.

Explanation: Internal error while creating an LDAP connection.

Action: Inform your DirX Identity support.

WAR(IDSJAPI016): Workflow status writer couldn't reconnect to LDAP (for entry '*string1*'). No monitor entries are created. Non-temporary LDAP error: *string2*.

Explanation: Internal error while creating an LDAP connection.

Action: Inform your DirX Identity support.

WAR(IDSJAPI075): Ignored modification *string1*, '/' is NOT permitted!

Explanation: The given character is not allowed in the name. Modifications are activity-specific.

Action: Change the modification setting in your activity.

WAR(IDSJAPI076): Ignored modification *string1*, should be a string!

Explanation: The modification is not a instance of string.

Action: Change the modification setting in your activity.

16.5.13. Logging Messages (IDSJBS)

This topic describes the logging messages returned by the DirX Identity Server BackupSlaveListener.

ERR(IDSJBS001): Error setting up JMS publishing in BackupSlaveListener: '*string1*'.

Explanation: Probably messaging configuration problem.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJBS002): Error in slave repository.

Explanation: Probably error in file-system.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJBS003): Error when recovering stored message in slave.

Explanation: Technical error.

Action: Inform DirX Identity technical support about the problem.

INF(IDSJBS004): Message replayed for topic *string1* and event '*string2*'.

Explanation: Message is recovered from backup repository and sent to master broker.

Action: None.

INF(IDSJBS005): Adaptor backup routine '*string1*' disconnected.

Explanation: Successfully disconnected.

Action: None.

ERR(IDSJBS006): Error shutting down publishing in adaptor backup: '*string1*'.

Explanation: Probably messaging configuration problem.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJBS007): Publishing backup message failed in adaptor '*string1*': *string2*.

Explanation: See why publishing does not work.

Action: Check the messaging service.

ERR(IDSJBS008): Internal error (cause: *string1*).

Explanation: Server encountered severe difficulties.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJBS009): Received a not supported message with messageID '*string1*' JMStype '*string2*'.

Explanation: A not supported message was received which is not a TextMessage or does not contain an operation code.

Action: Check the client which produces such a message.

ERR(IDSJBS010): Listener is in replay mode. Message with messageID '*string1*' JMStype '*string2*' ignored.

Explanation: Currently the backup listener is in replay mode. No new arriving messages are stored.

Action: Check the client which produces such a message.

WAR(IDSJBS011): Event with external ID '*string1*' not in BackupSlaveListerner repository found.

Explanation: The given event can not be deleted from the repository.

Action: If this happens frequently then inform DirX Identity technical support about the problem.

INF(IDSJBS012): Calling replayBackupedMessages finished successfully. Sent messages *string1*.

Explanation: Method to replay stored events was called.

Action: None.

16.5.14. Logging Messages (IDSJCFG)

This topic describes the logging messages returned by the DirX Identity Server Configuration.

ERR(IDSJCFG007): Unsuccessful LDAP search: *string1*.

Explanation: Probably wrong configuration.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJCFG020): Invalid classloader/baseurl: *string1*

Explanation: The URL for the classloader is malformed.

Action: Correct the configuration.

WAR(IDSJCFG040): Can’t enable audit connector: disabledAuditConnector not found

Explanation: The node disabledAuditConnector was not found.

Action: Correct the configuration.

WAR(IDSJCFG041): Can’t disable audit connector: disabledAuditConnector not found

Explanation: The node disabledAuditConnector/connector was not found.

Action: Correct the configuration.

WAR(IDSJCFG042): No attribute value for *string1* found in *string2*.

Explanation: Certain attribute values are not filled by default. This might not really result in an error.

Action: Correct configuration if you think the value is needed.

INF(IDSJCFG043): No attribute value for *string1* found in *string2*.

Explanation: Certain attribute values are not filled by default. This might not really result in an error.

Action: Correct configuration if you think the value is needed.

ERR(IDSJCFG044): Exception *string1* occurred while cloning job configuration of activity *string2*: *string3*.

Explanation: The jobConfig DOM node could not be cloned when cloning the job configuration. Thus, nationalization of mail texts may result in the wrong language.

Action: Check the message details. Correct installation and class loading problems.

WAR(IDSJCFG045): Job definition could not be loaded from the workflow's XML definition. Workflow: '*string1*'.

Explanation: There is an error while loading the XML definition for 'job' element.

Action: Check the workflow definition. If ok, inform the DirX Identity support as there might be XML parsing problems.

INF(IDSJCFG050): Found UserMapping *string1*

Explanation: UserMapping class added to list of configuration classes.

Action: None.

WAR(IDSJCFG051): UserMapping definition incorrect, tag *string1* ignored

Explanation: User mapping malformed.

Action: Correct configuration.

WAR(IDSJCFG061): Missing parameters in configuration for finding participants in activity '*string1*'.

Explanation: No participants finder configuration found.

Action: Check the workflow definition. If ok, inform the DirX Identity support as there might be XML parsing problems.

ERR(IDSJCFG062): Exception *string1* occurred reading identifier from subject order.

Explanation: Cannot read identifier from subject.

Action: Check your workflow definition.

ERR(IDSJCFG063): Can’t get subjectDN from: *string1*.

Explanation: Cannot get subjectDN from the order.

Action: Check your workflow definition.

ERR(IDSJCFG064): Cannot find participants in activity *string1* because of invalid participant finder type '*string2*'.

Explanation: Invalid participant configuration.

Action: Check your workflow definition.

ERR(IDSJCFG065): *string1* when finding participants from group '*string2*' in activity '*string3*': *string4*.

Explanation: Exception occurred when finding participants.

Action: Check your workflow definition.

ERR(IDSJCFG066): *string1* when trying to find participants by LDAP filter for activity '*string2*' parsing '*string3*' or filter '*string4*': *string5*.

Explanation: Exception occurred when finding participants.

Action: Check your workflow definition.

ERR(IDSJCFG067): Only *string1* configuration values for finding participants by access policy in activity'*string2*'.

Explanation: Invalid configuration.

Action: Check your workflow definition.

WAR(IDSJCFG068): Missing resource order - cannot get participants in activity '*string1*' by access policy for privileges.

Explanation: Invalid configuration.

Action: Check your workflow definition.

ERR(IDSJCFG069): *string1* when finding participants by class '*string2*' in activity '*string3*': *string4*.

Explanation: Invalid configuration. Check the exception message.

Action: Check your workflow definition.

ERR(IDSJCFG070): *string1* when instantiating *string2* class '*string3*': *string4*.

Explanation: Invalid configuration. Check the exception message.

Action: Check your workflow definition.

INF(IDSJCFG071): Context class loader *string1* failed to load class '*string2*': *string3*. Trying

again with local class loader.

Explanation: Loading a class failed. Another ClassLoader is tried.

Action: None.

ERR(IDSJCFG072): *string1* when trying to load class '*string2*': *string3*.

Explanation: Loading a class failed.

Action: Check your location of the custom jar file. If OK, inform the DirX Identity support.

ERR(IDSJCFG073): Missing configuration for finding participants in activity '*string1*'.

Explanation: Invalid configuration.

Action: Check your workflow definition.

ERR(IDSJCFG60): Failed to parse XML content: *string1*.

Explanation: XML parsing failed because of given exception.

Action: Correct configuration.

16.5.15. Logging Messages (IDSJJOB)

This topic describes the logging messages returned by the DirX Identity Server built-in jobs.

ERR(IDSJJOB000): Internal error (cause: *string1*)

Explanation: An unexpected error occurred during copying elements from the in-channel to the out-channel.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJJOB002): Invalid channel entry *string1*. Should be IDMRequest or IDMResponse.

Explanation: Combines only entries of given class.

Action: If this happens often inform DirX Identity technical support about the problem.

WAR(IDSJJOB003): Couldn't make a copy of *string1*

Explanation: A copy could not be made. See detailed exception message for more information.

Action: If this happens often inform DirX Identity technical support about the problem.

INF(IDSJJOB040): FindDeadlock:JVM does not support monitoring of ownable synchronizer (deadlocks monitoring).

Explanation: JVM does not support monitor function to find deadlocks.

Action: None.

INF(IDSJJOB041): FindDeadlock: No deadlock found.

Explanation: No deadlock can be found by JVM's method.

Action: Not all deadlocks found by the JVM. Several thread dump might still show a deadlock.

INF(IDSJJOB042): FindDeadlock: File '*string1*' with thread and deadlock info written.

Explanation: File with thread and deadlock info written.

Action: None.

ERR(IDSJJOB100): Compilation problem: *string1*

Explanation: Compiling failed. See the given messages.

Action: Check the error messages and correct the Java source.

WAR(IDSJJOB101): Failed to discard temporary files (cause: *string1*)

Explanation: Cleanup failed. See detailed message.

Action: If this happens often inform DirX Identity technical support about the problem.

INF(IDSJJOB150): Discarded e-mail to '*string1*', see the SMTP service configuration in the provisioning configuration (Map mail address); subject: '*string2*'.

Explanation: The SMTP service defined in the provisioning configuration contains a special value in the Map mail address property. If you enter dummy into this field, then no e-mail is sent at all.

Action: Check the settings of the SMTP service object in the provisioning configuration. Correct the value in the Map mail address property to enable sending e-mails. See the documentation for possible settings.

INF(IDSJJOB151): Sending e-mail to/from '*string1*' instead of '*string2*'; subject: '*string3*'.

Explanation: E-mail is delegated to another recipient.

Action: Check the configuration.

WAR(IDSJJOB152): Failed to convert e-mail address '*string1*' (subject: '*string2*'; cause: *string3*).

Explanation: Converting e-mail address failed. See detailed message.

Action: Check the configuration.

INF(IDSJJOB153): Notification to '*string1*' succeeded; subject: '*string2*'.

Explanation: Sending notification was successfully processed.

Action: None.

ERR(IDSJJOB154): Notification to '*string1*' failed (cause: *string2*)

Explanation: Sending e-mail failed. See detailed message.

Action: Check the configuration. Depending on the circumstances, inform DirX Identity technical support about the problem.

WAR(IDSJJOB155): Prevented delivery of e-mail (subject is '*string1*') since recipient is null!

Explanation: Mandatory parameter recipient is missing.

Action: Check the configuration.

WAR(IDSJJOB156): Recipient in field '*string1*' ignored because address is empty (subject is '*string2*').

Explanation: Recipient ignored.

Action: Check the configuration.

ERR(IDSJJOB157): Notification to '*string1*' from '*string2*' with subject '*string3*' and body '*string4*' failed (cause: *string5*)

Explanation: Sending e-mail failed. See detailed message.

Action: Check the configuration. Depending on the circumstances, inform DirX Identity technical support about the problem.

WAR(IDSJJOB158): Could not trigger further processing of activity '*string1*' of workflow with ID '*string2*'.

Explanation: Putting a new task into the java space failed because not all required parameters could be accessed (internal key map).

Action: Ask DirX Identity technical support about the issue.

ERR(IDSJJOB159): Exception *string1* when putting modifications for activity *string2* in workflow with UID='*string3*' with operation *string4* to space: *string5*.

Explanation: After sending a e-mail the configured modifications should be applied. The sendMail thread tries to pass these to a workflow thread via JavaSpace, but this fails.

Action: Check the message details and inform DirX Identity technical support about the problem.

INF(IDSJJOB160): No e-mail will be sent as e-mail user hook threw an exception.

Explanation: When e-mail user hook throws an exception, no e-mail is sent.

Action: Check the e-mail user hook.

WAR(IDSJJOB161): Decryption of the e-mail server’s password '*string1*' failed; error: *string2*.

Explanation: Decryption of the password failed for the given reason.

Action: Ask DirX Identity technical support about the issue.

WAR(IDSJJOB162): Sending an e-mail failed, but putting the e-mail request to the retry channel failed. Error: *string1*.

Explanation: Sending an e-mail failed, but putting the e-mail request to the retry channel failed.

Action: Ask DirX Identity technical support about the issue.

INF(IDSJJOB163): Sending an e-mail failed. E-mail request (To: '*string1*', subject: '*string2*') is put to the retry channel for processing later on.

Explanation: Sending an e-mail failed and therefore the e-mail request is put to the retry channel.

Action: None.

INF(IDSJJOB164): Prevented delivery of e-mail (subject is '*string1*') since no certificate for any of the recipients is available or user certificates are invalid!

Explanation: For none of the recipients a (valid) user certificate is available.

Action: Check the attribute userCertificate of the recipients.

WAR(IDSJJOB165): E-mail address '*string1*' found more than once in LDAP; first user with a valid user certificate is used.

Explanation: Several user objects hold the same e-mail address. First user with a valid user certificate is chosen.

Action: Check the e-mail addresses in your LDAP directory.

WAR(IDSJJOB166): User certificate of user '*string1*' (mail: '*string2*') is invalid.

Explanation: User certificate is invalid and therefore no encrypted e-mail can be sent.

Action: Check the user certificate in your LDAP directory.

WAR(IDSJJOB167): No user certificate available for user '*string1*' (mail: '*string2*').

Explanation: User certificate is not available and therefore no encrypted e-mail can be sent.

Action: Check the user certificate in your LDAP directory.

WAR(IDSJJOB168): Retrieval of user certificate for e-mail address '*string1*' failed: error: *string2*. No encrypted e-mail is sent.

Explanation: LDAP search operation failed with the given reason. No e-mail can be sent as the user certificate is unavailable.

Action: Check the LDAP error message.

WAR(IDSJJOB169): Loading and checking user certificate for e-mail address '*string1*' failed: error: *string2*.

Explanation: Retrieval of user certificate was successful but the user certificate could not be loaded as JAVA class. No e-mail can be sent as the user certificate is unavailable.

Action: Check the error message.

WAR(IDSJJOB170): Problem while generating recipient information for user '*string1*' (mail address '*string2*'): error: *string3*. No encrypted e-mail is sent.

Explanation: Retrieval of user certificate was successful but the user certificate could not be loaded as JAVA class. No e-mail can be sent as the user certificate is unavailable.

Action: Check the error message.

ERR(IDSJJOB171): Connecting to certificate server (Host: *string1*, Port: *string2*, User: *string3*) failed; error: *string4*. No encrypted e-mail is sent.

Explanation: Connection to certificate server could not be established. No e-mail can be sent as the user certificate is unavailable.

Action: Check the error message and/or the connection parameters for the certificate server.

WAR(IDSJJOB172): Decryption of the certificate server's password '*string1*' failed; error: *string2*.

Explanation: Decryption of the password failed for the given reason.

Action: Ask DirX Identity technical support about the issue.

WAR(IDSJJOB173): No user certificate available for any user with e-mail address '*string1*'.

Explanation: User certificate is not available and therefore no encrypted e-mail can be sent.

Action: Check the user certificate in your LDAP directory.

WAR(IDSJJOB174): Sending an e-mail failed, but putting the e-mail request to the error channel failed. Error: *string1*.

Explanation: Sending an e-mail failed, but putting the e-mail request to the error channel failed.

Action: Ask DirX Identity technical support about the issue.

INF(IDSJJOB175): Sending an e-mail failed. E-mail request (To: '*string1*', subject: '*string2*') is put to the error channel.

Explanation: Sending an e-mail failed and therefore the e-mail request is put to the error channel.

Action: None.

WAR(IDSJJOB176): Sending an e-mail failed, but putting the e-mail request to the response channel failed. Error: *string1*.

Explanation: Sending an e-mail failed, but putting the e-mail request to the response channel failed.

Action: Ask DirX Identity technical support about the issue.

INF(IDSJJOB200): Backup started.

Explanation: Synchronized backup is started.

Action: Check the configuration.

INF(IDSJJOB201): Backup succeeded.

Explanation: Backup was written.

Action: None.

WAR(IDSJJOB202): Backup is not enabled.

Explanation: Backup is not be enabled.

Action: Check the Java-based server configuration if backup is enabled.

ERR(IDSJJOB203): No backup path given for the DirX Messaging Server. No backup is written.

Explanation: You have to give a full qualified backup path for the DirX Messaging Server.

Action: Change the configuration of the workflow.

ERR(IDSJJOB204): No backup path given for the Connectivity datastore. No backup is written.

Explanation: You have to give a full qualified backup path for the Connectivity datastore.

Action: Change the configuration of the workflow.

ERR(IDSJJOB205): No backup path given for the Provisioning datastore. No backup is written.

Explanation: You have to give a full qualified backup path for the Provisioning datastore.

Action: Change the configuration of the workflow.

INF(IDSJJOB206): No target path and no userclass given. The default post backup operation is not executed.

Explanation: You have to provide your own means to move the written backup files to the desired location.

Action: None.

WAR(IDSJJOB207): Cannot create or access target path '*string1*'. The default post backup operation is not executed.

Explanation: You have to provide your own means to move the written backup files to the desired location.

Action: Check your target path.

ERR(IDSJJOB208): Default backup post operation: Cannot move file '*string1*' to '*string2*':

string3.

Explanation: The file could not moved.

Action: Check the given exception message.

ERR(IDSJJOB209): Default backup post operation: Cannot move directory '*string1*' to '*string2*': *string3*.

Explanation: The directory could not moved.

Action: Check the given exception message.

ERR(IDSJJOB210): Removing file '*string1*' for cleanup after failed backup creation failed.

Explanation: The file could not removed.

Action: Check the given file.

ERR(IDSJJOB211): Removing directory '*string1*' for cleanup after failed backup creation failed.

Explanation: The directory and its content could not removed.

Action: Check the given file.

WAR(IDSJJOB250): Userclass not supported yet.

Explanation: Functionality is not supported yet.

Action: Ask DirX Identity technical support about the issue.

ERR(IDSJJOB251): Creating (parent) directory '*string1*' failed: *string2*.

Explanation: Creating the directory including parent directories failed.

Action: Check the given pathname (access rights).

ERR(IDSJJOB300): JMSException while creating JMS message: '*string1*'.

Explanation: Message can not be created.

Action: Call DirX Identity support.

ERR(IDSJJOB301): Cannot create session for JMS notification: message service to connect to '*string1*'.

Explanation: Message can not be sent.

Action: Check message service.

ERR(IDSJJOB302): JMSException while sending JMS notification: '*string1*'.

Explanation: Message can not be sent.

Action: Call DirX Identity support.

ERR(IDSJJOB303): Exception while sending notification request: *string1*.

Explanation: An exception occurred while creating the e-mail object or while sending the e-mail itself.

Action: Check the detailed message.

ERR(IDSJJOB350): Exception while parsing SMS text message for mobile no. '*string1*' of account '*string2*' with input text '*string3*'.

Explanation: The placeholders in the given text message cannot be replaced.

Action: Check the detailed message.

WAR(IDSJJOB351): Wrong event type '*string1*' received. Event is not processed.

Explanation: A not expected event type is received. This event is not processed.

Action: Check the detailed message. Check who is the sender of this wrong event.

ERR(IDSJJOB352): Plugin class '*string1*' for SMS gateway can not be found.

Explanation: The mandatory class can not be found.

Action: Check your jar file with the implementation.

ERR(IDSJJOB353): Plugin class '*string1*' for SMS gateway can not be casted to interface 'SMSGatewayPlugin'.

Explanation: The given plugin class does not fit to the interface.

Action: Check your implementation.

ERR(IDSJJOB354): Plugin class '*string1*' for SMS gateway can not be instantiated and opened.

Explanation: The given plugin class does not fit to the interface.

Action: Check the detailed exception message.

ERR(IDSJJOB355): Mandatory plugin class is not defined.

Explanation: You must supply a plugin class name.

Action: Add the name of your plugin class implementation.

WAR(IDSJJOB356): Exception occurred while closing SMS plugin.

Explanation: An exception occurred.

Action: Check the detailed message.

ERR(IDSJJOB357): Cannot call SMS plugin because some mandatory fields are missing.

Explanation: Missing fields can be destination URL, pluginclass, or username and password if authentication is needed.

Action: Supply the missing mandatory fields.

ERR(IDSJJOB358): Unexpected error while initializing crypt support function for text message for mobile no. '*string1*' of account '*string2*' with input text '*string3*'.

Explanation: The crypt functionality cannot be used.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJJOB359): No one-time password value given in message for text message for text message for mobile no. '*string1*' of account '*string2*' with input text '*string3*'.

Explanation: Cannot send text message without one-time password value.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJJOB370): destination URL not defined - no text message sent.

Explanation: Mandatory parameter not defined.

Action: Define the URL.

WAR(IDSJJOB371): '*string1*' is null or empty - no text message sent.

Explanation: Mandatory parameter not defined.

Action: Define the missing parameter.

WAR(IDSJJOB372): authenticate is true but no user or password given no authentication is used.

Explanation: Parameter not defined.

Action: Define the missing parameter or set authenticate to FALSE.

WAR(IDSJJOB373): Proxy server *string1* is invalid - not using any proxy.

Explanation: Missing proxy parameter.

Action: Define the missing parameter.

ERR(IDSJJOB374): An exception occurred while sending the request.

Explanation: A general exception occurred.

Action: Check the detailed exception message.

WAR(IDSJJOB375): Proxy server port is missing - not using any proxy.

Explanation: Missing proxy parameter.

Action: Define the missing parameter.

WAR(IDSJJOB376): Couldn’t decrypt password: '*string1*'.

Explanation: Password given for authentication to SMS gateway couldn’t be decrypted.

Action: Check the given exception.

WAR(IDSJJOB377): No crypt support provided. Can’t decrypt/descramble passwords.

Explanation: Password given for authentication to SMS gateway couldn't be decrypted/descrambled.

Action: Check the Java server log files.

ERR(IDSJJOBEMailNotification): no JMS_Context found in context.

Explanation: EMailNotification can not find context to get a connection to a JMS provider.

Action: Call DirX Identity support.

16.5.16. Logging Messages (IDSJUTL)

This topic describes the logging messages returned by the DirX Identity Server Utils.

WAR(IDSJUTL000): Couldn't initialize JMX support for '*string1*'.

Explanation: Failed to initialize the Java Management Extension for the denoted component.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJUTL002): The XML parser isn't XInclude aware or does not support DEFER_NODE_EXPANSION: *string1*.

Explanation: XML Parser doesn't support xsi.include instructions or to defer node expansion.

Action: None.

WAR(IDSJUTL003): ignored filter instruction *string1*.

Explanation: Failed to apply logging level filter.

Action: None.

WAR(IDSJUTL004): Unexpected problems encountered (cause: *string1*).

Explanation: An internal error occurred, check the message text for details.

Action: Inform DirX Identity technical support about the problem.

WAR(IDSJUTL005): NumberFormatException occurred while reading configuration parameter '*string1*' (value: '*string2*'). The default value '*string3*' will be used.

Explanation: The given string of the parameter cannot be parsed into a number.

Action: Check the given configuration parameter.

WAR(IDSJUTL006): *string1* exception while performing place holder replacement for source string '*string2*'.

Explanation: The source string whose place holders should be replaced is wrong.

Action: Check the source string.

WAR(IDSJUTL007): *string1* exception while performing generic place holder replacement for source string '*string2*'.

Explanation: The source string whose place holders should be replaced is wrong.

Action: Check the source string.

WAR(IDSJUTL008): Xerces parsing warning '*string1*'.

Explanation: Xerces parsing warning occurred.

Action: Check the source string.

ERR(IDSJUTL039): Unexpected problems encountered (cause: *string1*).

Explanation: A serious internal error occurred, check the message text for details.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJUTL040): Failed to instantiate XML document (cause: *string1*).

Explanation: Possible configuration error, check the message text for details.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJUTL041): Failed to load document from String '*string1*' (cause: *string2*).

Explanation: Possible configuration error, check the message text for details.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJUTL042): Exception in getProperty of XML configuration of path '*string1*' with defaults '*string2*': Stacktrace *string3* URL *string4*).

Explanation: Possible configuration error, check the message text for details.

Action: Inform DirX Identity technical support about the problem.

ERR(IDSJUTL043): Xerces (Fatal) parsing error '*string1*'.

Explanation: Xerces parsing (fatal) error occurred.

Action: Check the source string.

16.5.17. Logging Messages (IOJ)

This topic describes the logging messages returned by the Java-based DirX Identity server order job integration.

INF(IOJ200): All provisioning changes are implemented.

Explanation: The Check Provisioning activity succeeds since all expected provisioning changes are implemented.

Action: None.

INF(IOJ201): Some provisioning changes are still pending.

Explanation: The Check Provisioning activity goes to a waiting state since the expected provisioning changes are not yet fully implemented.

Action: None.

INF(IOJ202): Membership *string1=string2* has not yet changed for DN=*string3*.

Explanation: The Check Provisioning activity is waiting for a change of the membership.

Action: None.

INF(IOJ203): Membership *string1=string2* change is implemented for DN=*string3*.

Explanation: The Check Provisioning detected the implementation of the membership change.

Action: None.

INF(IOJ204): The change of the TS state for DN=*string1* to '*string2*' is still pending.

Explanation: The Check Provisioning activity is waiting for a change of the dxrTSState attribute.

Action: None.

INF(IOJ205): The change of the TS state for DN=*string1* to '*string2*' is implemented.

Explanation: The Check Provisioning detected the implementation of the dxrTSState change.

Action: None.

INF(IOJ206): No related ticket found for workflow instance ID=*string1*. Creating ticket for due date *string2*.

Explanation: The workflow instance contains an order with due date but no related ticket is found. The missing tickets are created.

Action: None.

INF(IOJ207): No related ticket found for workflow instance ID='*string1*'.

Explanation: The workflow instance contains an order with activation date but no related ticket is found.

Action: None.

INF(IOJ208): Update of an order in LDAP will not be done as the workflow is locked (Workflow: '*string1*'); 'changesApplied' flag will not be set.

Explanation: Successful execution of an order could not be stored in LDAP (in the order) because the workflow is locked.

Action: Check the error problem.

WAR(IOJ412): Job finished with errors: *string1*.

Explanation: Check the detailed messages of the job's run.

Action: Check and repair the source of the errors.

WAR(IOJ413): Job finished with errors.

Explanation: Check the detailed messages of the job's run.

Action: Check and repair the source of the errors.

WAR(IOJ414): Exception *string1* in JobSessionFactory: Cannot get session from pool: *string2*.

Explanation: Failed to get a SvcSession from the session pool. Thus, a session clone is used.

Action: Check the message details.

WAR(IOJ415): Exception *string1* occurred in JobSessionFactory while returning session to pool: *string2*.

Explanation: Failed to return a SvcSession to the session pool.

Action: Check the message details. Maybe a reload configuration cleared the pool.

WAR(IOJ420): A second order exists to add attribute=*string1* value=*string2* to object DN=*string3*.

Explanation: Some modifications created by a ticketing workflow are superfluous: Multiple attempts to add the same attribute value to an object.

Action: Check your privilege structure and your workflow structure.

WAR(IOJ421): A second order exists to remove attribute=*string1* value=*string2* from object DN=*string3*.

Explanation: Some modifications created by a ticketing workflow are superfluous: Multiple attempts to remove the same attribute value from an object.

Action: Check your privilege structure and your workflow structure.

WAR(IOJ422): A second order exists to add object DN=*string1*.

Explanation: Some modifications created by a ticketing workflow are superfluous: Multiple orders to add the same object.

Action: Check your privilege structure and your workflow structure.

WAR(IOJ423): A second order exists to delete object DN=*string1*.

Explanation: Some modifications created by a ticketing workflow are superfluous: Multiple orders to delete an object.

Action: Check your privilege structure and your workflow structure.

WAR(IOJ430): Updating ticket state to '*string1*' failed for ticket '*string2*' related to workflow '*string3*': *string4*.

Explanation: The state of a related ticket could not be set.

Action: Check the message details. Set state manually. Wait for the TicketProcessor workflow to set the state.

WAR(IOJ431): Exception '*string1*' occurred while searching for tickets related to workflow instance '*string2*': *string3*. Cannot set ticket state.

Explanation: The related ticket could not be determined. Thus, its state cannot be set. Maybe the ticket is removed.

Action: Check the message details. Wait for the TicketProcessor workflow to set the state.

WAR(IOJ432): Exception '*string1*' occurred while searching workflow instance '*string2*': *string3*. Cannot set ticket state.

Explanation: The workflow instance cannot be determined from its ID. The related tickets cannot be searched. Setting the state at related tickets is not possible.

Action: Check the message details. Wait for the TicketProcessor workflow to set the state.

WAR(IOJ433): Cannot find workflow instance for instance id='*string1*': No entry returned by the search. Cannot set ticket state.

Explanation: The workflow instance cannot be determined from its ID. The related tickets cannot be searched. Setting the state at related tickets is not possible.

Action: Wait for the TicketProcessor workflow to set the state.

WAR(IOJ434): Duplicate workflow instances found for instance id='*string1*'. Cannot set ticket state.

Explanation: The workflow instance cannot be determined from its ID. The related tickets cannot be searched. Setting the state at related tickets is not possible.

Action: Wait for the TicketProcessor workflow to set the state.

WAR(IOJ435): Conversion of order to LDAP attribute failed (Workflow: '*string1*'). Order will not be updated in LDAP ('changesApplied' flag will not be set).

Explanation: Successful execution of an order could not be stored in the order.

Action: Check the error problem.

WAR(IOJ436): Update of an order in LDAP failed (Workflow: '*string1*'); 'changesApplied' flag will not be set.

Explanation: Successful execution of an order could not be stored in LDAP (in the order).

Action: Check the error problem.

ERR(IOJ500): Cannot access bind profile manager.

Explanation: ApplyChanges has no access to the bind profile manager. Maybe the job does not run privileged or it runs on a remote worker.

Action: Use a resource family for the activity that assures running the ApplyChanges job collocated. Add ApplyChanges to the privileged jobs in the server configuration.

ERR(IOJ501): Cannot access bind profile 'private/domain'.

Explanation: ApplyChanges has no access to the bind profile 'private/domain'. Maybe the profile does not exist, the job does not run privileged or it runs on a remote worker.

Action: Check/create the private bind profile for the domain. Use a resource family for the activity that assures running the ApplyChanges job collocated. Add ApplyChanges to the privileged jobs in the server configuration.

ERR(IOJ502): Port is missing in bind profile 'private/domain'.

Explanation: No port is configured in the domain’s bind profile.

Action: Add LDAP server port to the domain’s bind profile.

ERR(IOJ503): An illegal port number is configured in 'private/domain'.

Explanation: An illegal port is configured in the domain’s bind profile.

Action: Enter a numeric port number > 0 in the domain’s bind profile.

ERR(IOJ504): The workflow using ApplyChanges is not a request workflow. Aborting ApplyChanges.

Explanation: The job ApplyChanges is configured in a workflow that is not a request workflow. Thus, ApplyChanges cannot access subject and resources, and ApplyChanges is aborted.

Action: Correct your workflow configuration.

ERR(IOJ505): Cannot create job of class *string1*: *string2*.

Explanation: The job class cannot be loaded. Thus, the job cannot be run.

Action: Check your installation, especially if you use an own job class. Check your job class name for typing errors.

ERR(IOJ506): Creating a connected template session failed: *string1*.

Explanation: The session pool is built from one template session being connected to the data store. For some reason the template session cannot be created in a bound state.

Action: Check the bind credentials. Check class loading. Check if the LDAP server is up and accepts connections.

ERR(IOJ507): Error in JobSessionFactory: Cannot get template session.

Explanation: The session clone is built from one template session being connected to the data store. For some reason the template session cannot be created in a bound state.

Action: Check the preceding messages.

ERR(IOJ508): Exception *string1* in JobSessionFactory: Cannot get session clone: *string2*.

Explanation: The job failed to create a new session as a clone of the template session.

Action: Check the message details.

ERR(IOJ509): Exception *string1* occurred in workflow inst=*string2* while processing workflow info, startedRequestWorfklows=*string3*: *string4*.

Explanation: An exception occurred while constructing the workflow info from the startedRequestWorfklows context attribute. Update of started child workflows failed.

Action: Check the message details.

ERR(IOJ510): Cannot read identifier for workflow ID=*string1* from repository: Identifier not found.

Explanation: The workflow with the given uid does not exist in the repository. Its context is not updated.

Action: Check the installation and preceeding messages.

ERR(IOJ511): Exception *string1* occurred while reading identifier for workflow ID=*string2* from repository: *string3*. Context not updated.

Explanation: An error occurred. The workflow's context is not updated.

Action: Check the message details.

ERR(IOJ512): Exception *string1* occurred while reading workflow ID=*string2* from repository: *string3*.

Explanation: An error occurred. The workflow's context is not updated.

Action: Check the message details.

ERR(IOJ513): Exception *string1* occurred while updating context of workflow ID=*string2*: *string3*.

Explanation: An error occurred. The workflow's context is not updated.

Action: Check the message details.

ERR(IOJ520): Orders exist to add AND to delete attribute=*string1* value=*string2* for object DN=*string3*.

Explanation: Some modifications created by a ticketing workflow are inconsistent: Modifications exist to add and to delete the same attribute value to/from an object.

Action: Check your privilege structure and your workflow structure.

ERR(IOJ521): Orders exist to add AND to delete object DN=*string1*.

Explanation: Some modifications created by a ticketing workflow are inconsistent:

Orders exist to add and to delete the same object.

Action: Check your privilege structure and your workflow structure.

ERR(IOJ522): Reading object DN=*string1* from the data store failed: *string2*. Cannot determine its TS state.

Explanation: An object's TS state cannot be determined since reading the object from the LDAP server failed.

Action: Check the details of the message.

ERR(IOJ523): Reading object DN=*string1* from the data store failed: *string2*. Cannot determine its member state.

Explanation: An object's member state cannot be determined since reading the object from the LDAP server failed.

Action: Check the details of the message.

ERR(IOJ524): Exception *string1* occurred in while processing workflow change tracking info for id=*string2*, accountGroupChanges=*string3*: *string4*.

Explanation: The account group change info cannot be processed. Presumably the data is corrupt.

Action: Check the message details.

ERR(IOJ525): Cannot read identifier from add request *string1*: Cannot check provisioning.

Explanation: The add request in the change tracking data is corrupt. It does not contain an identifier. Thus, the object to be added cannot be determined.

Action: Check the software creating the change tracking info. Check the installation for class loading problems.

ERR(IOJ526): Cannot read identifier from modify request *string1*: Cannot check provisioning.

Explanation: The modify request in the change tracking data is corrupt. It does not contain an identifier. Thus, the object to be modified cannot be determined.

Action: Check the software creating the change tracking info. Check the installation for class loading problems.

ERR(IOJ527): Cannot read identifier from delete request *string1*: Cannot check provisioning.

Explanation: The delete request in the change tracking data is corrupt. It does not contain an identifier. Thus, the object to be deleted cannot be determined.

Action: Check the software creating the change tracking info. Check the installation for class loading problems.

ERR(IOJ528): The name of the object description to create the new object cannot be retrieved.

Explanation: The subject add order cannot be created since no object description name is provided.

Action: Check the preceding error messages.

ERR(IOJ529): Creating the add order for the subject failed: *string1*.

Explanation: The subject add order cannot be created from the temporary object.

Action: Check the installation and the message details.

ERR(IOJ530): Reading the associated object from the data store as a template for subject creation failed: *string1*.

Explanation: An error occurred reading the template object for subject creation.

Action: Check the message details. Check network and server connection.

ERR(IOJ531): The associated object's DN is missing in the workflow context: The context attribute '*string1*' is empty.

Explanation: The workflow context attribute expecting to contain the associated entry's DN is empty.

Action: Check the component that was starting the workflow. It should pass the parameter to the workflow context.

ERR(IOJ532): The configuration parameter 'contextAttributeForEntry' is empty.

Explanation: This parameter has to contain the name of the workflow context attribute holding the dn the associated entry.

Action: Complete the activity configuration by adding this name (e.g. associatedEntry).

ERR(IOJ533): Cannot read the object description name to create the object of type '*string1*' from template *string2* of type '*string3*'.

Explanation: The name of the object description to be used to create the object cannot be read from the template object's object description.

Action: Configure the objectDescriptionName for the destinationType in the template's createFrom tag.

ERR(IOJ534): The workflow context attribute '*string1*' does not exist.

Explanation: The workflow context attribute expecting to contain the destination type is empty.

Action: Check the component that was starting the workflow. It should pass the parameter to the workflow context.

ERR(IOJ535): The configuration parameter 'contextAttributeForDestinationType' is empty.

Explanation: This parameter has to contain the name of the workflow context attribute holding the object description name of the object to be created.

Action: Complete the activity configuration by adding this name (e.g. destinationType).

16.5.18. Logging Messages (IPO)

This topic describes the logging messages returned by the DirX Identity connection and job pools.

ERR(IPO500): Error clearing pool of type *string1*: *string2*.

Explanation: An error occurred during clear of pool of the specified type. Presumable a connection could not be closed.

Action: Check detailed message.

ERR(IPO501): Value given for System property '*string1*' not correct. Using instead *string3*: *string4*.

Explanation: The given value for the System property is malformed.

Action: Check the given value for the System property.

16.5.19. Logging Messages (JAL)

This topic describes the logging messages returned by the IdS-J extension for the JMS Audit Log Handler that delivers audit records to DirX Audit.

WAR(JAL001): Exception '*string1*' while configuring extensions '*string2*': *string3*.

Explanation: JMS Audit Logger Extensions will not be manageable via JMX due to initialization error.

Action: Check the configuration of the audit log handler and correct it if necessary. If the problem persists inform DirX Identity technical support.

WAR(JAL002): Exception '*string1*' while opening the JMS factory: *string2*.

Explanation: JMS Audit Logger Extensions could not open the connection to the JMS provider. It tries to connect later, when the next audit record is to be published.

Action: Check that the JMS messaging server of DirX Audit is started. Check the address and the other configuration options. Correct the configuration if necessary.

WAR(JAL003): Exception '*string1*' when sending audit record to JMS: *string2*.

Explanation: JMS Audit Logger Extensions could not publish an audit record to the JMS queue. In case of a lost connection, it tries to reconnect and resend it.

Action: Check that the JMS messaging server of DirX Audit is started. Check the address and the other configuration options. Correct the configuration if necessary.

WAR(JAL004): Exception '*string1*' when sending audit record to JMS: *string2*.

Explanation: The JMS Audit Logger Extensions could not publish an audit record to the JMS queue after reconnecting. It saves the record to a local file and tries to resend it

after it has reestablished the connection.

Action: Check that the JMS messaging server of DirX Audit is started. Check the address and the other configuration options. Correct the configuration if necessary.

ERR(JAL005): No queue Sender available.

Explanation: The JMS Audit Logger Extensions could not create the JMS sender queue.

Action: Check that the JMS messaging server of DirX Audit is started. Check the address and the other configuration options. Correct the configuration if necessary.

WAR(JAL006): Exception '*string1*' when sending audit record to JMS: *string2*.

Explanation: The JMS Audit Logger Extensions could not publish an audit record to the JMS queue after reconnecting. It saves the record to a local file and tries to resend it after it has reestablished the connection.

Action: Check that the JMS messaging server of DirX Audit is started. Check the address and the other configuration options. Correct the configuration if necessary.

ERR(JAL007): Missing configuration option(s): *string1*.

Explanation: The JMS Audit Logger Extensions need these option(s) in order to connect to a JMS provider.

Action: Check the configuration options and provide values for missing ones.

WAR(JAL008): Exception '*string1*' while configuring extension '*string2*': *string3*.

Explanation: The JMS Audit Logger Extensions could not be started due to an incorrect configuration.

Action: Check the handler configuration in the LDAP server or in local extension file. Correct the configuration.

WAR(JAL009): The configured log path '*string1*' is invalid: *string2*.

Explanation: The path for recoverable audit records of the JMS Audit Logger Extension is invalid: not existing, not a directory or not writable.

Action: Specify a valid log path. In the meantime audit records are temporarily stored to an extension home folder.

ERR(JAL010): The log path '*string1*' is invalid.

Explanation: The path for recoverable audit records of the JMS Audit Logger Extension is invalid. The logger cannot save audit records, if the JMS server is not available.

Action: Specify a valid log path.

ERR(JAL011): Exception '*string1*' when storing audit record to file '*string2*': *string3*.

Explanation: The JMS Audit Logger Extension could not save an audit record temporarily to a file.

Action: Make sure that the service has the access right to create a file in this folder.

ERR(JAL012): Exception '*string1*' when reading audit record file '*string2*': *string3*.

Explanation: The JMS Audit Logger Extension could not read an audit record from this file and could not send its content to the configured JMS provider. The file is ignored.

Action: Check the file and repair it. Use off-line tooling to send its content to the JMS provider.

ERR(JAL013): Exception '*string1*' when sending content of file '*string2*' to JMS: *string3*.

Explanation: The JMS Audit Logger Extension could not send an audit record from this file to the configured JMS provider. The file is ignored.

Action: Check the connection to the JMS provider. Use off-line tooling to send the file content later.

WAR(JAL014): Exception '*string1*' while opening the JMS temporary audit file folder: *string2*.

Explanation: The JMS Audit Logger Extensions could not create its temporary audit file folder. If later on the JMS Provider is not available, it cannot store audit records temporarily to files.

Action: Check the configuration options and the existence of the file folders.

INF(JAL015): Starting to send *string1* file(s) in '*string2*' to Message Server ...

Explanation: The JMS Audit Logger found files "audit*" in the temporary audit file folder. It expects one audit record per file and tries to send them to the message server.

Action: None.

INF(JAL016): Sent *string1* file(s) in '*string2*' to the Message server. Still *string3* file(s) left.

Explanation: The JMS Audit Logger found files "audit*" in the temporary audit file folder and sent them to the Audit message server.

Action: None.

WAR(JAL017): Could not delete audit record file: *string1*.

Explanation: The JMS Audit Logger sent the content of the file as audit record to the message server. Afterwards it could not delete this file. It continues with the remaining files.

Action: Delete the file manually.

ERR(JAL018): Cannot store audit record because file '*string1*' exists. Audit record is: *string2*.

Explanation: The JMS Audit Logger Extension tried to save more than 100 records in the same millisecond.

Action: Make sure the JMS message server is reachable, copy the record from the log to a file and move it to the appropriate input folder of the Audit Server.

WAR(JAL019): *string1* when sending temporary audit files in folder '*string2*': *string3*.

Explanation: JMS Audit Logger could not send an audit record from a temporary file to the JMS queue. In case of a lost connection, it tries to reconnect and resend it.

Action: Check that the JMS messaging server of DirX Audit is started. Check the address and the other configuration options. Correct the configuration if necessary.

16.5.20. Logging Messages (JINI)

This topic describes the logging messages returned by the DirX Identity JINI infrastructure.

WAR(JINI001): Couldn't initialize JMX support for '*string1*'.

Explanation: JINI infrastructure won't be manageable via JMX due to initialization error.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI002): Unexpected problems encountered (cause: *string1*).

Explanation: Serious problem, probably limited work capability.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI003): Unexpected problems encountered (cause: *string1*).

Explanation: Recoverable problem, components still working.

Action: Inform DirX Identity technical support about the problem.

INF(JINI004): Client died (cause: *string1*). Going to cleanup data structures.

Explanation: A remote client has died. Doing housekeeping, cleaning up data structures.

Action: None.

WAR(JINI005): lookup *string1* failed (cause: *string2*), trying Moniker.alias '*string3*' now ...

Explanation: Couldn't locate RMI-registry, trying fallback now.

Action: Check network infrastructure, inform DirX Identity technical support about the problem.

WAR(JINI006): Failed to execute command '*string1*' (cause: *string2*).

Explanation: Server was started in batchmode (runServer.bat), command entered by user (console) caused a problem.

Action: None.

INF(JINI007): Lease with non-infinite expiration defined '*string1*'.

Explanation: A lease with non-infinite expiration has been created.

Action: None.

ERR(JINI050): Failed to write entry *string1* to the JavaSpace (cause: *string2*).

Explanation: See error message (cause) for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI051): Failed while reading entry *string1* from JavaSpace due to a problem in the underlying repository (cause: *string2*).

Explanation: See error message (cause) for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI052): Failed while taking entry *string1* from JavaSpace due to a problem in the underlying repository (cause: *string2*).

Explanation: See error message (cause) for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI053): Promotion of participant *string1* to transaction *string2* failed (cause: *string3*).

Explanation: Transaction probably corrupt, see error message (cause) for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI054): Failed to abort participant *string1* because the deletion in the underlying repository reported an error (cause: *string2*).

Explanation: Transaction probably corrupt, see error message (cause) for further information.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI055): Create channel '*string1*' encountered problems while checking the existence of the channel (cause: *string2*), but continues operation.

Explanation: Non-critical error while creating a channel.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI056): Destroy channel '*string1*' encountered problems while checking the existence of the channel (cause: *string2*), but continues operation.

Explanation: Non-critical error while destroying a channel.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI057): Inconsistent channel *string1*, head=*string2*, tail=*string3*.

Explanation: JavaSpace corrupted.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI058): Channel *string1*: wait for ACK *string2* failed, timeout was *string3*ms.

Explanation: Producer wrote entry to synchronous channel. Received timeout while waiting for acknowledgment.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI060): ChannelListener for channel *string1* terminated unexpectedly (cause: *string2*).

Explanation: JavaSpace will probably get congested because there is no channel-consumer.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI061): *string1* is no LogRecord.

Explanation: A Channel-Consumer which processes log-messages read a channel-entry which isn't a LogRecord.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI062): ChannelHandler failed to publish message *string1* to queue *string2* (cause: *string3*).

Explanation: See error message (cause) for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI063): ChannelHandler for channel *string1* terminated unexpectedly (cause: *string2*).

Explanation: JavaSpace will probably get congested because there is no channel-consumer.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI064): ChannelHandler failed to publish *string1* message(s) to channel *string2* (cause: *string3*), sleeping *string4* ms before next retry.

Explanation: See error message (cause) for further information.

Action: Inform DirX Identity technical support about the problem.

INF(JINI065): JavaSpace: At least one segment is full - waiting *string1*s for free capacity before continuing 'add' operation.

Explanation: Too much JavaSpace entries, temporarily suspending operation.

Action: Inform DirX Identity technical support about the problem.

INF(JINI066): Free capacity available now, continuing 'add' operation.

Explanation: Continuing previously suspended 'add' operation.

Action: None.

INF(JINI067): JavaSpace restored *string1* entries from persistent storage.

Explanation: JavaSpace is currently starting up, recovered previously stored entries.

Action: None.

WAR(JINI068): JavaSpace: Remote client died, going to remove Listener (cause: *string1*).

Explanation: A client with a pending operation has died.

Action: None.

WAR(JINI069): JavaSpace: unusable IndexedEntry *string1* (cause: *string2*).

Explanation: Internal error, entry won't be indexed. This will degrade performance of retrieval operations.

Action: None.

WAR(JINI070): JavaSpace: Restoring TAKE of *string1* failed, no entry found.

Explanation: Internal error.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI071): JavaSpace: Restoring READ of *string1* failed, no entry found.

Explanation: Internal error.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI072): Failed to destroy channel '*string1*' because it doesn't exist.

Explanation: A channel may only be deleted if it previously existed.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI073): Channel *string1*: event '*string2*'.

Explanation: Handled event when the error occurred.

Action: None.

ERR(JINI080): Internal error in repository: operation *string1* failed. Identifier *string2* not attached to transaction.

Explanation: Entry referred by 'identifier' should be a member of a transaction.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI081): Illegal record *string1* found while loading segment#*string2* of repository *string3* from file *string4*.

Explanation: Internal error, segment corrupt.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI082): Failed to read record at position *string1* from segment#*string2* of repository *string3* (file is *string4*).

Explanation: Internal error, segment corrupt.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI083): Couldn’t create iterator for segment#*string1* of repository *string2*.

Explanation: Internal error, segment corrupt. Failed to iterate thru segment.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI084): Internal error: 'remove' operation for segment#*string1* of repository *string2* isn’t implemented and must NOT be called!

Explanation: Internal coding error.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI085): Iterator for repository *string1* failed to remove entry identified by *string2* (cause: *string3*).

Explanation: Internal error, see detailed error message for further information.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI086): Removing segment *string1*. It is inconsistent due to a crashed purge.

Explanation: Internal error, segment corrupt.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI087): Couldn’t delete segment '*string1*' (file is *string2*).

Explanation: File might be blocked by another user.

Action: Inform DirX Identity technical support about the problem.

INF(JINI088): Repository *string1*: open done, *string2* segments recovered.

Explanation: Repository is starting up, recovered entries from persistent storage (filesystem).

Action: None.

WAR(JINI089): Repository *string1*: Couldn’t rename file *string2* to *string3*.

Explanation: As a consequence, segment isn’t cleaned up.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI090): Repository *string1*: Couldn’t delete file *string1*.

Explanation: Segments are periodically reorganized. Failed to delete unneeded reorganized file.

Action: Remove file manually, inform DirX Identity technical support about the problem.

WAR(JINI091): Repository *string1*: Tried to delete non existing entry #*string2* while

recovering segment #*string3*.

Explanation: Internal error.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI092): Repository *string1*: Couldn’t update lease for entry #*string2* of segment #*string3* - entry doesn’t exist.

Explanation: Internal error, maybe timeout occurred.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI093): Repository *string1*: Couldn’t update lease for locator identified by *string2*.

Explanation: Internal error, maybe timeout occurred.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI094): repository *string1* doesn’t support purge().

Explanation: Internal error, this repository doesn’t support the 'purge' operation.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI095): Failed to load segment '*string1*' at position *string2* (file length=*string3*): *string4*. Trying to truncate file at given position.

Explanation: Server was unable to load the given segment in total. Repository content is partly corrupted and not usable. If truncation succeeds then that part up to given position is recovered. This could happen if a former run was unexpectedly stopped.

Action: If this happens more often inform DirX Identity technical support about the problem.

INF(JINI096): Renamed corrupted repository *string1* to *string2* for offline analysis.

Explanation: A corrupted repository was detected and renamed for offline analysis.

Action: Depending on the type of the repository you might contact DirX Identity technical support about the issue.

ERR(JINI097): Renamed corrupted repository *string1* to *string2* for offline analysis failed!

Explanation: A corrupted repository was detected but could not renamed.

Action: Check for reasons that the renaming failed. Inform DirX Identity technical support about the problem.

WAR(JINI098): Repository *string1*: Couldn’t read element with index *string2*.

Explanation: Internal error, maybe repository is corrupted.

Action: If this happens more often inform DirX Identity technical support about the problem.

INF(JINI099): Loaded *string1* (*string2* ms).

Explanation: The given segment file was loaded.

Action: None.

ERR(JINI100): Failed to restore transactions for queue *string1* (cause: *string2*).

Explanation: Queue is currently recovering its state (prior to a system crash). Failed to restore a transaction.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI101): Internal error: Nested transactions (for queue *string1*) aren't supported yet

Explanation: Internal coding error.

Action: Inform DirX Identity technical support about the problem.

INF(JINI102): BatchQueue *string1*: more than *string2* entries (*string3*,*string4*), sleeping now to avoid congestion ...

Explanation: Queue temporarily suspended operation, too many entries.

Action: None.

ERR(JINI150): Internal transactional error while performing operation '*string1*' (cause: *string2*).

Explanation: Transaction corrupt, see detailed error cause for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI151): Encountered problems while restoring transaction participant from file *string1* (cause: *string2*).

Explanation: Transaction corrupt, IO-error.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI152): Successfully restored transaction participant, but failed to delete file *string1*.

Explanation: IO-error, file probably locked by another user or insufficient access rights.

Action: Remove file manually, inform DirX Identity technical support about the problem.

ERR(JINI153): Failed to commit transaction #*string1* (cause: *string2*).

Explanation: Transaction corrupt, see detailed error cause for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI154): Transaction #*string1* has been committed before.

Explanation: A transaction can only be committed once.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI155): Illegal crash count encountered while committing transaction #*string1*

(cause: *string2*).

Explanation: Transaction corrupt, see detailed error cause for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI156): Failed to commit transaction #*string1* because 'join' operation reported an error (cause: *string2*).

Explanation: Transaction participant failed to join transaction, see detailed error cause for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI157): Operation 'vote' in transaction #*string1* has been interrupted.

Explanation: Operation 'vote' detected an interrupt.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI158): No local txn-manager found, using remote one.

Explanation: If txn-manager and txn-clients reside in the same VM, local calls should be used in favor of remote calls. Local calls are magnitudes faster then remote calls.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI159): Unknown txn-manager *string1*, seems to be a 3rd-party txn manager.

Explanation: Future usage only, should never happen in this version.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI160): Restoring Transaction *string1*: Reversing member *string2*.

Explanation: Possible reasons: txn doesn’t exist any more (client doesn’t exist, Lease timed out etc).

Action: Inform DirX Identity technical support about the problem.

WAR(JINI161): Restoring transactional file '*string1*' failed: *string2*.

Explanation: The transactional file could not be restored. File will be renamed.

Action: This could happen in case of a full device. Otherwise inform DirX Identity technical support about the problem.

WAR(JINI162): Renaming corrupt transactional file '*string1*' to '*string2*' failed.

Explanation: The transactional file could not be restored. File could not be renamed.

Action: Check the device. Otherwise inform DirX Identity technical support about the problem.

ERR(JINI163): Failed to commit in repository '*string1*' operation for id '*string2*'.

Explanation: Committing a change couldn’t be done.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI200): TransactionManager couldn't set new state of txn #*string1* in underlying repository (cause: *string2*).

Explanation: See detailed error cause for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI201): TransactionManager failed to create a nested transaction of parent txn #*string1* because join of txn #*string2* reported an illegal crashcount (cause: *string3*).

Explanation: See detailed error cause for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(JINI202): RemoteTxnManagerImpl: Internal error, more than one instance.

Explanation: Only one transaction manager may exist.

Action: Inform DirX Identity technical support about the problem.

INF(JINI203): different crash counts - aborting txn #*string1*.

Explanation: Corrupt transaction.

Action: None.

INF(JINI204): reversing txn #*string1* since a client died.

Explanation: Owner of a transaction died. TxnManager is going to reverse transaction.

Action: None.

WAR(JINI250): LeaseManager failed to cleanup leases (cause: *string1*).

Explanation: LeaseManager periodically checks leases. Encountered a problem while cleaning up, see error cause for further information.

Action: Inform DirX Identity technical support about the problem.

WAR(JINI300): Repair channel *string1*: Wrong number of entries! read=*string2*, tail=*string3*, head=*string4*, tail-head=*string5*.

Explanation: Inconsistent Head or Tail entry - doesn't match number of entries.

Action: None.

WAR(JINI301): Repair channel *string1*: Failed to read Tail.

Explanation: Could not read Tail entry - a new one will be created.

Action: None.

WAR(JINI302): Repair channel *string1*: Failed to read Head.

Explanation: Could not read Head entry - a new one will be created.

Action: None.

WAR(JINI303): Repair channel *string1*: Element #*string2* has position=*string3*, should be *string4*.

Explanation: Corrected a corrupt element index.

Action: None.

WAR(JINI304): Repair channel *string1*: Element #*string2* is flagged as EOF, bug EOF has already been set before.

Explanation: Duplicate EOF flags. Only the last entry may be marked as EOF.

Action: None.

WAR(JINI305): Repair channel *string1*: Setting EOF flag at (last) element #*string2*.

Explanation: Added missing EOF-flag.

Action: None.

WAR(JINI306): Repair channel *string1*: Removing EOF flag from element #*string2* since it is not the last one.

Explanation: Removed EOF-flag.

Action: None.

WAR(JINI307): Repair channel *string1*: Setting EOF flag in Tail.

Explanation: Corrupt tail - added EOF flag.

Action: None.

WAR(JINI308): Repair channel *string1*: Removing EOF flag from Tail.

Explanation: Corrupt tail - removed EOF flag.

Action: None.

WAR(JINI309): Repair channel *string1*: Number of repaired inconsistencies: *string2*.

Explanation: Shows summary of repair operation.

Action: None.

INF(JINI310): Backup for repository *string1* is due in *string2* (hh:mm:ss).

Explanation: The backup will start (again) in the given time interval.

Action: None.

INF(JINI311): Backup for repository *string1*: Acquiring exclusive lock ....

Explanation: The backup thread starts for the given repository.

Action: None.

INF(JINI312): Backup for repository *string1* finished in *string2* ms.

Explanation: The backup thread has finished in the given time.

Action: None.

WAR(JINI313): Getting entry count: repository *string1* is not opened.

Explanation: Asking for entry count, but repository is not opened.

Action: None.

ERR(JINI314): Deleting events from DB table failed: *string1*.

Explanation: Deleting of records from DB repository failed.

Action: None.

ERR(JINI315): Reading expired events from DB table: *string1*.

Explanation: Reading of expired records from DB repository failed.

Action: None.

ERR(JINI316): Dropping of DB table failed: *string1*.

Explanation: Dropping of DB repository table failed.

Action: None.

INF(JINI317): Creating of DB table failed: *string1*.

Explanation: Creating of DB repository table failed.

Action: None.

INF(JINI318): Starting migration to DB.

Explanation: Starting migration from FS repository to DB repository.

Action: None.

INF(JINI319): Migration to DB finished in *string1* s.

Explanation: Migration from FS repository to DB repository finished.

Action: None.

INF(JINI320): Executing query to repository *string1*: *string2*.

Explanation: SQL query to DB repository.

Action: None.

INF(JINI321): Executing update to repository *string1*: *string2*.

Explanation: SQL update to DB repository.

Action: None.

INF(JINI322): Opening repository *string1*.

Explanation: Opening repository.

Action: None.

INF(JINI323): Repository *string1* opened in *string2* s. Used driver type *string3*.

Explanation: Repository opened.

Action: None.

INF(JINI324): Closing repository *string1*.

Explanation: Closing repository.

Action: None.

INF(JINI325): Repository *string1* closed in *string2* s.

Explanation: Repository closed.

Action: None.

INF(JINI326): *string1* repository is moved to '*string2*'.

Explanation: Repository is moved to the new server-specific folder.

Action: None.

WAR(JINI327): *string1* repository: Could not create new folder structure '*string2*'.

Explanation: New server-specific folder structure could not be created. Repository is therefore not working.

Action: Check and repair why the folder structure could not created. Restart server.

WAR(JINI328): *string1* repository: Could not copy file '*string2*': *string3*.

Explanation: New server-specific folder structure could not be created. Repository is therefore not working.

Action: Check and repair why the folder structure could not created. Restart server.

WAR(JINI329): *string1* repository: Could not delete old file '*string2*': *string3*.

Explanation: New server-specific folder structure for the repository could not be moved completely.

Action: Check file why it cannot be deleted after copying.

WAR(JINI330): SQLException during executing update to repository *string1*: statement: '*string2*' byte size: *string3*.

Explanation: SQL update error to DB repository.

Action: None.

WAR(JINI331): SQLException during closing repository *string1*.

Explanation: Closing repository caused some issue.

Action: None.

INF(JINI400): Purging file '*string1*' with *string2* old entries (new *string3*) and *string4* old records (new *string5*) successfully done (*string6* ms).

Explanation: Reorganizing repository segment file was processed.

Action: None.

WAR(JINI401): Deleting empty segment file *string1* failed.

Explanation: An empty segment file could't be deleted.

Action: Delete file manually.

WAR(JINI402): Cannot read unique ID *string1* in *string2* repository.

Explanation: Wrong unique identifier.

Action: None.

WAR(JINI999): Lookup of '*string1*' in RMI-registry failed (cause: *string2*).

Explanation: Couldn't find RMI-registry needed to connect to (remote) javaspace, repository or transaction manager.

Action: Check network infrastructure, inform DirX Identity technical support about the problem.

16.5.21. Logging Messages (JINIAPI)

This topic describes the logging messages returned by the DirX Identity JINI infrastructure API.

WAR(JINIAPI000): Couldn't initialize JMX support for '*string1*'

Explanation: Failed to initialize the Java Management Extension for the denoted component.

Action: Inform DirX Identity technical support about the problem.

WAR(JINIAPI001): Couldn't unregister '*string1*' from JMX registry

Explanation: The Java Management Extension reported an error while unregistering a component.

Action: Inform DirX Identity technical support about the problem.

WAR(JINIAPI003): Unexpected problems encountered (cause: *string1*)

Explanation: An internal error occurred, check the message text for details.

Action: Inform DirX Identity technical support about the problem.

WAR(JINIAPI004): Low memory! committed=*string1*kb, max=*string2*kb, used=*string3*kb, sleeping now ...

Explanation: Available memory has fallen below configured high water threshold. Adaptors are blocked until low water mark is reached.

Action: None.

WAR(JINIAPI005): High water mark value *string1* reached in *string2*. Adaptor is going to state 'waiting' (blocking now!).

Explanation: The high water mark has been reached for given resource. No events are read until low water threshold is reached.

Action: If possible add further, appropriate threads to scale processing of events.

ERR(JINIAPI020): Failed to deserialize representation of TransactionManager from *string1* (cause: *string2*)

Explanation: Internal error while deserialize the representation of TransactionManager. Check the message text for details.

Action: Inform DirX Identity technical support about the problem.

INF(JINIAPI030): Sufficient memory now, continuing work.

Explanation: There is enough memory (above configured high-water threshold) to proceed.

Action: None.

INF(JINIAPI031): Low water mark value *string1* reached in *string2*. Adaptor is going to state 'listening'.

Explanation: The low water mark has been reached for given resource. Events are read again.

Action: None.

WAR(JINIAPI646): Instance tracker cannot keep trace of repository operations: Identifier *string1*, type *string2* (exception *string3*)

Explanation: An exception occurred while getting notifications of repository operations.

Action: Refresh the view.

16.5.22. Logging Messages (JOIN)

This topic describes the logging messages returned by the Join engine.

INF(JOIN201): User "*string1*": Password changed.

Explanation: User password has successfully been changed.

Action: None.

INF(JOIN202): Update of target system account successfully initiated; Identifier: *string1*.

Explanation: Update of account in the target system has successfully been initiated.

Action: None.

INF(JOIN203): User DN: "*string1*"; Account: "*string2*": Update of target system account successfully initiated.

Explanation: Update of account in the target system has successfully been initiated.

Action: None.

INF(JOIN204): Account "*string1*" successfully updated.

Explanation: Target system account has successfully been updated.

Action: None.

INF(JOIN205): Target system account successfully updated; User DN: *string1*; Account: *string2*.

Explanation: Target system account has successfully been updated.

Action: None.

INF(JOIN206): User "*string1*": Password already exists.

Explanation: User password already exists.

Action: None.

INF(JOIN207): Object "*string1*" successfully created.

Explanation: The object has successfully been created.

Action: None.

INF(JOIN208): Object "*string1*" successfully modified.

Explanation: The object has successfully been updated.

Action: None.

INF(JOIN209): Object "*string1*" successfully deleted.

Explanation: The object has successfully been deleted.

Action: None.

INF(JOIN210): ****** Synchronization 'Identity Domain --> TS' started. Source channel name: '*string1*', target channel name: '*string2*'.

Explanation: The synchronization from Identity domain to target system has started.

Action: None.

INF(JOIN211): ****** Synchronization 'Identity Domain --> TS' successfully terminated."

Explanation: The synchronization from Identity domain to target system has successfully terminated.

Action: None.

INF(JOIN212): ****** Synchronization 'TS --> Identity Domain' started. Source channel name: '*string1*', target channel name: '*string2*'.

Explanation: The synchronization from target system to Identity domain has started.

Action: None.

INF(JOIN213): ****** Synchronization 'TS --> Identity Domain' successfully terminated."

Explanation: The synchronization from target system to Identity domain has successfully terminated.

Action: None.

INF(JOIN214): ****** Validation for source channel '*string1*', target channel '*string2*' started.

Explanation: The validation workflow has started using the given channel definition.

Action: None.

INF(JOIN215): ****** Validation for source channel '*string1*', target channel '*string2*' successfully terminated.

Explanation: The validation workflow has successfully terminated.

Action: None.

INF(JOIN216): ****** Update of cross references started. Group name: '*string1*'.

Explanation: Updates of membership started.

Action: None.

INF(JOIN217): ****** Update of cross references terminated. Group name: '*string1*'.

Explanation: Updates of membership terminated.

Action: None.

INF(JOIN218): Object '*string1*' doesn't exist.

Explanation: Object doesn't exist in the LDAP directory. That situation may be ok, if the object should later on be added. It's not ok, if the object really should exist (in case later on a DELETE or MODIFY operation should be executed.

Action: None.

INF(JOIN220): There are no audit policies present or enabled for domain root '*string1*' and

audit object type '*string2*'. No audit records related to that audit policy are generated.

Explanation: No audit policies are enabled or present or audit object type doesn't match.

Action: None.

INF(JOIN221): Auditing is not enabled at the domain root '*string1*'. No audit records are generated.

Explanation: The attribute "dxrWriteHistory" is not set or set to FALSE (at the domain root).

Action: Check the attribute "dxrWriteHistory" at the domain root.

INF(JOIN222): The object description name (used for evaluation of audit and chage event policy) is missing in the channel "*string1*". No audit records are generated.

Explanation: The field "Object Description Name" is not set in the channel.

Action: Assign a correct value for object description name in the channel definition.

INF(JOIN223): No matching channel found that matches base object and filter condition from export section; incoming identifier: '*string1*'; attribute list: *string2*.

Explanation: There is no channel available that matches base object and/or filter from export section.

Action: Check the configuration.

INF(JOIN224): Primary attribute '*string1*' not present in the entry '*string2*'.

Explanation: An attribute used to find an entry via the secondary channel is not present.

Action: Check the database.

INF(JOIN225): Secondary attribute '*string1*' not present in the entry '*string2*'.

Explanation: An attribute used to find an entry via the secondary channel is not present.

Action: Check the database.

INF(JOIN226): Creation request workflow for object "*string1*" successfully instantiated.

Explanation: The request workflow has successfully been instantiated.

Action: None.

INF(JOIN227): Modification request workflow for object "*string1*" successfully instantiated.

Explanation: The request workflow has successfully been instantiated.

Action: None.

INF(JOIN228): Deletion request workflow for object "*string1*" successfully instantiated.

Explanation: The request workflow has successfully been instantiated.

Action: None.

INF(JOIN229): No notification about password changes is sent as there are no recipients.

Explanation: The notification objects as neither defined recipients nor copy recipients nor blind copy recipients.

Action: Check the recipients defined in the notification object of the workflow.

INF(JOIN230): Incoming password update event for account '*string1*' (target system '*string2*' is ignored as account is no primary account.

Explanation: Password updates are only done for primary accounts.

Action: None.

INF(JOIN231): No password update event generated for account '*string1*' as account is no primary account.

Explanation: Password updates are only done for primary accounts.

Action: None.

INF(JOIN232): Cluster workflow detects a change in the target system and therefore switches to the following target system: Type=*string1*,Cluster=*string2*,DomainName=*string3*,Address=*string4*,Port=*string5*.

Explanation: Provides information when cluster workflow switches to a different target system.

Action: None.

INF(JOIN233): Delta state object '*string1*' ('dxrLastDeltaValue=*string2*') was successfully created.

Explanation: The object that holds the highest value for delta synchronisation was successfully created.

Action: None.

INF(JOIN234): creation of superior node '*string1*' for delta state object '*string2*' succeeded.

Explanation: The superior object for delta state object was successfully created.

Action: None.

INF(JOIN235): Modification of delta state object '*string1*' ('dxrLastDeltaValue=*string2*') succeeded.

Explanation: The delta state object was successfully modified.

Action: None.

INF(JOIN236): Retrieval of delta state object '*string1*' succeeded; 'dxrLastDeltaValue=*string2*'.

Explanation: The delta state object was successfully readd.

Action: None.

INF(JOIN237): Delta state object '*string1*' doesn't exists; full synchronisation started.

Explanation: The delta state object is not present; therefore a full synchronisation is done the very first time.

Action: None.

INF(JOIN238): Ignoring additional entries in search result (class name: *string1*).

Explanation: Search result contains other items than search result entries, too. Such entries (e.g. Size Limit Problem indicator) are ignored.

Action: None.

INF(JOIN239): The processing was interrupted when executing interface "*string1*" of userhook "*string2*".

Explanation: The userhook implementation stopped processing of the SPML request.

Action: None.

INF(JOIN240): No userLink attribute found for account *string1*.

Explanation: The given account has not assigned any user link attribute.

Action: Check the LDAP directory why there is no user link attribute. Keep in mind that anyway privileged accounts have no user link.

INF(JOIN241): ****** Synchronization 'TS --> Identity Domain' terminated with errors."

Explanation: The synchronization from target system to Identity domain has terminated, but there were some errors.

Action: Check the error messages.

INF(JOIN242): No userLink attribute found for account *string1*. Password will be ignored.

Explanation: The given account has not assigned any user link attribute (e.g. if the account is still in state IMPORTED). The password event will not be processed and it will not be stored in DLQ (due to domain flag set).

Action: Check the LDAP directory why there is no user link attribute. Keep in mind that anyway privileged accounts or accounts in state IMPORTED have no user link.

INF(JOIN243): Switching to TS: *string1*.

Explanation: For cluster workflow an additional message is written to the monitor entry in order to identity the operation executed for the specific target systems..

Action: None.

INF(JOIN244): ****** Update of cross references started. Account name: *string1*.

Explanation: Updates of membership started.

Action: None.

INF(JOIN245): ****** Update of cross references terminated. Account name: *string1*.

Explanation: Updates of membership terminated.

Action: None.

INF(JOIN246): Running workflow without target system data.

Explanation: No target system data available, but still continuing.

Action: None.

INF(JOIN247): Object '*string1*' is not synchronized with state '*string2*'. Valid values are ENABLED, DISABLED, DELETED.

Explanation: Objects are only synchronized if their 'dxrState' is ENABLED or DISABLED or DELETED.

Action: None.

WAR(JOIN401): Update for '*string1*' currently ignored as a pending request has been found.

Explanation: A previous move/rename operation to Lotus Notes has not yet been terminated successfully in the Lotus Notes system. No second move/rename operation is executed until the previous one terminates.

Action: Check that a previous move/rename operation to Lotus Notes terminates successfully and the pending request gets removed.

WAR(JOIN402): Move operation for '*string1*' not propagated as first the rename operation has to terminate successfully.

Explanation: A rename and move operation can’t be propagated the same time. Therefore the move operation has been delayed and will only be processed in next synchronization cycle.

Action: Check that the rename operation terminates successfully.

WAR(JOIN450): Could not get the state of the account "*string1*", omitting.

Explanation: The join engine could not get the value of the attribute dxrState of the given account.

Action: Check consistence of the given account.

WAR(JOIN451): Could not get the type of the TS for the account "*string1*", omitting.

Explanation: The join engine could not get the value of the attribute dxrType of the TS for the given account.

Action: Check existence of the missing attribute for the related TS.

WAR(JOIN452): Could not get the information about preventing password synchronization of the TS for the account "*string1*", omitting.

Explanation: The join engine could not get the value of the attribute dxrPreventPwdSync of the TS for the given account.

Action: Check existence of the missing attribute for the related TS.

WAR(JOIN453): Could not get the name of the external application for the account "*string1*", omitting.

Explanation: The join engine could not get the value of the attribute dxrTSDomainName of the TS for the given account.

Action: Check existence of the missing attribute for the related TS.

WAR(JOIN454): Could not get the login for the external application for the account "*string1*", omitting.

Explanation: The join engine could not get the value of the attribute dxrName of the given account.

Action: Check existence of the missing attribute for the related account.

WAR(JOIN461): Error while reading the audit flag at the domain root '*string1*'; error: *string2*. No audit records are generated.

Explanation: There is an error while reading the attribute "dxrWriteHistory".

Action: Check the error message.

WAR(JOIN462): Error while reading the audit policies for domin root '*string1*'; error: *string2*. No audit records are generated.

Explanation: There is an error while reading the audit policies.

Action: Check the error message.

WAR(JOIN465): More than one audit policy found for audit object type "*string1*"; using audit policy definition from object "*string2*".

Explanation: The audit object type is not unique. The join engine uses the first definition that has been found.

Action: Provide just one audit policy.

WAR(JOIN466): Searching the identifier attributes for object "*string1*" (for audit purposes) failed. Error: *string2*.

Explanation: Searching identifier attributes failed. The error will be ignored and audit records are written without these attributes.

Action: Check the error message.

WAR(JOIN467): Searching change event policies in domain "*string1*" failed. Error: *string2*. (Error will be ignored, but no event messages will be generated.)

Explanation: Searching change event policies failed. The error will be ignored and no JMS event messages will be generated.

Action: Check the error message.

WAR(JOIN468): The component "cn" can’t be extracted from the DN '*string1*'.

Explanation: For audit records the field 'Name' needs to be set in the identifier attributes. That field is derived from the CN of the bind DN, but is empty.

Action: Check the bind DN.

WAR(JOIN469): Searching for attribute 'dxrTopicIncludesDomain' in domain '*string1*' failed. Error: *string2*. (Error will be ignored, but no event messages will be generated.)

Explanation: Searching the attribute failed. The error will be ignored, but no JMS change messages will be generated.

Action: Check the error message.

WAR(JOIN470): Join engine terminates due to Interrupt exception (timeout).

Explanation: The join engine thread will be aborted by server due to timeout.

Action: Increase time out for realtime workflow / activities.

WAR(JOIN471): Account "*string1*" doesn’t exist for target system "*string2*".

Explanation: The given target system doesn’t hold the account name.

Action: Check why the target system doesn’t hold the account; furthermore check why you received a password update request for a non-existing account.

WAR(JOIN472): Workflow finishes with WARNING as there was *string1* update operation that failed.

Explanation: An update operation failed with errors and therefore the workflow finishes with state WARNING.

Action: Check the log file for erroneous update operations.

WAR(JOIN473): Workflow finishes with WARNING as there were *string1* update operations that failed.

Explanation: Many update operations failed with errors and therefore the workflow finishes with WARNING.

Action: Check the log file for erroneous update operations.

WAR(JOIN474): The class '*string2*' returned a '*string1*' when calling its close method: *string3*.

Explanation: In the close method an error was encountered while releasing the resources.

Action: Check the log messages.

WAR(JOIN475): Workflow finishes with ERROR.

Explanation: A serious error was detected while running the workflow.

Action: Check the log file for related error messages.

WAR(JOIN476): Attribute list for membership calculation may be incomplete as evaluation of object '*string1*' didn't provide a value.

Explanation: In cross membership scenario the membership attribute could not be setup correctly.

Action: Check the attribute list of the given object.

ERR(JOIN501): Searching account; Exception: *string1*.

Explanation: Searching the account name in the LDAP directory failed.

Action: Check the given error reason.

ERR(JOIN502): Searching for additional accounts for user *string1* failed.

Explanation: Searching all accounts of the given user failed.

Action: Check the error reason of a previously called search operation.

ERR(JOIN504): User "*string1*": Password change failed; Exception: *string2*.

Explanation: Password update for the given user failed.

Action: Check the given error reason.

ERR(JOIN505): Domain is not defined.

Explanation: There is domain name defined in the job configuration.

Action: Check the job configuration.

ERR(JOIN506): Topic prefix is not defined.

Explanation: There is no topic prefix defined in the job configuration.

Action: Check the job configuration.

ERR(JOIN507): No matching target system found: Type=*string1*; Domain=*string2*; Cluster=*string3*.

Explanation: No target system has been found for the given target system parameters.

Action: Check the target systems parameters; if ok then check why you received a password update request for an unknown target system.

ERR(JOIN508): Multiple matching target systems found: Type=*string1*; Domain=*string2*; Cluster=*string3*; Matching target systems: *string4*.

Explanation: Several target systems match the subset of given criteria.

Action: Check the target systems; if ok then check why the password update request only contains a subset of matching attributes.

ERR(JOIN509): No target system found.

Explanation: No target systems found in the LDAP directory.

Action: Check the LDAP directory.

ERR(JOIN510): Error while searching target systems.

Explanation: Searching all the target systems failed.

Action: Check the error reason of a previously called search operation.

ERR(JOIN511): No search result available when searching target systems.

Explanation: No search result received when looking for all the target systems.

Action: Check the LDAP directory.

ERR(JOIN512): Processing terminated as no target systems are available.

Explanation: As no target systems are available no requests (password updates) can be processed.

Action: Check the LDAP directory.

ERR(JOIN513): Target system type is missing.

Explanation: The password update request doesn't hold the target system type component.

Action: Check why you received an incomplete password update request.

ERR(JOIN514): Search for account "*string1*" for target system "*string2*" failed; error: *string3*

Explanation: Searching the account failed.

Action: Check the error reason of a previously called search operation.

ERR(JOIN515): User DN: "*string1*"; Account: "*string2*"; Error while initiating update of target system account: "*string3*".

Explanation: Starting an additional job for updating the account in the target system fails.

Action: Check the target system job and the target system itself.

ERR(JOIN517): Multiple matches found for account "*string1*" in target system "*string2*".

Explanation: The account name is not unique within the target system.

Action: Check the LDAP directory why there are non unique accounts.

ERR(JOIN518): Account "*string1*": Update failed; Error: "*string2*".

Explanation: Updating the account in the target system failed.

Action: Check the error reason.

ERR(JOIN519): Error while initiating updates of target system accounts.

Explanation: AT least one update of a target system failed.

Action: Check the target system connectors.

ERR(JOIN520): Connector *string1* doesn't exist.

Explanation: The given connector name has not been found in the workflow configuration.

Action: Check the workflow configuration.

ERR(JOIN521): Error while running Java script *string1*: *string2*.

Explanation: An error occurred while running a Java script.

Action: Check the given error reason.

ERR(JOIN522): Error while running Java script.

Explanation: An error occurred while running a Java script.

Action: Check the given error reason.

ERR(JOIN523): Processing of error request failed.

Explanation: An error occurred while running the error activity.

Action: Check the error activity.

ERR(JOIN524): Processing of error request failed; Exception: *string1*.

Explanation: An error occurred while running the error activity.

Action: Check the error reason.

ERR(JOIN525): Illegal target system state: *string1*; Account: *string2*.

Explanation: The target system state of the account forbids updating the account.

Action: Check why the target system state is not ok.

ERR(JOIN526): Missing parameters (e.g. "to") for mail connector request.

Explanation: An invalid request for the mail connector has been generated by the Java script.

Action: Check the Java script.

ERR(JOIN527): Multiple matching target systems found (exact match): Type=*string1*; Domain=*string2*; Cluster=*string3*; Matching target systems: *string4*.

Explanation: Multiple target systems have been found when matching all the given parameters that came along with the password update request.

Action: Check the attributes of the target systems.

ERR(JOIN528): Searching object "*string1*" (with filter "*string2*") failed; error: *string3*

Explanation: Searching an object in the LDAP directory failed.

Action: Check the error reason.

ERR(JOIN529): Error writing temporary file "*string1*"; error: *string2*

Explanation: An error occurred when writing the attachments when sending mails.

Action: Check the error reason.

ERR(JOIN530): Temporary file "*string1*" not used in mail attachment.

Explanation: An error occurred while generating a mail attachment; therefore the mail is sent without attachment.

Action: Check the error reason.

ERR(JOIN531): Error while sending result using *string1*: *string2*.

Explanation: AN error occurred while using the given connector.

Action: Check the error reason.

ERR(JOIN532): Multiple matches found for source object '*string1*'.

Explanation: The search criteria are not unique.

Action: Check the LDAP directory and/or the parameters of the search operation.

ERR(JOIN533): Object "*string1*" doesn't exist.

Explanation: Object has not been found in the connector database.

Action: Check the database and/or SPML search operation.

ERR(JOIN534): No value available for attribute *string1*; object name *string2*.

Explanation: An attribute value that should be available is not present.

Action: Check the objects' attributes.

ERR(JOIN535): Creation of object "*string1*" failed; error: *string2*

Explanation: Creation of object has not been successful.

Action: Check the error reason.

ERR(JOIN536): Update of object "*string1*" failed; error: *string2*

Explanation: Modification of an object has not been successful.

Action: Check the error reason.

ERR(JOIN537): Deletion of object "*string1*" failed; error: *string2*

Explanation: Deletion of an object has not been successful.

Action: Check the error reason.

ERR(JOIN538): No matching target object found for source object "*string1*" while joining.

Explanation: Object has not been found in the connector database.

Action: Check the database and/or SPML search operation.

ERR(JOIN539): Error while creating SPML request file '*string1*' for simulation mode; error: *string2*

Explanation: An error has occurred while creating the SPML request file used if simulation mode is switched on.

Action: Check the reason.

ERR(JOIN540): Error while creating an SPML marshaller; error: *string1*

Explanation: An error has occurred while creating an SPML marshaller for simulation mode.

Action: Check the reason.

ERR(JOIN541): Error while marshalling SPML request into file; error: *string1*

Explanation: An error has occurred while marshalling an SPML request into file.

Action: Check the reason.

ERR(JOIN542): Invalid simple expression '*string1*'; error at '*string2*'.

Explanation: The given simple expression has an invalid syntax.

Action: Check the simple expression at the given error position.

ERR(JOIN543): ****** Synchronization 'Identity Domain --> TS' failed; error: "*string1*".

Explanation: Said synchronization terminated with errors.

Action: Check the error message and the trace file for problems.

ERR(JOIN544): ****** Synchronization 'TS --> Identity Domain' failed; error: "*string1*".

Explanation: Said synchronization terminated with errors.

Action: Check the error message and the trace file for problems.

ERR(JOIN545): ****** Validation for source channel '*string1*', target channel '*string2*' failed; error: "*string3*".

Explanation: Validation terminated with errors.

Action: Check the error message and the trace file for problems.

ERR(JOIN546): Connector *string1* doesn't implement SPMLv1 requests.

Explanation: The given connector needs to implement DxmRequestor interface with methods addRequest(), modifyRequest(), etc.

Action: Take a connector class that implements this SPMLv1 requests!

ERR(JOIN547): No value for identifier available while evaluating search base expression '*string1*'.

Explanation: No identifier could be generated while evaluating the simple expression in the search base definition.

Action: Check the simple expression.

ERR(JOIN548): Target variables are only allowed in simple expressions while performing join operation; invalid simple expression '*string1*'.

Explanation: Target variables in simple expressions can only be listed in the join section. They are not allowed in the search base section of the export section or in the mapping section.

Action: Check the simple expression.

ERR(JOIN549): Executing the extended request failed; error: *string1*.

Explanation: An error occurred while executing an extended request.

Action: Check the error message.

ERR(JOIN550): The attribute mapping didn't set a proper request operation type.

Explanation: The request operation type is not one of add, modify, delete.

Action: Check the attribute mapping.

ERR(JOIN551): Invalid user link.

Explanation: The method for searching for Dashboard external applications was called with null or empty parameter (dxrUserLink).

Action: Check the mapping script for Dashboard password synchronization workflow.

ERR(JOIN552): Invalid token delimiter.

Explanation: The token delimiter for Dashboard password synchronization job is either null or empty.

Action: Check the configuration of the Dashboard password synchronization workflow.

ERR(JOIN553): Searching for external applications for user "*string1*" failed.

Explanation: Searching for external applications of the given user failed.

Action: Check the error reason of a previously called search operation.

ERR(JOIN554): Could not get the TS for the DN "*string1*", omitting.

Explanation: The join engine could not get the TS for the given DN.

Action: Check the validity of the given DN.

ERR(JOIN555): No target system found.

Explanation: The join engine could not find any target system in the LDAP directory.

Action: Check the LDAP directory, check if the 'domain' parameter is correctly set.

ERR(JOIN556): Error while searching target systems.

Explanation: Searching for all the target systems failed.

Action: Check the error reason of a previously called search operation.

ERR(JOIN557): No search result available when searching target systems.

Explanation: No search result received when looking for all the target systems.

Action: Check the LDAP directory, check if the 'domain' parameter is correctly set.

ERR(JOIN558): Processing terminated as no target systems are available.

Explanation: As no target systems are available no requests (password updates in external applications) can be processed.

Action: Check the LDAP directory, check if the 'domain' parameter is correctly set.

ERR(JOIN559): Invalid domain.

Explanation: The join engine for searching for Dashboard external applications was called with null or empty parameter (domain).

Action: Check the mapping script for Dashboard password synchronization workflow.

ERR(JOIN560): Invalid Dashboard target system DN.

Explanation: The join engine for searching for Dashboard external applications was called with null or empty parameter (dashboardTSDN).

Action: Check the mapping script for Dashboard password synchronization workflow.

ERR(JOIN561): Invalid context.

Explanation: The join engine for searching for Dashboard external applications was called with null or empty parameter (context).

Action: Check the mapping script for Dashboard password synchronization workflow.

ERR(JOIN562): Connector "*string1*" does not exist.

Explanation: The given connector name has not been found in the Dashboard workflow configuration.

Action: Check the Dashboard password synchronization workflow configuration.

ERR(JOIN563): Could not get the value of dxrOptions (authenticationType) for the Dashboard TS "*string1*".

Explanation: The join engine could not get the value of the attribute dxrOptions

(authenticationType) of the Dashboard TS.

Action: Check existence of the missing attribute for the related TS.

ERR(JOIN570): The user hook '*string1*' returned an error.

Explanation: The user hook didn’t return the result code 'success'.

Action: Check the user hook function.

ERR(JOIN571): Reading superior object '*string1*' failed; error: *string2*.

Explanation: Retrieval of a superior node failed.

Action: Check the error message.

ERR(JOIN572): Error while creating superior object of '*string1*' failed; error: *string2*.

Explanation: Creating a superior node failed.

Action: Check the error message.

ERR(JOIN573): Corresponding source object '*string1*' doesn’t exist.

Explanation: While creating a new superior object the corresponding object has not been found.

Action: Check the DIT.

ERR(JOIN574): Object '*string1*' is not a descendant of '*string2*'.

Explanation: The object in the DIT is not located in the same subtree has the export search base.

Action: Check the DIT and the identifier mapping.

ERR(JOIN575): Superior nodes of '*string1*' and '*string2*' differ.

Explanation: The superior nodes of source and target DN are different.

Action: Check the DIT and the identifier mapping.

ERR(JOIN576): Updating attribute '*string1*' in object '*string2*' failed; error: *string3*.

Explanation: Attribute modification failed.

Action: Check the error message..

ERR(JOIN577): More than one default profile exists.

Explanation: There are several profiles available in the database; all of them are marked as default profile.

Action: Check the database and administer only one default profile.

ERR(JOIN578): No default profile has been found.

Explanation: There is no default profile available in the database.

Action: Check the database and create a default profile.

ERR(JOIN579): No attribute type for mapping of memberships found.

Explanation: In case of cross memberships an attribute mapping for 'member' has not been found.

Action: Check the attribute mapping of the member channels.

ERR(JOIN580): Attribute mapping for target attribute '*string1*' must be a direct mapping.

Explanation: The attribute mapping for the given attribute type must be a direct mapping.

Action: Check the attribute mapping for the given attribute type.

ERR(JOIN581): Attribute mapping for target attribute '*string1*' not found.

Explanation: The attribute mapping for the given attribute type does not exist.

Action: Check the attribute mapping for the given attribute type.

ERR(JOIN582): Object '*string1*' doesn't have an attribute value for attribute '*string2*'.

Explanation: The given attribute type is missing.

Action: Check the account/group for existence of given attribute type.

ERR(JOIN583): Update of cross references failed. Group name: '*string1*'.

Explanation: Updates of membership terminated with error.

Action: Check the information from logfile

ERR(JOIN586): No target system available while processing request identifier '*string1*'; TS type: '*string2*'; TS cluster: '*string3*'; TS domain: '*string4*'.

Explanation: Using the information from the realtime event, no target system could be identified.

Action: Check the database.

ERR(JOIN587): The target system '*string1*' doesn't exist.

Explanation: The given target system is not available in the database.

Action: Check the database.

ERR(JOIN588): User "*string1*": Password change failed. LDAP error: "*string2*".

Explanation: Password update for the given user failed due to LDAP problems.

Action: Check the LDAP error message.

ERR(JOIN589): Reason for retry request: *string1*.

Explanation: An update request fails and requires a retry.

Action: Check the error message.

ERR(JOIN590): Password update request for object "*string1*" is only supported for MODIFY operation.

Explanation: A password update event doesn't result in a MODIFY operation.

Action: Check why any other update operation than MODIFY is generated.

ERR(JOIN591): Error while creating superior object of '*string1*' failed (reading the corresponding object failed); error: *string2*.

Explanation: For creation of a superior node the corresponding object either in the target system or the identity domain needs to be read. Reading that object failed.

Action: Check the error message.

ERR(JOIN592): Exception "*string1*" when executing interface "*string2*" of userhook "*string3*": *string4*.

Explanation: The user defined user hook returned an exception.

Action: Check the error message.

ERR(JOIN593): Instantiation of creation request workflow for object "*string1*" failed; error: *string2*

Explanation: Instantiation of request workflow has not been successful.

Action: Check the error reason.

ERR(JOIN594): Instantiation of modification request workflow for object "*string1*" failed; error: *string2*

Explanation: Instantiation of request workflow has not been successful.

Action: Check the error reason.

ERR(JOIN595): Instantiation of deletion request workflow for object "*string1*" failed; error: *string2*

Explanation: Instantiation of request workflow has not been successful.

Action: Check the error reason.

ERR(JOIN596): '*string1*' when running workflow; error: *string2*.

Explanation: A general problem has occurred.

Action: Check the error reason.

ERR(JOIN597): Resolving simple expression '*string1*' failed ; no value for '*string2*' variable: '*string3*'.

Explanation: The given variable could not be resolved.

Action: Check the variable expression. The name may be misspelled.

ERR(JOIN598): Error while reading domain flags (dxrSyncPrimaryAccountsOnly); error: '*string1*'.

Explanation: Problem when reading the attribute 'dxrSyncPrimaryAccountsOnly' from the domain.

Action: Check the error reason.

ERR(JOIN599): Error while creating delta state object '*string1*' ('dxrLastDeltaValue=*string2*'); error: *string3*.

Explanation: Creation of object that holds the highest value for delta synchronisation failed.

Action: Check the error reason.

ERR(JOIN600): Error while creating superior node '*string1*' for delta state object '*string2*'; error: *string3*.

Explanation: Creation of superior object failed.

Action: Check the error reason.

ERR(JOIN601): Error while modifying delta state object '*string1*' ('dxrLastDeltaValue= *string2*'); error: *string3*.

Explanation: Modification of object that holds the highest value for delta synchronisation failed.

Action: Check the error reason.

ERR(JOIN602): Exception received while updating delta state object '*string1*' ('dxrLastDeltaValue=*string2*'); error: *string3*.

Explanation: Update of object that holds the highest value for delta synchronisation failed.

Action: Check the error reason.

ERR(JOIN603): Exception received while reading delta state object '*string1*'; error: *string2*.

Explanation: Creation of object that holds the highest value for delta synchronisation failed.

Action: Check the error reason.

ERR(JOIN604): There is no delta configuration item in the workflow definition for channel '*string1*'.

Explanation: The XML configuration of the workflow doesn’t include a delta component.

Action: Check the workflow configuration for delta item..

ERR(JOIN605): For delta workflow, the target system DN could not be evaluated from export search base of channel '*string1*'.

Explanation: In order to decide where to store the delta state object the target system DN needs to be evaluated (from export search basye); but evaluation failed.

Action: Check the workflow configuration for export search base.

ERR(JOIN606): *string1* while mapping and creating/updating the entry '*string2*'.

Explanation: An exception was caught while mapping or creating/updating an entry.

Action: Check the error message.

ERR(JOIN607): Attribute list for membership calculation may be incomplete as evaluation of object '*string1*' resulted in a mapping exception: *string2*.

Explanation: In cross membership scenario the membership attribute could not be setup correctly.

Action: Check the error message and the attribute list of the given object.

ERR(JOIN608): ****** Update of cross references terminated with error: *string1*. Account name: *string2*.

Explanation: Updates of membership terminated with error.

Action: Check the error message.

ERR(JOIN609): Found ADD operation for primary channel and MODIFY operation for secondary channel. This is an illegal state, when referenceFromSecondary=true

Explanation: An error occurred while updating primary and secondary entries.

Action: Check the error message.

FTL(JOIN901): 'add' not implemented.

Explanation: The 'add' method is not supported by the connector.

Action: Check why you received an Add request.

FTL(JOIN902): Initialization of CryptSupport library failed.

Explanation: Loading the crypt support library (used for decrypting passwords) failed.

Action: Check the error reason.

FTL(JOIN903): Initialization of PwdManagement failed.

Explanation: Setup of the password management environment failed.

Action: Check the error reason.

FTL(JOIN904): 'modify' not implemented.

Explanation: The 'modify' method is not supported by the connector.

Action: Check why you received a Modify request.

FTL(JOIN905): Source request file name missing for simulation mode.

Explanation: When running in simulation mode the file name for the SPML operation of the source connector is required, but is missing.

Action: Specify the file name for listing the SPML requests of the source connector.

FTL(JOIN906): Target request file name missing for simulation mode.

Explanation: When running in simulation mode the file name for the SPML operation of the target connector is required, but is missing.

Action: Specify the file name for listing the SPML requests of the target connector.

FTL(JOIN907): Channel id '*string1*' doesn't exist for port '*string2*'.

Explanation: There is no channel available that matches the given channel id.

Action: Check the configuration and correct the channel references.

FTL(JOIN909): No join expressions found.

Explanation: The configuration doesn't contain join expressions.

Action: Check the configuration.

FTL(JOIN910): The Java mapping class '*string1*' for attribute '*string2*' doesn't implement the required interface '*string3*'.

Explanation: The Java class used in the attribute mapping is wrong.

Action: Use the correct Java class in the attribute mapping.

FTL(JOIN911): '*string1*' exception while loading the Java mapping class '*string2*' for attribute '*string3*'; error = *string4*.

Explanation: Loading of the Java class failed.

Action: Check the error reason.

FTL(JOIN912): The Java mapping class '*string1*' for operational attribute '*string2*' doesn't implement the required interface '*string3*'.

Explanation: The Java class used in the operational attribute mapping is wrong.

Action: Use the correct Java class in the operational attribute mapping.

FTL(JOIN913): '*string1*' exception while loading the Java mapping class '*string2*' for operational attribute '*string3*'; error = *string4*.

Explanation: Loading of the Java class failed.

Action: Check the error reason.

FTL(JOIN914): The Java mapping class '*string1*' for parameter '*string2*' doesn't implement the required interface '*string3*'.

Explanation: The Java class used in the parameter mapping of an extended request is wrong.

Action: Use the correct Java class in the parameter mapping.

FTL(JOIN915): '*string1*' exception while loading the Java mapping class '*string2*' for parameter '*string3*'; error = *string4*.

Explanation: Loading of the Java class failed.

Action: Check the error reason.

FTL(JOIN916): The Java mapping class '*string1*' for identifier mapping doesn't implement the required interface '*string2*'.

Explanation: The Java class used in the identifier mapping is wrong.

Action: Use the correct Java class in the identifier mapping.

FTL(JOIN917): '*string1*' exception while loading the Java mapping class '*string2*' for identifier mapping; error = *string3*.

Explanation: Loading of the Java class failed.

Action: Check the error reason.

FTL(JOIN918): The Java mapping class '*string1*' for post mapping doesn't implement the required interface '*string2*'.

Explanation: The Java class used in the post mapping is wrong.

Action: Use the correct Java class in the post mapping.

FTL(JOIN919): '*string1*' exception while loading the Java mapping class '*string3*' for post mapping; error = *string4*.

Explanation: Loading of the Java class failed.

Action: Check the error reason.

FTL(JOIN920): The Java class '*string1*' in the user hook definition doesn't implement the required interface '*string2*'.

Explanation: The Java class used as user hook is wrong.

Action: Use the correct user hook Java class.

FTL(JOIN921): Loading the user hook Java class '*string1*' failed; error = *string2*.

Explanation: Loading of the Java class failed.

Action: Check the error reason.

FTL(JOIN922): Compilation of Java class '*string1*' failed; number of errors: *string2*.

Explanation: Errors while compiling a Java class occurred.

Action: Check the errors and correct the Java source.

FTL(JOIN923): Error at line *string1*: *string2*.

Explanation: Errors while compiling a Java class occurred.

Action: Check the errors and correct the Java source.

FTL(JOIN924): Error while reading file '*string1*'; error: *string2*.

Explanation: An error was detected while reading the file.

Action: Check the error.

FTL(JOIN925): Connector doesn’t support extended requests.

Explanation: The connector can’t execute the extended requests.

Action: Check the error.

FTL(JOIN926): No target system node found for DN '*string1*' in the channel '*string2*'.

Explanation: The export search Base has no target system node in the upper parts of the DN.

Action: Check the export search base.

FTL(JOIN927): The component "searchMembers" is either not set or wrong.

Explanation: The component "searchMembers" of the workflow configuration is missing or is not configured with correct values.

Action: Check the workflow configuration.

FTL(JOIN928): Compilation of Java class '*string1*' failed; exception = '*string2*'; error = *string3*.

Explanation: Errors while compiling a Java class occurred.

Action: Check the error and the stack trace.

FTL(JOIN929): Executing the Java attribute mapping class '*string1*' for attribute '*string2*' failed; exception = '*string3*'; error = *string4*.

Explanation: Executing the Java class failed.

Action: Check the error reason.

FTL(JOIN930): Decryption of password of '*string1*' failed; exception = '*string2*'.

Explanation: Decryption/Unscrambling of password failed.

Action: Check the error reason.

FTL(JOIN931): No join attributes defined in secondary channel '*string1*'.

Explanation: The secondary channel does not have any join attributes.

Action: Check the primary definition of the channel.

FTL(JOIN932): No corresponding channel defined for channel with channel id '*string1*' and port '*string2*'.

Explanation: The given channel has no corresponding channel.

Action: Check the definition of the channel.

FTL(JOIN933): No password channel found in port '*string1*'.

Explanation: There needs to be a password channel available when password events need to be processed.

Action: Check the configuration.

FTL(JOIN934): Executing the Java identifier mapping class '*string1*' failed; exception = '*string2*'; error = *string3*.

Explanation: Executing the Java class failed.

Action: Check the error reason.

FTL(JOIN935): Executing the Java operational attribute mapping class '*string1*' for attribute '*string2*' failed; exception = '*string3*'; error = *string4*.

Explanation: Executing the Java class failed.

Action: Check the error reason.

FTL(JOIN936): Executing the Java post-mapping class '*string1*' failed; exception = '*string2*'; error = *string3*.

Explanation: Executing the Java class failed.

Action: Check the error reason.

FTL(JOIN937): Executing the Java userhook class '*string1*' failed; exception = '*string2*'; error = *string3*.

Explanation: Executing the Java class failed.

Action: Check the error reason.

FTL(JOIN938): An unsupported encoding exception occurred with exception message: *string1*.

Explanation: An exception occurred with the given exception message.

Action: Check the exception message text.

16.5.23. Logging Messages (LCP)

This topic describes the logging messages returned by the DirX Identity ldap connection pool.

INF(LCP001): Bind to LDAP server '*string1*' with port '*string2*' as user '*string3*' successful.

Explanation: The connection to the LDAP server has been established successfully.

Action: None.

INF(LCP002): Reconnect to LDAP Server '*string1*' with port '*string2*' as user '*string3*' successful.

Explanation: The connection to the LDAP server has been re-established successfully.

Action: None.

INF(LCP003): Key/Truststore '*string1*' contains *string2* entries.

Explanation: Displays the aliases in the named key/truststore.

Action: None.

INF(LCP004): Aliasname '*string1*'.

Explanation: Displays the alias name in the named key/truststore.

Action: None.

ERR(LCP200): Failed to setup a SocketFactory for SSL for host '*string1*' and port *string2*, keystore '*string3*', truststore '*string4*': *string5*.

Explanation: SSL will not work properly, since server/worker encountered severe difficulties while initializing the SocketFactory.

Action: Inform DirX Identity technical support about the problem.

16.5.24. Logging Messages (REQWF)

This topic describes the logging messages returned by the IDSJ extension for Request-Workflows.

WAR(REQWF001): Couldn't initialize JMX support for extension '*string1*'.

Explanation: Request-Workflows won't be manageable via JMX due to initialization error.

Action: Inform DirX Identity technical support about the problem.

WAR(REQWF002): Couldn't initialize JMX support for repository '*string1*'.

Explanation: Request-Workflow repository won't be manageable via JMX due to initialization error.

Action: Inform DirX Identity technical support about the problem.

ERR(REQWF003): Task *string1* failed aborting transaction *string2*.

Explanation: Task execution failed, transaction will be aborted in order to reverse changes.

Action: Inform DirX Identity technical support about the problem.

WAR(REQWF004): Failed to configure UserMapping since server configuration doesn’t contain tag 'usermapping'.

Explanation: Invalid configuration file.

Action: Correct configuration file.

ERR(REQWF005): Failed to load configuration. 'DomainConfiguration' extension probably missing.

Explanation: Extension for Request-Workflows depends on extension 'DomainConfiguration'.

Action: Inform DirX Identity technical support about the problem.

WAR(REQWF006): Cannot determine if audit is enabled or not. Request workflow service does not do auditing (disabled).

Explanation: The routine to determine the audit flag timed out.

Action: Check previous messages to see why this flag can not be determined. Otherwise restart the IdS-J service.

ERR(REQWF007): Cannot get session. No check of participant dn is possible.

Explanation: A session to the domain is not available.

Action: Inform DirX Identity technical support about the problem.

WAR(REQWF008): DN '*string1*' of participant no. *string2* in collection is invalid: *string3*.

Explanation: DN of participant in not valid and is removed from the list of participants.

Action: Check the calculation of participants. If you use a class to find participant then the calculation might be wrong.

WAR(REQWF009): Participant no. *string1* in collection is empty (String is null or length is 0).

Explanation: DN of participant in not valid and is removed from the list of participants.

Action: Check the calculation of participants. If you use a class to find participant then the calculation might be wrong.

WAR(REQWF010): Participant no. *string1* in collection is not a String: *string2*.

Explanation: DN of participant in not valid and is removed from the list of participants.

Action: Check the calculation of participants. If you use a class to find participant then the calculation might be wrong.

WAR(REQWF011): *string1* when saving participants of activity '*string2*' in attribute '*string3*': *string4*.

Explanation: Just the dxrUserLink attribute could not be set during saving the activity of a workflow.

Action: Check the given message.

ERR(REQWF012): Internal error (cause: *string1*).

Explanation: Server encountered severe difficulties.

Action: Inform DirX Identity technical support about the problem.

INF(REQWF050): User '*string1*' REJECTED workflow '*string2*'.

Explanation: Backward compatibility to legacy approval workflows: User rejected workflow i.e. all activities where he/she is a participant.

Action: None.

INF(REQWF051): User '*string1*' ACCEPTED workflow '*string2*'.

Explanation: Backward compatibility to legacy approval workflows: User accepted workflow i.e. all activities where he/she is a participant.

Action: None.

INF(REQWF060): Activating session: *string1* objects in pool, *string2* objects managed.

Explanation: A session is borrowed from the request workflow session pool.

Action: None.

INF(REQWF061): Destroying session: *string1* objects in pool, *string2* objects managed.

Explanation: A session is destroyed in the request workflow session pool.

Action: None.

INF(REQWF062): Creating new session instance: *string1* objects in pool, *string2* objects managed.

Explanation: A new session is created in the workflow session pool.

Action: None.

WAR(REQWF063): *string1* when returning session to pool: *string2*.

Explanation: A session should be returned to the session pool. The problem is ignored.

Action: Check the details of the message.

WAR(REQWF064): *string1* getting session. Pooling was *string2*: *string3*.

Explanation: A request workflow component failed to obtain a session to Identity domain. If pooling was enabled, pool manager was used. Otherwise a 'master' session was cloned. The operation fails.

Action: Check the details of the message.

INF(REQWF065): Passivating object: *string1* objects in pool, *string2* objects managed.

Explanation: A session is returned to the workflow session pool.

Action: None.

ERR(REQWF100): PatternResolver failed (cause: *string1*).

Explanation: Failed to generate participants. See error cause for further information.

Action: Inform DirX Identity technical support about the problem.

WAR(REQWF101): No predecessor for activity '*string1*'.

Explanation: Failed to find predecessor while generating participants.

Action: Inform DirX Identity technical support about the problem.

WAR(REQWF150): Couldn’t switch on LDAP-tracing (cause: *string1*).

Explanation: Non-critical error. Tried to turn on LDAP-Tracing but failed. See error cause for further information.

Action: Inform DirX Identity technical support about the problem.

INF(REQWF151): Discarded session because it is NOT authenticated! Trying to reconnect.

Explanation: Sometimes occurs when request workflow instances are written/read to/from LDAP. Session will be discarded and a new one created.

Action: None.

WAR(REQWF152): Failed to update dxrResourceLink (cause: *string1*).

Explanation: LDAP error, see error cause for further information.

Action: Inform DirX Identity technical support about the problem.

WAR(REQWF153): Failed to update dxrSubjectLink (cause: *string1*).

Explanation: LDAP error, see error cause for further information.

Action: Inform DirX Identity technical support about the problem.

ERR(REQWF154): Saving workflow instance DN=*string1* failed: *string2*. Update of inconsistent subject/resource orders aborted.

Explanation: Inconsistencies between subject/resource orders and subject/resource DNs were detected. The inconsistencies cannot be fixed, since saving the corrected workflow instance failed.

Action: Check the message details. In case of corrupt subject or resource DNs, check in which process step the problem arises. Inform DirX Identity technical support about the problem.

WAR(REQWF155): *string1* while deleting object '*string2*': *string3*.

Explanation: Deletion of object was not successful due to the given reason.

Action: Check the message details.

WAR(REQWF156): *string1* while replacing object '*string2*' (entry '*string3*'): *string4*

Explanation: Replacement of object was not successful due to the given reason.

Action: Check the message details.

WAR(REQWF157): *string1* while creating object '*string2*': *string3*.

Explanation: Creation of object was not successful due to the given reason.

Action: Check the message details.

WAR(REQWF158): *string1* while reading object '*string2*': *string3*.

Explanation: Retrieval of object was not successful due to the given reason.

Action: Check the message details.

WAR(REQWF159): *string1* while searching for objects: *string2*.

Explanation: Searching for objects was not successful due to the given reason.

Action: Check the message details.

WAR(REQWF160): *string1* while searching for objects with filter '*string2*': *string3*.

Explanation: Searching for objects was not successful due to the given reason.

Action: Check the message details.

WAR(REQWF161): *string1* while calculating number of objects: *string2*.

Explanation: Number of objects unknown due to the given reason.

Action: Check the message details.

WAR(REQWF162): Cannot read request workflow without DN.

Explanation: The request workflow repository was asked to read a workflow, but no DN was given.

Action: Make sure that the client provides a valid identifier.

WAR(REQWF163): Workflow with DN='*string1*' exists in Cache, but not in LDAP.

Explanation: The request workflow with the given DN exists in the Cache, but not in LDAP anymore. It will be deleted from the Cache.

Action: Check, if the workflow is already finished.

WAR(REQWF164): *string1* when searching for workflows and activities beneath '*string2*': *string3*.

Explanation: The request workflow repository tried to check, if the workflow or one of its activities has to be updated from LDAP. It takes the object from its Cache.

Action: Check the subtree in LDAP.

ERR(REQWF165): *string1* when reading workflow with DN = '*string2*': *string3*.

Explanation: The request workflow repository tried to read the workflow with the given DN. The request is rejected.

Action: Check the workflow entry in LDAP and the availability of the LDAP server.

ERR(REQWF166): *string1* transforming RequestWorkflow Entry with DN = '*string2*' to Repository Instance: *string3*.

Explanation: The request workflow repository has read the workflow with the given DN from LDAP and tried to transform it to an internal representation. The request is rejected.

Action: Inform DirX Identity support about the problem.

WAR(REQWF167): Wrong class for RequestActivityInstance with DN = '*string1*': *string2*.

Explanation: The request workflow repository tried to add the activities to the workflow, but received an unexpected Java class. The activity is ignored.

Action: Check the entry in LDAP and the attributes that determine the object description.

ERR(REQWF168): *string1* while reading and adding activities to workflow with DN = '*string2*': *string3*.

Explanation: The request workflow repository tried to add the activities to the workflow, but received an exception. The request is rejected.

Action: Check the activity entries in LDAP.

ERR(REQWF169): *string1* when building request activity instance from Node with DN = '*string2*': *string3*.

Explanation: The request workflow repository has read the activity from LDAP and tries to transform it to an internal representation. The request is rejected.

Action: Check the activity entries in LDAP.

ERR(REQWF170): *string1* reading worklist with filter = '*string2*': *string3*.

Explanation: The request workflow repository was about to search the activities for a given user with the LDAP filter. The request is rejected.

Action: Check the filter and the matching activity entries in LDAP.

WAR(REQWF171): *string1* when parsing the template expression '*string2*' for property '*string3*': *string4*.

Explanation: The request workflow repository tried to create a value for the given property (for a workflow or activity based on the given template). A empty value is returned.

Action: Check the template.

ERR(REQWF172): *string1* while reading request activities with filter = '*string2*' and base = '*string3*': *string4*.

Explanation: The request workflow repository was about to search for activities matching the given LDAP filter. The request is rejected.

Action: Check the filter.

WAR(REQWF173): *string1* when checking consistency of DN's in workflow with DN = '*string2*': *string3*.

Explanation: The request workflow repository tried to synchronize DN for subject and resource(s) in XML-based orders. The failure is ignored.

Action: Check the workflow and the DN's used for subject and resource(s).

WAR(REQWF174): *string1* when checking consistency of DN's in workflow '*string2*': *string3*.

Explanation: Inconsistencies between subject/resource orders and subject/resource DNs were detected. The inconsistencies cannot be fixed, since saving the corrected workflow instance failed.

Action: Check the message details. In case of corrupt subject or resource DNs, check in which process step the problem arises. Inform DirX Identity technical support about the problem.

WAR(REQWF175): Empty filter or missing user in worklist filter: *string1*.

Explanation: The filter for finding the worklist of a user MUST specify a user.

Action: Make sure the client provides a user DN.

ERR(REQWF176): *string1* while writing activities of workflow '*string2*' to LDAP: *string3*.

Explanation: A workflow or one of its activities has been created or changed and needed to be updated in the Identity domain. Some changes may be lost.

Action: Check the details. Check the workflow and repair it manually if needed.

ERR(REQWF177): No object descriptor for *string1*.

Explanation: In order to write activities to LDAP the request workflow repository needs their object description, but cannot obtain one.

Action: Check the object description.

ERR(REQWF178): *string1* while writing workflow '*string2*' to Cache and LDAP: *string3*.

Explanation: A workflow or one of its activities has been created or changed and needed to be updated in the memory Cache and in the Identity domain. At least some of the changes are lost.

Action: Check the details. Repair the workflow manually if needed.

ERR(REQWF179): *string1* while (re-)storing the workflow '*string2*' in LDAP: *string3*.

Explanation: A workflow entry has been created or changed and needed to be stored in the Identity domain. All changes are lost.

Action: Check the details. Repair the workflow manually if needed.

INF(REQWF180): Cache for Request Workflows created with size *string1*.

Explanation: The cache for the request workflows is created with the given maximum size, which is configurable in server.xml.

Action: None.

ERR(REQWF181): *string1* in checkPermission for User '*string2*', operation create, object type *string3*: *string4*.

Explanation: The exception occurred while the request workflow component tried to read the user from LDAP, search and evaluate the appropriate access policies. The request is rejected.

Action: Check the details. Check access policies.

ERR(REQWF182): *string1* in checkPermission for User '*string2*', operation = *string3*, resource = *string4*: *string5*.

Explanation: The exception occurred while the request workflow component tried to read the user from LDAP, search and evaluate the appropriate access policies. The request is rejected.

Action: Check the details. Check access policies.

ERR(REQWF183): *string1* during creation of AclManager: *string2*.

Explanation: The request workflow component obtained a session to the domain and instantiated the access control manager.

Action: Check the details. Is the LDAP server available?

WAR(REQWF184): Start time value ignored: '*string1*'.

Explanation: The start time pattern could not be parsed. Value is ignored.

Action: Correct value to pattern 'hh:mm'.

INF(REQWF185): Rebuild key map will be processed initially at or after *string1*.

Explanation: The key map is used for faster internal access of request workflows. It is rebuild initially around the given time.

Action: None.

INF(REQWF186): Rebuild key map finished (*string1* items, *string2* msec).

Explanation: The key map is used for faster internal access of request workflows. Key map is rebuild.

Action: None.

WAR(REQWF187): Loading XML content of workflow or activity failed; error: '*string1*'.

Explanation: Error while loading the attribute 'dxmContent' of workflow or activity.

Action: Check the error message.

INF(REQWF188): Workflow '*string1*' is locked.

Explanation: Retrieval of object was not successful because it is locked by some other thread. The read operation is executed later again.

Action: None.

INF(REQWF189): *string1* while replacing object '*string2*' (entry '*string3*'): *string4*

Explanation: Replacement of object was not successful due to the given reason. The entry is locked by some other thread.

Action: Check the message details.

WAR(REQWF190): *string1* while creating object '*string2*': *string3*.

Explanation: Creation of object was not successful due to the given reason. The entry is locked by some other thread.

Action: Check the message details.

INF(REQWF200): Fullcheck started (checking all workflow entries).

Explanation: Fullcheck for Requestworkflows started.

Action: None.

INF(REQWF201): Fullcheck found expired activity *string1* of workflow *string2*.

Explanation: Fullcheck information about expired "people" activity.

Action: None.

INF(REQWF202): Fullcheck finished: Checked Workflows: *string1*, Reevaluated Workflows: *string2*, Workflows that already exist: *string3*.

Explanation: Fullcheck finished with the given statistics about workflows with expired "people" activities.

Action: None.

INF(REQWF203): Fullcheck started (checking expired workflows).

Explanation: Fullcheck for Requestworkflows started.

Action: None.

INF(REQWF204): No request workflows processed by Fullcheck on this server.

Explanation: Fullcheck for Requestworkflows is not running on this server.

Action: None.

INF(REQWF205): Fullcheck checks running activities/workflows after server restart.

Explanation: After restart of server, all already existing and running workflows/activities need to be restarted.

Action: None.

INF(REQWF206): Fullcheck failed to check/restart all relevant activities/workflows after server restart. Number of workflows: *string1* of *string2* workflows.

Explanation: After server restart some running activities/workflows couldn't be processed successfully.

Action: None.

INF(REQWF207): Fullcheck successfully checked/restarted all relevant activities/workflows after server restart. Number of workflows: *string1*.

Explanation: After server restart all running activities were successfully processed.

Action: None.

INF(REQWF208): Fullcheck created scheduler task for workflow ID '*string1*' and activity '*string2*'; scheduled for '*string3*'; operation: '*string4*'.

Explanation: A scheduler task has been created.

Action: None.

WAR(REQWF209): '*string1*' when checking running activities (after server restart): *string2*.

Explanation: Checking the running activities after server restart failed.

Action: Check the error message.

WAR(REQWF210): '*string1*' when retrieving running activities (after server restart): *string2*.

Explanation: Checking the running activities after server restart failed.

Action: Check the error message.

WAR(REQWF211): '*string1*' when reactivating workflow '*string2*' (after server restart): *string3*.

Explanation: Checking the running activities after server restart failed.

Action: Check the error message.

INF(REQWF213): WorkflowTask for workflow '*string1*}' not created due to memory problems.

Explanation: Due to heap memory problems FullCheck doesn't put the workflow to the Java Space. It will eventually be done in the next cycle of FullCheck.

Action: None.

INF(REQWF215): Fullcheck converts XML content of running activities/workflows.

Explanation: Fullcheck performs conversion of XML content attribute to LDAP attribute "dxmSpecificAttributes". That’s required for running workflows that were created by an earlier installation < V8.2B.

Action: None.

INF(REQWF216): Fullcheck: conversion of XML content successfully done.

Explanation: The XML content was successfully converted to "dxmSpecificAttributes".

Action: None.

WAR(REQWF217): Conversion of XML content by FullCheck failed.

Explanation: Reading and storing the workflow failed.

Action: Check the error message.

WAR(REQWF218): '*string1*' when converting XML content: *string2*.

Explanation: There are problems when converting the XML content.

Action: Check the error message.

WAR(REQWF219): '*string1*' when updating the domain after conversion of XML content has been done: *string2*.

Explanation: There are problems when updating the domain.

Action: Check the error message.

INF(REQWF220): Fullcheck: copying next approval date and participants to workflow.

Explanation: For fast / bulk approval some activity data is copied to the workflow.

Action: None.

INF(REQWF221): Fullcheck: copying next approval date and participants to workflow has been done.

Explanation: Copying some activity data to the workflow has been done.

Action: None.

WAR(REQWF222): '*string1*' when updating the domain after copying activity data to the worklfow has been done: *string2*.

Explanation: There are problems when updating the domain.

Action: Check the error message.

WAR(REQWF223): Copying activity data to workflow failed.

Explanation: Copying activity data failed.

Action: Check the error message.

WAR(REQWF224): '*string1*' when copying next approval date and participant to workflow: *string2*.

Explanation: There are problems when copying activity data to the workflow.

Action: Check the error message.

INF(REQWF225): Fullcheck checks for activities with application state 'RESOLVED' that need to be activated.

Explanation: Due to bulk approval there migth be workflows/activities that need to be activated.

Action: None.

INF(REQWF226): Fullcheck failed to activate all relevant activities/workflows with state 'RESOLVED'. Number of workflows: *string1* of *string2* workflows.

Explanation: After server restart some activities/workflows couldn’t be processed successfully.

Action: None.

INF(REQWF227): Fullcheck successfully activated all relevant activities/workflows with state 'RESOLVED'. Number of workflows: *string1*.

Explanation: After server restart all activities with state 'RESOLVED’were successfully processed.

Action: None.

WAR(REQWF229): '*string1*' when activating activities with state 'RESOLVED': workflow: *string2*; error: *string3*.

Explanation: Checking the activities with state 'RESOLVED' failed.

Action: Check the error message.

WAR(REQWF230): '*string1*' when retrieving activities with state 'RESOLVED': *string2*.

Explanation: Retrieval of activities with state 'RESOLVED' failed.

Action: Check the error message.

INF(REQWF231): Fullcheck is now sleeping for *string1* sec (for load balancing).

Explanation: FullCheck has generated many WorkflowTasks (via JMS) and sleeps now so that FullCheck will not find the same workflows again in next cycle (in case the tasks have not been processed in time by the workflow engine.).

Action: None.

WAR(REQWF232): '*string1*' when activating activities with state 'RESOLVED': *string2*.

Explanation: Checking the activities with state 'RESOLVED' failed.

Action: Check the error message.

WAR(REQWF233): Ignoring workflow '*string1*' since workflow was not read successfully.

Explanation: The workflow couldn’t be processed as the workflow couldn’t be read successfully from LDAP.

Action: Check the error while reading the workflow.

INF(REQWF300): "abort" started; authenticated user: '*string1*', workflowID: '*string2*', timeout: *string3*.

Explanation: Abort started.

Action: None.

INF(REQWF301): "abort" started; authenticated user: '*string1*', workflowID: '*string2*', activityName: '*string3*', timeout: *string4*.

Explanation: Abort started.

Action: None.

INF(REQWF302): "abort" of workflowID '*string1*' terminated successfully.

Explanation: Abort terminated successfully.

Action: None.

INF(REQWF303): *string1* during "abort" of workflowID '*string2*': *string3*.

Explanation: Abort terminated with error.

Action: None.

INF(REQWF304): "activate" started; authenticated user: '*string1*', workflowID: '*string2*', activityName: '*string3*', activationKey: *string4*, language: *string5*.

Explanation: Activate started.

Action: None.

INF(REQWF305): "activate" of workflowID '*string1*' terminated successfully.

Explanation: Activate terminated successfully.

Action: None.

INF(REQWF306): *string1* during "activate" of workflowID '*string2*', terminated with error *string3*.

Explanation: Activate terminated with error.

Action: None.

INF(REQWF307): "createWorkflowInstance" started; authenticated user: *string1*, workflowName: *string2*, language: *string3*.

Explanation: CreateWorkflowInstance started.

Action: None.

INF(REQWF308): "createWorkflowInstance" of workflowID '*string1*' terminated successfully.

Explanation: CreateWorkflowInstance terminated successfully.

Action: None.

INF(REQWF309): *string1* during creation of workflowID '*string2*': *string3*.

Explanation: CreateWorkflowInstance terminated with error.

Action: None.

INF(REQWF310): "createWorkflowInstanceFromDN" started; authenticated user: *string1*, wrorkflowDN: *string2*, language: *string3*.

Explanation: CreateWorkflowInstanceFromDN started.

Action: None.

INF(REQWF311): "createWorkflowInstanceFromDN" terminated successfully.

Explanation: CreateWorkflowInstanceFromDN terminated successfully.

Action: None.

INF(REQWF312): "createWorkflowInstanceFromDN" terminated with error *string1*.

Explanation: CreateWorkflowInstanceFromDN terminated with error.

Action: None.

INF(REQWF313): "createWorkflowWaitForIdle" started; authenticated user: '*string1*', workflowID: '*string2*', subject: '*string3*', resources: *string4*, timeout: *string5*, language: *string6*.

Explanation: CreateWorkflowWaitForIdle started.

Action: None.

INF(REQWF314): "createWorkflowWaitForIdle" terminated successfully.

Explanation: CreateWorkflowWaitForIdle terminated successfully.

Action: None.

INF(REQWF315): *string1* during "createWorkflowWaitForIdle" of workflowID '*string2*': *string3*.

Explanation: CreateWorkflowWaitForIdle terminated with error.

Action: None.

INF(REQWF316): "deny" started; authenticated user: '*string1*', actor: '*string2*', workflowID: '*string3*', reason: *string3*.

Explanation: Deny started.

Action: None.

INF(REQWF317): "deny" of user *string1* in workflow '*string2*' terminated successfully.

Explanation: Deny terminated successfully.

Action: None.

INF(REQWF318): *string1* during deny of User *string2* in workflow '*string3*': *string4*.

Explanation: Deny terminated with error.

Action: None.

INF(REQWF319): "getWorkflowInstance" started; authenticated user: *string1*, workflowID: '*string2*', language: *string3*.

Explanation: GetWorkflowInstance started.

Action: None.

INF(REQWF320): "getWorkflowInstance" of workflowID '*string1*' terminated successfully.

Explanation: GetWorkflowInstance terminated successfully.

Action: None.

INF(REQWF321): *string1* during "getWorkflowInstance" of workflowID '*string2*': *string3*.

Explanation: GetWorkflowInstance terminated with error.

Action: None.

INF(REQWF322): "getWorklist" started; authenticated user: *string1*, user: *string2*, language: *string3*.

Explanation: GetWorklist started.

Action: None.

INF(REQWF323): "getWorklist" for user *string1* terminated successfully. Found workflows: *string2*. Found activities(tasks): *string3*. Size limit exceeded: *string4*.

Explanation: GetWorklist terminated successfully.

Action: None.

INF(REQWF324): *string1* during "getWorklist" for user *string2*: *string3*.

Explanation: GetWorklist terminated with error.

Action: None.

INF(REQWF325): "grant" started; authenticated user: '*string1*', actor: '*string2*', workflowID: '*string3*', reason: *string4*.

Explanation: Grant started.

Action: None.

INF(REQWF326): "grant" of actor '*string1*' of workflowID '*string2*' terminated successfully.

Explanation: Grant terminated successfully.

Action: None.

INF(REQWF327): *string1* during "grant" of actor '*string2*' of workflowID '*string3*': *string4*.

Explanation: Grant terminated with error.

Action: None.

INF(REQWF328): "listDefinitions" started; authenticated user: *string1*, language: *string2*.

Explanation: ListDefinitions started.

Action: None.

INF(REQWF329): "listDefinitions" terminated successfully returning *string1* workflows.

Explanation: ListDefinitions terminated successfully.

Action: None.

INF(REQWF330): *string1* during "listDefinitions" : *string2*.

Explanation: ListDefinitions terminated with error.

Action: None.

INF(REQWF331): "listInstances" started; authenticated user: *string1*, language: *string2*.

Explanation: ListInstances started.

Action: None.

INF(REQWF332): "listInstances" terminated successfully. Found entries *string1*.

Explanation: ListInstances terminated successfully.

Action: None.

INF(REQWF333): *string1* during "listInstances": *string2*.

Explanation: listInstances terminated with error.

Action: None.

INF(REQWF334): "lookupUser" started; authenticated user: *string1*, userToken: *string2*.

Explanation: LookupUser started.

Action: None.

INF(REQWF335): "lookupUser" terminated successfully.

Explanation: LookupUser terminated successfully.

Action: None.

INF(REQWF336): "lookupUser" terminated with error *string1*.

Explanation: LookupUser terminated with error.

Action: None.

INF(REQWF337): "modify" started; authenticated user: '*string1*', workflowID: '*string2*', timeout: *string3*.

Explanation: Modify started.

Action: None.

INF(REQWF338): "modify" for workflowID '*string1*' terminated successfully.

Explanation: Modify terminated successfully.

Action: None.

INF(REQWF339): *string1* during workflow modification; workflowID '*string2*': *string3*.

Explanation: Modify terminated with error.

Action: None.

INF(REQWF340): "notifyParticipantChanged" started; authenticated user: '*string1*', workflowID: '*string2*'.

Explanation: NotifyParticipantChanged started.

Action: None.

INF(REQWF341): "notifyParticipantChanged" terminated successfully; workflowID: '*string1*'.

Explanation: NotifyParticipantChanged terminated successfully.

Action: None.

INF(REQWF342): "notifyParticipantChanged" terminated with error *string1*.

Explanation: NotifyParticipantChanged terminated with error.

Action: None.

INF(REQWF343): "resume" started; authenticated user: '*string1*', workflowID: '*string2*', timeout: *string3*.

Explanation: Resume started.

Action: None.

INF(REQWF344): "resume" started; authenticated user: '*string1*', workflowID: '*string2*',

activityName: '*string3*', timeout: *string4*.

Explanation: Resume started.

Action: None.

INF(REQWF345): "resume" of workflowID '*string1*' terminated successfully.

Explanation: Resume terminated successfully.

Action: None.

INF(REQWF346): *string1* during "resume" of activity '*string2*' of workflowID '*string3*': *string4*.

Explanation: Resume terminated with error.

Action: None.

INF(REQWF347): "suspend" started; authenticated user: '*string1*', workflowID: '*string2*', timeout: *string3*.

Explanation: Suspend started.

Action: None.

INF(REQWF348): "suspend" of workflowID '*string1*' terminated successfully.

Explanation: Suspend terminated successfully.

Action: None.

INF(REQWF349): *string1* during "suspend" of workflowID '*string2*': *string3*.

Explanation: Suspend terminated with error.

Action: None.

INF(REQWF350): "verify" started; authenticated user: *string1*, workflowName: *string2*, subject: *string3*, timeout: *string4*.

Explanation: Verify started.

Action: None.

INF(REQWF351): "verify" terminated successfully.

Explanation: Verify terminated successfully.

Action: None.

INF(REQWF352): "verify" finished with error: *string1*.

Explanation: Verify terminated with error.

Action: None.

INF(REQWF3522): No matching workflow definition found for verify workflow for subject '*string1*'.

Explanation: Verify terminated with error.

Action: Check the message details and the subject string.

INF(REQWF353): "waitForIdle" started; authenticated user: '*string1*', workflowID: '*string2*', timeout: *string3*.

Explanation: WaitForIdle started.

Action: None.

INF(REQWF354): "waitForIdle" of workflowID '*string1*' terminated successfully.

Explanation: WaitForIdle terminated successfully.

Action: None.

INF(REQWF355): *string1* during "waitForIdle" of workflowID '*string2*': *string3*.

Explanation: WaitForIdle terminated with error.

Action: None.

INF(REQWF356): "getWorkflowInstance": workflowID '*string1*' does not exist.

Explanation: GetWorkflowInstance couldn’t find a workflow with the given workflowID.

Action: None.

ERR(REQWF357): Repository "requestworkflow" has wrong type: *string1*.

Explanation: The repository implementation used for request workflow instances has not the expected class name.

Action: Internal error. Inform DirX Identity support.

ERR(REQWF358): Unknown type of filter-criteria: *string1*.

Explanation: The filter type for searching workflow definitions is not known.

Action: Internal error. Inform DirX Identity support.

ERR(REQWF359): workflowID '*string1*' either does not exist or user '*string2*' is not allowed to modify.

Explanation: If a workflow with this ID exists, the authenticated user has not access rights to modify it.

Action: Check the workflowID and the user’s access rights.

ERR(REQWF360): Activity '*string1*' of workflowID '*string2*' does not exist. No modification is applied to workflow.

Explanation: A set of modifications in the modify request applied to an activity with the given name. The workflow has no activity with this name.

Action: Check the workflowID and the activity name.

ERR(REQWF361): *string1* while modifying workflowID '*string2*'; error: *string3*.

Explanation: The workflow service performed a modification on a workflow.

Action: Check the details of this messages and the ones before.

ERR(REQWF362): *string1* while aborting transaction after failure of request '*string2*': *string3*.

Explanation: The workflow service tried to abort a transaction.

Action: Internal error. Inform DirX Identity support.

ERR(REQWF363): *string1* when writing audit record for request '*string2*': *string3*.

Explanation: The workflow service failed to write an audit record.

Action: Either the request parameters are substantially wrong or some issue with the LDAP connection occured.

ERR(REQWF364): *string1* when extracting creator from subject order '*string2*': *string3*.

Explanation: The workflow service tried to extract the creator of an order. It returns an empty string.

Action: Check the message details and the order content.

ERR(REQWF365): *string1* when setting subject '*string2*' in RequestWorkflowDefinitionFilter: *string3*.

Explanation: The workflow service tried to build a filter for searching workflow definitions. No workflow matches.

Action: Check the message details and the filter.

ERR(REQWF366): *string1* reading workflow definition for subject '*string2*': *string3*.

Explanation: The workflow service tried to suitable workflow definitions for the given subject. No workflow matches.

Action: Check the message details and the subject string.

ERR(REQWF367): *string1* when extracting resource order(s) from '*string2*': *string3*.

Explanation: The workflow service tried to extract the resources of an order. It returns an empty string.

Action: Check the message details and the order content.

ERR(REQWF368): *string1* when setting subject '*string2*' and resource orders '*string3*' in RequestWorkflowDefinitionFilter: *string4*.

Explanation: The workflow service tried to build a filter for searching workflow definitions. No workflow matches.

Action: Check the message details and the filter.

ERR(REQWF370): *string1* when creating legacy approval workflow with name '*string2*':

string3.

Explanation: The workflow service tried to start an old approval workflow. The request is rejected.

Action: Check the message details and the filter.

ERR(REQWF371): *string1* when creating a request workflow with name '*string2*': *string3*.

Explanation: The workflow service tried to start a request workflow. The request is rejected.

Action: Check the message details and the filter.

ERR(REQWF372): *string1* waiting for workflow with name '*string2*' going to be *string3*: *string4*.

Explanation: After having created a request workflow the workflow service waits until the workflow is in an idle state. The service continues.

Action: Check the message details and the filter.

ERR(REQWF373): RequestWorkflow Repository is of wrong class: *string1*.

Explanation: The workflow service tries to obtain the repository for request workflows from the server, but the class is not as expected. The request is rejected.

Action: Internal error. Inform DirX Identity support.

ERR(REQWF374): *string1* writing workflow task '*string2*' to Space: *string3*.

Explanation: The workflow service tries to notify the workflow engine to re-calculate the workflow. The request is ignored.

Action: Internal error. Inform DirX Identity support.

INF(REQWF375): No workflow found for given DN: *string1*.

Explanation: The request workflow repository was requested to read the workflow with the given DN. It return null.

Action: Check the monitor subtree, if you think there must be a workflow with this DN.

INF(REQWF376): Workflow not up-to-Date in Cache, reading from LDAP. Update Time in Cache: *string1*; Newer LDAP Entries: *string2*.

Explanation: The request workflow repository checked, if the workflow in its Cache is still up-to-date. Either the workflow entry or one of its activities were changed later in LDAP. The whole workflow is updated from LDAP.

Action: None.

ERR(REQWF377): Reading task list with filter '*string1*' failed: *string2*.

Explanation: Retrieval of activities failed due to LDAP problems.

Action: Check detailed message.

ERR(REQWF378): Reading workflow with DN '*string1*' failed: *string2*.

Explanation: Retrieval of workflow failed due to LDAP problems.

Action: Check detailed message.

ERR(REQWF379): *string1* while unmarshalling "subject" field of listDefinitionsRequest: *string2*.

Explanation: The workflow service tried to evaluate the XML Content of a workflow definition filter representing the Subject.

Action: Make sure the workflow client sends correct XML requests.

ERR(REQWF380): *string1* while unmarshalling Resource of listDefinitionsRequest: *string2*.

Explanation: The workflow service tried to evaluate the XML Content of a workflow definition filter representing the Resource.

Action: Make sure the workflow client sends correct XML requests.

INF(REQWF381): Workflow *string1* read from cache.

Explanation: The request workflow was read from the repository cache.

Action: None.

INF(REQWF382): Workflow *string1* stored in cache.

Explanation: The request workflow was stored in the repository cache.

Action: None.

INF(REQWF383): Workflow *string1* deleted from cache.

Explanation: The request workflow was deleted from the repository cache.

Action: None.

WAR(REQWF384): *string1* while transforming time string '*string2*' in attribute *string3* of workflow activity '*string4*' to Gregorian Calendar: *string5*.

Explanation: When reading request workflow activities from LDAP, the service transforms the various datetime strings to a Java Calendar object. When this fails, this date is ignored.

Action: Check the wrong attribute in the activity instance!

WAR(REQWF385): *string1* while transforming time string '*string2*' in attribute *string3* of workflowID '*string4*' to Gregorian Calendar: *string5*.

Explanation: When reading request workflows from LDAP, the service transforms the various datetime strings to a Java Calendar object. When this fails, this date is ignored.

Action: Check the wrong attribute in the workflow instance!

ERR(REQWF386): *string1* while transforming attribute '*string2*' of workflowID '*string3*' to

XML object: *string4*. XML String was: *string5*.

Explanation: When reading request workflows from LDAP, the service transforms subject and resources to order request objects. The request fails.

Action: Check the subject/resource attributes in the workflow instance!

INF(REQWF387): "getInstance": missing workflowID.

Explanation: GetInstance couldn't return a workflow because the request did not contain an Identifier.

Action: Make sure that the client passes a workflow identifier.

INF(REQWF388): "deltaUpdateCertification" started; authenticated user: '*string1*', workflowID: '*string2*', activityName: '*string3*', submit: *string4*, timeout: *string5*, language: *string6*.

Explanation: Update of certifications started.

Action: None.

INF(REQWF389): "deltaUpdateCertification" of workflowID '*string1*' terminated successfully.

Explanation: Update of certifications terminated successfully.

Action: None.

INF(REQWF390): *string1* during deltaUpdateCertification of workflowID '*string2*' terminated with error *string3*.

Explanation: Update of certifications terminated with error.

Action: None.

WAR(REQWF391): *string1* when writing audit record for request '*string2*': *string3*.

Explanation: The workflow service audit handler failed to write an audit record embedded in an signature element with empty signature. It now tries to write the unsigned record.

Action: Probably serialization of the audit record fails. Or some classloader problems during embedding into the <signedAuditRecord> element.

WAR(REQWF392): *string1* when writing audit record for request: *string2*.

Explanation: The workflow service audit handler failed to write an audit record. No audit record can be written.

Action: Probably serialization of the audit record fails. Or the internal channel communication to the audit listener(s) failed.

WAR(REQWF393): Producing default audit record for unknown request *string1*.

Explanation: The workflow service audit handler received an unknown request type. It writes a default audit record.

Action: Inform DirX Identity support.

WAR(REQWF394): Request subject not an OrderRequest, but *string1*.

Explanation: The workflow service audit handler received a request containing a <subject> element which is supposed to be of class OrderRequest. It ignores the <subject> for the audit record.

Action: Verify that the client sends a correct <subject> element.

WAR(REQWF395): Request resource not an OrderRequest, but *string1*.

Explanation: The workflow service audit handler received a request containing a <resource> element which is supposed to be of class OrderRequest. It ignores the <resource> for the audit record.

Action: Verify that the client sends a correct <resource> element.

WAR(REQWF396): Cannot find audit policy for object type *string1*: *string2*.

Explanation: The workflow service audit handler searches the audit policy for the given object type in order to know which identifying attributes to write. It writes no identifying attributes into the audit record.

Action: Check the audit policies in the DirX Identity domain and their configuration.

ERR(REQWF397): *string1* trying to obtain session to Identity domain: *string2*.

Explanation: The workflow service audit handler wasn't able to obtain a session during initialization in order to check if audit is enabled.

Action: Check the connection to the DirX Identity domain.

ERR(REQWF398): *string1* during initialization of Audit Handler: *string2*.

Explanation: The workflow service audit handler wasn't able to initialize: check if audit is enabled.

Action: Check other error messages. Inform DirX Identity support.

INF(REQWF399): Audit Handler successfully initialized; auditEnabled = *string1*.

Explanation: The workflow service audit handler finished initialization successfully. If auditEnabled = true, audit logs will be written.

Action: None.

ERR(REQWF400): *string1* when processing a message (outbound=*string2*) in Audit Handler: *string3*.

Explanation: The workflow service audit handler processed a message either before (outbound=false) or after it was handled by the Web Service. The audit record could not be written.

Action: Check other error messages. Inform DirX Identity support.

ERR(REQWF401): *string1* when processing a fault message (outbound=*string2*) in Audit Handler: *string3*.

Explanation: The workflow service audit handler processed a fault message either before (outbound=false) or after it was handled by the Web Service. The audit record could not be written.

Action: Check other error messages. Inform DirX Identity support.

ERR(REQWF402): *string1* when processing an incoming workflow request: *string2*.

Explanation: The workflow service audit handler processed a request and tried to transform the XML string to a Java class. This failed. No audit record will be written.

Action: Check the details of the error messages. Verify that the client sends a valid XML request.

ERR(REQWF403): *string1* when processing an outgoing workflow response: *string2*.

Explanation: The workflow service audit handler processed a response and tried to write the audit record. This failed. No audit record will be written.

Action: Check the details of the error messages.

ERR(REQWF404): *string1* when processing an outgoing workflow fault response: *string2*.

Explanation: The workflow service audit handler processed a fault response and tried to write the audit record. This failed. No audit record will be written.

Action: Check the details of the error messages.

WAR(REQWF405): *string1* when filling identifier attributes for entry with DN='*string2*': *string3*.

Explanation: The workflow service audit handler tried to read the configured identifier attributes from the LDAP user. No identifier attributes will be written to the audit record.

Action: Check the audit policy for the object type (probably user). Check the LDAP entry and its attributes.

WAR(REQWF406): *string1* when sending a signed audit record of activity '*string2*' to the Audit Channel: *string3*. The signed record was: *string4*.

Explanation: The modify workflow request contained an activity attribute named 'signedAudit'. The audit handler could not pass it to the audit channel listener.

Action: Either some IDS-J internal error occurred (check other error messages) or the given audit record is not valid.

WAR(REQWF407): *string1* when getting generalized time from activity *string2* '*string3*': *string4*.

Explanation: The activity to be audited contained a string for a date. The handler tries to transform it to generalized time, which failed. This attribute will not be audited.

Action: Check the date strings of the activity. Check also that the client sends a valid format.

WAR(REQWF408): *string1* when getting generalized time from workflow *string2* '*string3*': *string4*.

Explanation: The workflow to be audited contained a string for a date. The handler tries to transform it to generalized time, which failed. This attribute will not be audited.

Action: Check the date strings of the workflow. Check also that the client sends a valid format.

WAR(REQWF409): Fault message is null in FaultResponse.

Explanation: The workflow service audit handler received a fault response with an empty <fault> element. No error details can be written to the audit record.

Action: The Web Service has to return a valid error information. Inform DirX Identity support.

WAR(REQWF410): Unexpected XML element in Fault message: *string1*.

Explanation: The workflow service audit handler received a fault response which does not contain a <fault> element. No error details can be written to the audit record.

Action: The Web Service has to return a valid error information. Inform DirX Identity support.

WAR(REQWF411): No child <detail>/<faultResponse> in Fault message: *string1*.

Explanation: The workflow service audit handler received a fault response which does not contain a <detail>/<faultResponse> element. No error details can be written to the audit record.

Action: The Web Service has to return a valid error information. Inform DirX Identity support.

ERR(REQWF412): Exception *string1* occurred while reading identifier for workflow ID=*string2* from repository: *string3*. Check of child workflows aborted.

Explanation: An error occurred while evaluating the workflow's state by the states of its children. The workflow's state is not re-evaluated.

Action: Check the message details.

ERR(REQWF413): Cannot read identifier for workflow ID=*string1* from repository: Identifier not found.

Explanation: The workflow with the given uid does not exist in the repository. The workflow's state is not evaluated.

Action: Check the installation and preceding messages. Inform DirX Identity support.

ERR(REQWF414): Exception *string1* occurred while reading identifier for child workflow ID=*string2* of parent ID=*string3* from repository: *string4*.

Explanation: The child workflow with the given uid does not exist in the repository. The parent workflow's state is not updated with this child info.

Action: Check the message details.

ERR(REQWF415): Exception *string1* occurred while reading child workflow ID=*string2* for parent workflow uid=*string3* from repository: *string4*.

Explanation: The child workflow with the given uid cannot be read from the repository. The parent workflow's state is not updated with this child info.

Action: Check the message details.

ERR(REQWF416): Exception *string1* occurred in while processing workflow change tracking info for child=*string2*, parent id=*string3*, accountGroupChanges=*string4*: *string5*.

Explanation: The account group change info cannot be processed. Presumably the data is corrupt. The child's changes are not passed to the parent.

Action: Check the message details.

INF(REQWF417): Child workflow ID=*string1* of parent workflow uid=*string2* is in state *string3*. Child info not updated at the parent.

Explanation: The child workflow has not yet finished. Thus, the current workflow remains in a waiting state.

Action: None.

ERR(REQWF418): Exception *string1* occurred while reading workflow instance ID=*string2* from repository: *string3*. Check of child workflows aborted.

Explanation: The workflow instance cannot be read from the repository.

Action: Check the message details.

ERR(REQWF419): Exception *string1* occurred while evaluating synchronization activity *string2* of workflow *string3*: *string4*.

Explanation: The evaluation of finished vs. started child workflows failed. Presumable the XML format of the stored information is corrupt.

Action: Check the message details.

ERR(REQWF420): Exception *string1* occurred in while processing workflow change tracking info for id=*string2*, accountGroupChanges=*string3*: *string4*.

Explanation: The account group change info cannot be processed. Presumably the data is corrupt.

Action: Check the message details.

ERR(REQWF421): Exception *string1* occurred in workflow id=*string2* during context update: *string3*.

Explanation: The result of the child state analysis (started/finished child workflows,

account group changes) cannot be written to the workflow context.

Action: Check the message details.

WAR(REQWF422): *string1* while providing SvcSession (service session).

Explanation: The current session could not be evaluated (either cloning of session failed or session could not be returned from session pool).

Action: Check the message details.

WAR(REQWF423): Can’t return LDAP session from SvcSession object.

Explanation: The current LDAP session should be evaluated from an SvcSession object, but the operation failed.

Action: Check the message details.

WAR(REQWF424): Prepare for activity '*string1*' failed, set state to *string2*, applicationState to *string3* (cause: *string4*).

Explanation: Failed to set state and applicationState.

Action: Check workflow definition.

INF(REQWF425): ConstraintViolation for activity '*string1*' for workflowID '*string2*'. Set state to *string3*, applicationState to *string4*. Violation details: *string5*).

Explanation: The participants constraints implementation for a people activity threw a violation exception to indicate that the number of participants is not sufficient.

Action: None.

ERR(REQWF426): Exception occurred during list workflow definitions: *string1*

Explanation: List defintions failed.

Action: Check the message details.

WAR(REQWF427): *string1* when reading active participant with DN='*string2*': *string3*.

Explanation: The workflow service audit handler tried to read the identifying attributes and dxrUid from the LDAP user. No UID or identifying attributes will be written to the audit record.

Action: Check the LDAP entry and its attributes.

ERR(REQWF428): *string1* while bulk operation modifying workflow; error: *string2*.

Explanation: The workflow service performed a bulk operation.

Action: Check the details of this messages and the ones before.

INF(REQWF429): "bulk" started; authenticated user: *string1*.

Explanation: Bulk started.

Action: None.

INF(REQWF430): "bulk" terminated successfully.

Explanation: Bulk terminated successfully.

Action: None.

INF(REQWF431): *string1* during workflow bulk: *string2*.

Explanation: Bulk terminated with error.

Action: None.

WAR(REQWF432): Bulk modify - workflowID '*string1*' either does not exist or user '*string2*' is not allowed to modify.

Explanation: Bulk modify - If a workflow with this ID exists, the authenticated user has not access rights to modify it.

Action: Bulk modify - Check the workflowID and the user's access rights.

INF(REQWF434): Starting bulk approval notification thread...

Explanation: Creating new thread for engine notification.

Action: None.

INF(REQWF435): Bulk notification workflow UID: *string1*.

Explanation: Notification sent for given workflow uid.

Action: None.

ERR(REQWF436): Bulk approval interrupted due to error when retrieving the worklist.

Explanation: The workflow service performed a bulk operation.

Action: Check the details of this messages and the ones before.

INF(REQWF437): The task (DN: *string1*) was already approved by someone else.

Explanation: Task was already approved.

Action: None.

WAR(REQWF438): Could not start all nested activities: '*string1*'.

Explanation: A nested activity could not be started. It will try to start as many activities as necessary.

Action: Check the given error message.

ERR(REQWF439): *string1* executing workflow task '*string2*' for workflow '*string3*': *string4*.

Explanation: The workflow service tries to execute a people activity, but an error occurred.

Action: Internal error. Check the error message and eventually inform DirX Identity support.

INF(REQWF440): The SOAP response for a Workflow SOAP request does not contain any worklow ID.

Explanation: For some reason, the workflow ID is missing in the request to write an audit record.

Action: Most probably no workflow definition was found while executing 'createWorkflowInstance'. Check other log messages or your configuration to see why no workflow could be created.

INF(REQWF441): No SOAP response is available in the audit handler's context for a Workflow SOAP request; using Workflow ID 'NOT_CREATED'.

Explanation: The audit record uses a dummy workflow ID.

Action: Check other log messages or your configuration to see why no workflow could be created.

ERR(REQWF442): Error while searching entries with search base '*string1*' and filter '*string2*'; error: *string3*.

Explanation: Retrieval of activities failed due to LDAP problems.

Action: Check detailed message.

INF(REQWF443): "Null context: last page reached. Processing of simple paged results finished.".

Explanation: Last page of a paged search result reached.

Action: None.

INF(REQWF444): Setting cookie '*string1*' for next search.

Explanation: Shows the cookie for a paged search.

Action: None.

WAR(REQWF445): No master attribute available in activity '*string1*' of workflow '*string2*'.

Explanation: The attribute "dxmSpecificAttributes(master)" is not available in the given activity.

Action: Check the workflow in LDAP for attribute "dxmSpecificAttributes(master)". Note that only sub-activities with suffix "-0", "-1" etc. hold this attribute. Check whether or not the modifications for an activity are sent to the sub-activity, e.g. "-0", "-1".

INF(REQWF446): "modify" for workflowID '*string1*' terminated without approving (task is already approved by someone else or expired).

Explanation: Modify terminated but approval is not executed. Approval has either been done by someone else or the workflow or activity expired.

Action: None.

ERR(REQWF500): '*string1*' when processing a message (outbound=*string2*) in UWL Handler: *string3*.

Explanation: The workflow service UWL handler processed a message either before (outbound=false) or after it was handled by the Web Service.

Action: Check other error messages. Inform DirX Identity support.

ERR(REQWF501): '*string1*' when processing an outgoing workflow response: *string2*. Response type: *string3*.

Explanation: The workflow service UWL handler failed to process an response. Response was not modified. Client will probably fail.

Action: Check other error messages. Inform DirX Identity support.

ERR(REQWF502): '*string1*' when processing an incoming workflow request: *string2*. Request type: *string3*.

Explanation: The workflow service UWL handler failed to process an request. Response was not modified. Client will probably fail.

Action: Check other error messages. Inform DirX Identity support.

16.5.25. Logging Messages (RSKCTRL)

This topic describes the logging messages returned by the DirX Identity Risk Governance Controller .

INF(RSKCTRL000): *string1* The risk governance controller open at '*string2*'.

Explanation: The risk governance open.

Action: None.

INF(RSKCTRL001): *string1* The risk governance start at '*string2*'.

Explanation: The risk governance start.

Action: None.

INF(RSKCTRL002): *string1* The risk governance close at '*string2*' - processTime for open-close: '*string3*' ms.

Explanation: The risk governance close.

Action: None.

INF(RSKCTRL005): *string1* Statistics: eventsProcessedOk = *string2*, eventsProcessedFailed = *string3*, updates = *string4*, upToDate = *string5*.

Explanation: The event controller informs about the number, type and result of processed events within one workflow run.

Action: None.

ERR(RSKCTRL006): Error starting risk domain processing with credential: *string1*, domain: *string2* and Exception: *string3*.

Explanation: Domain processing failed to start due to one or more invalid attribute values.

Action: Check the server information, credentials or the risk policy.

WAR(RSKCTRL007): The user hook with class name '*string2*' returned a '*string1*' when calling it's close method: *string3*.

Explanation: In the close method the user hook may terminate its resources.

Action: Check for more details the log messages of the user hook.

ERR(RSKCTRL011): Connection option missing: server = '*string1*', port = '*string2*', user = '*string3*', domain = '*string4*'.

Explanation: The <connection> element of the job controller does not contain all necessary parameters.

Action: Make sure all needed connection parameters are provided.

ERR(RSKCTRL012): *string1* caught exception *string2* while reading configuration options: '*string3*'.

Explanation: In its open() method the controller caught an exception.

Action: Make sure all needed configuration parameters are provided.

16.5.26. Logging Messages (RTWF)

This topic describes the logging messages returned by the IDSJ extension for Realtime-Workflows.

WAR(RTWF001): Couldn't initialize JMX support for extension '*string1*'.

Explanation: Repository won't be manageable via JMX due to an initialization error.

Action: Inform the system administrator.

WAR(RTWF002): Couldn't initialize JMX support for repository '*string1*'.

Explanation: Repository won't be manageable via JMX due to an initialization error.

Action: Inform the system administrator.

16.5.27. Logging Messages (SCE)

This topic describes the logging messages returned by the Java-based DirX Identity encryption and decryption services.

ERR(SCE701): No port configured to get PSE.

Explanation: There is no port configured to get the PSE from the C++-based DirX Identity server.

Action: Configure the secure port at the system related to the C++-based DirX Identity server.

ERR(SCE702): 'Unknown host' exception caught while setting up socket connection to local host using address '*string1*': *string2*.

Explanation: Error in IP configuration: Cannot connect to localhost.

Action: Correct your network configuration.

ERR(SCE703): Cannot connect to the C++-based DirX Identity server to get the PSE via socket *string1*: *string2*. Check if the secure port socket number is free. Use another port. Restart C++-based DirX Identity server.

Explanation: Received an IO exception while trying to connect to the C++-based DirX Identity server to get the PSE via a socket.

Action: Check if the secure port socket number is free. Use another port. Restart C++-based DirX Identity server.

ERR(SCE704): Wrong message format received. number expected: *string1*.

Explanation: The client received a wrong message from the C++-based DirX Identity server. Cannot decrypt passwords.

Action: Check if the installed C++-based DirX Identity server version and the client version are compatible. Restart C++-based DirX Identity server and client. Create new PSE/certificate pair for the server_admin.

ERR(SCE705): Unexpected message from server received: *string1*.

Explanation: A message was received from the C++-based DirX Identity server that cannot be interpreted. Cannot decrypt passwords.

Action: Check if the installed C++-based DirX Identity server version and the client version are compatible. Restart C++-based DirX Identity server and client. Create new PSE/certificate pair for the server_admin.

ERR(SCE706): Unknown content received from server: Len *string1* content *string2*.

Explanation: A message was received that cannot be interpreted. Cannot decrypt passwords.

Action: Check if the installed C++-based DirX Identity server version and the client version are compatible. Restart C++-based DirX Identity server and client. Create new PSE/certificate pair for the server_admin.

ERR(SCE707): IOException caught while initializing PSE from the C++-based DirX Identity server: *string1*.

Explanation: An IO Error occurred while reading the PSE from the C++-based DirX Identity server. Cannot decrypt passwords.

Action: Check if the secure port socket number is free. Use another port. Restart C++-based DirX Identity server.

ERR(SCE708): Timeout expired while initializing PSE from the C++-based DirX Identity server (timeout: *string1* seconds).

Explanation: Timeout expired. Cannot decrypt passwords.

Action: Increase the timeout. Restart C++-based DirX Identity server and client.

ERR(SCE709): No message from server received.

Explanation: No reply from the C++-based DirX Identity server. Cannot decrypt passwords.

Action: Check availability of the C++-based DirX Identity server. Check if the secure port socket number is free. Use another port. Restart C++-based DirX Identity server.

16.5.28. Logging Messages (SCKJOB)

This topic describes the logging messages returned by the DirX Identity Socketed Job.

WAR(SCKJOB400): Exception *string1* in JobSessionFactory. Cannot get session from pool: *string2*.

Explanation: Failed to get a SvcSession from the session pool. Thus, a session clone is used.

Action: Check the message details.

WAR(SCKJOB401): Exception *string1* occurred in JobSessionFactory while returning session to pool: *string2*.

Explanation: Failed to return a SvcSession to the session pool.

Action: Check the message details. Maybe a reload configuration cleared the pool.

ERR(SCKJOB402): Could not parse configured LDAP port value *string1*, using default value *string2*.

Explanation: The job tried to read LDAP port value as configured in .../private/domain bind profile but failed.

Action: Check the LDAP port value configured in .../private/domain bind profile and correct it. If the problem persists, inform DirX Identity technical support about the problem.

ERR(SCKJOB500): Cannot access bind profile manager.

Explanation: ApplyChanges has no access to the bind profile manager. Maybe the job does not run privileged or it runs on a remote worker.

Action: Use a resource family for the activity that assures running the ApplyChanges job collocated. Add ApplyChanges to the privileged jobs in the server configuration.

ERR(SCKJOB501): Cannot access bind profile 'private/domain'.

Explanation: ApplyChanges has no access to the bind profile 'private/domain'. Maybe the profile does not exist, the job does not run privileged or it runs on a remote worker.

Action: Check/create the private bind profile for the domain. Use a resource family for the activity that assures running the ApplyChanges job collocated. Add ApplyChanges to the privileged jobs in the server configuration.

ERR(SCKJOB502): Creating a connected template session failed: *string1*.

Explanation: The session pool is built from one template session being connected to the data store. For some reason the template session cannot be created in a bound state.

Action: Check the bind credentials. Check class loading. Check if the LDAP server is up and accepts connections.

ERR(SCKJOB503): Error in JobSessionFactory: Cannot get template session.

Explanation: The session clone is built from one template session being connected to the data store. For some reason the template session cannot be created in a bound state.

Action: Check the preceding messages.

ERR(SCKJOB504): Exception *string1* in JobSessionFactory: Cannot get session clone: *string2*.

Explanation: The job failed to create a new session as a clone of the template session.

Action: Check the message details.

ERR(SCKJOB505): Could not add pool specification *string1*.

Explanation: The job tried to specify pool for reusing of the service session but failed.

Action: If the problem persists, inform DirX Identity technical support about the problem.

ERR(SCKJOB506): Could not add pool specification *string1* due to an exception: *string2*.

Explanation: The job tried to specify pool for reusing of the service session but failed.

Action: If the problem persists, inform DirX Identity technical support about the problem.

ERR(SCKJOB507): Error during socketed job execution, an exception occurred: *string1*.

Explanation: The socketed job failed during execution of the configured custom class because of an exception.

Action: Check the cause of the exception and try to solve it. If the problem persists, inform DirX Identity technical support about the problem.

16.5.29. Logging Messages (TCK)

This topic describes the logging messages returned by the DirX Identity Ticket Control.

ERR(TCK000): *string1* unsupported operation: '*string2*'.

Explanation: The controller does not implement this operation.

Action: Make sure the operation is not passed to the controller.

INF(TCK001): *string1* received *string2* with id = '*string3*' and topic = *string4*.

Explanation: The activity reader has passed this event to the activity controller.

Action: None.

ERR(TCK002): Connection option missing: server = '*string1*', port = '*string2*', user = '*string3*', domain = '*string4*'.

Explanation: The <connection> element of the job controller does not contain all necessary parameters.

Action: Make sure all needed connection parameters are provided.

ERR(TCK004): Cannot bind to domain *string1* on '*string2*:_string3_' as '*string4* with ssl = '*string5*'; ErrorCode = *string6*.

Explanation: Controller cannot bind to DirX Identity domain.

Action: Check address and bind credentials. Is LDAP server available?

ERR(TCK005): Exception *string1* when loading configuration from domain '*string2*': *string3*.

Explanation: Controller cannot load configuration from DirX Identity domain.

Action: Check domain configuration: Object descriptions, etc.

WAR(TCK006): Exception *string1* when closing connection to domain '*string2*': *string3*.

Explanation: On close controller tried to dispose and disconnect the session to the DirX Identity domain.

Action: Check if the LDAP server is still available.

INF(TCK007): No tickets to process beneath '*string1*'.

Explanation: At the moment there are no tickets available that need to be processed.

Action: None.

ERR(TCK008): createTicketProcessor failed Exception: *string1*.

Explanation: Unable to create Ticket processor. Processing of tickets not possible.

Action: Check the details of the exception.

ERR(TCK009): Save of Ticket '*string1*' failed with rc: *string2*.

Explanation: Ticket was successfully processed but the save to set the State APPLY_CHANGE_COMPLETED failed.

Action: Check rc and perform appropriate action.

ERR(TCK010): applyTicket for '*string1*' failed.

Explanation: applyTicket to process the orders of given ticket failed. State of ticket does not change.

Action: Check logging of TicketProcessor.

ERR(TCK011): Exception *string1* during applyTicket for DN = '*string2*'.

Explanation: applyTicket to process the orders of given ticket throws an exception. State of ticket does not change.

Action: Check the details of the exception.

ERR(TCK012): Exception *string1* in *string2* while reading configuration options: *string3*.

Explanation: The activity controller tried to read configured options and address and bind credentials from LDAP reader.

Action: Check the details of the exception. Check the configuration options.

ERR(TCK013): Exception '*string1*' during search for tickets to process: baseDN = '*string2*' filter='*string3*'.

Explanation: Search for tickets to be processed failed. No tickets were processed.

Action: Check the details of the exception.

INF(TCK014): Ticket '*string1*' successfully processed, state changed to '*string2*'.

Explanation: Ticket succesfully process. All orders applied. State changed to APPLY_CHANGE_COMPLETED.

Action: None.

WAR(TCK015): Saving the orders of approval workflow '*string1*' at ticket '*string2*' failed with rc *string3*.

Explanation: Approver may have changed some attribute values of the order. These changes should be stored at the ticket. But the save operations fails.Changes of approvers may be lost.

Action: Check logging.

INF(TCK016): Orders of approval workflow '*string1*' successfully transferred to ticket '*string2*'.

Explanation: Approver may have changed some attribute values of the order. These changes are now also stored at the ticket.

Action: None.

WAR(TCK017): Exception '*string1*' while fetching workflow instance '*string2*'of ticket '

string3'.

Explanation: Ticket has set a dxrWorkflowLink attribute which seems not to reference a existing workflow instance. DxrWorkflowLink is ignored. Ticket processing continues.

Action: Check the exception details. Also check the dxrWorkflowLink attribute.

INF(TCK018): Setting state of ticket '*string1*' to '*string2*': Reason approval workflow '*string3*' failed with state *string4*.

Explanation: Approval workflow failed. State of ticket changes to appropiate state.

Action: None.

INF(TCK019): Setting state of ticket '*string1*' to '*string2*': Reason approval workflow '*string3*' succeeded with applicationstate *string4*.

Explanation: Approval workflow succeeded but ticket in state in_approval. State of ticket changes to appropiate state.

Action: None.

WAR(TCK020): Saving state changes at ticket '*string1*' with rc : *string2*.

Explanation: State of approval workflow causes a state change of the ticket. The save fails. Ticket processing continues.

Action: Check the message details.

INF(TCK021): State of ticket '*string1*' successfully changed.

Explanation: State of approval workflow causes a state change of the ticket. State changed successfully.

Action: None.

INF(TCK022): Applying Master Ticket first '*string1*'.

Explanation: Process Master Ticket first. It’s a prerequisite for the Sub Ticket.

Action: None.

INF(TCK023): Ticket '*string1*' : master ticket '*string2*' not processed, skip it.

Explanation: Prerequisite Master Ticket could not be processed. Sub Ticket is not processed.

Action: Check state of Master ticket.

INF(TCK024): Ticket '*string1*' : master ticket '*string2*' processing failed, skip it.

Explanation: Prerequisite Master Ticket processing failed. Sub Ticket is not processed.

Action: Check state of Master ticket.

ERR(TCK025): Exception *string1* resolving user with DN = '*string2*' due to an addEvent: *string3*.

Explanation: After the controller received an addEvent it tried to resolve the user, but got a runtime exception.

Action: Check the message details.

ERR(TCK026): Exception *string1* resolving user with DN = '*string2*' due to a modifyEvent: *string3*.

Explanation: After the controller received a modifyEvent it tried to resolve the user, but got a runtime exception.

Action: Check the message details.

ERR(TCK027): Exception *string1* while reading object of type *string2* with DN = '*string3*': *string4*.

Explanation: The controller tries to read the node from the domain using the identifier from the event.

Action: Check the message details. Does the object exist and is it of expected type? Is the class allowed to read the node?

ERR(TCK028): Exception *string1* processing modify event for object of type *string2* and DN = '*string3*': *string4*.

Explanation: After the controller received a modifyEvent it tried to read and process the node, but got a runtime exception.

Action: Check the message details.

ERR(TCK029): Exception *string1* when reading LDAP entry with DN='*string2*' during link check of object with DN = '*string3*': *string4*.

Explanation: While processing a change event for the object, the controller tries to read all referenced LDAP entries and received an exception different to NO_SUCH_OBJECT.

Action: Check the LDAP entry.

ERR(TCK030): The LDAP entry with DN='*string1*' referenced from object with DN='*string2*' in attribute '*string3*' does not exist.

Explanation: While processing a change event for the object, the controller tries to read all referenced LDAP entries to detect broken links and received an LDAP exception code NO_SUCH_OBJECT. The link is deleted.

Action: Check the LDAP entry.

ERR(TCK031): Exception *string1* while saving object with DN = '*string2*': *string3*.

Explanation: The controller tries to save changes to the object referenced in the change event.

Action: Check the message details and the object to be changed.

ERR(TCK032): Exception *string1* while processing an *string2*Event with requestID = '*string4*' for object of type *string3* with DN = '*string5*': *string6*.

Explanation: The event controller received an exception while processing the change event.

Action: Check the message details and the object which has been changed and processed.

ERR(TCK033): User hook configuration for event controller '*string1*' does not contain a class name.

Explanation: The controller configuration contains a <userhook> element, but the class name for the user hook is missing.

Action: Enter a class name for the user hook or drop the <userhook> element.

ERR(TCK034): Event controller '*string1*' cannot load user hook with class name '*string2*': *string3*.

Explanation: The class loader returns an exception when trying to load the class.

Action: Check the class name for the user hook in the <userhook> element and make sure the respective jar file is in the classpath.

ERR(TCK035): The user hook with class name '*string2*' returned a '*string1*' when calling it's open method: *string3*.

Explanation: In the open method the user hook evaluates its configuration and initializes itself.

Action: Check for more detailed log messages of the user hook. Check the user hook configuration.

WAR(TCK036): The user hook with class name '*string2*' returned a '*string1*' when calling it's close method: *string3*.

Explanation: In the close method the user hook may terminate its resources.

Action: Check for more detailed log messages of the user hook.

ERR(TCK037): The user hook with class name '*string3*' returned a '*string1*' when processing an *string2*Event for Entry with DN='*string4*': *string5*.

Explanation: The event controller passed the event and the affected LDAP entry to the user hook. It ignores the exception and stores it's own modifications.

Action: Check the user hook implementation and more detailed log messages of the user hook.

INF(TCK038): *string1* statistics: eventsProcessedOk = *string2*, eventsProcessedFailed = *string3*, updates = *string4*, upToDate = *string5*.

Explanation: The event controller informs about the number, type and result of processed events within one workflow run.

Action: None.

INF(TCK039): *string1* finished processing a *string2* for Entry with DN='*string3*:' *string4*ms.

Explanation: Time the event controller needed for processing this event.

Action: None.

ERR(TCK040): *string1* expected associated entry for '*string4*' of type *string2*, but received entry of class *string3*. DN or string representation of entry: *string5*.

Explanation: The event controller searched for associated objects and checked their types.

Action: Check the associated object: either remove the reference to the entry handled in the event or change to another type.

INF(TCK041): *string1* is updating associated entry of class *string2* with DN '*string3*.

Explanation: The event controller is going to change the associated entry, because a reference or other attribute needs to be deleted or changed.

Action: None.

ERR(TCK042): *string1* received *string2* when initializing rule context: *string3*.

Explanation: On startup the event controller searches consistency rules and initializes the rule context.

Action: Check the LDAP connection or inform DirX Identity Support about the problem.

ERR(TCK043): *string1* occurred when applying rule with DN='*string2*' to entry with DN='*string3*': *string4*.

Explanation: The event runs the consistency rule for the entry identified in the change event. It ignores the problem.

Action: Check the rule implementation.

WAR(TCK045): Expected consistency rule but obtained object with DN = '*string1*' of class '*string2*'.

Explanation: The rule provider searched for consistency rules. The returned object is a storage object but not consistency rule. It ignores this object.

Action: Check and correct the LDAP entry with the given DN.

ERR(TCK046): Skipping processing of consistency rule with name = '*string1*' after initialization failure: *string2*.

Explanation: The consistency rule provider tried to read search filter, base and operation from found consistency rule entry.

Action: Check the message details. Check the referenced operator and the filter definition.

ERR(TCK047): Exception *string1* while searching consistency rules using filter = '*string2*': *string3*.

Explanation: The consistency rule provider searches consistency rules, but did not get a

result.

Action: Check the message details.

ERR(TCK048): Cannot read consistency rules due to missing session.

Explanation: The consistency rule provider was requested to read consistency rules for the first time or after timeout, but did not get a session with LDAP access.

Action: Internal problem. Inform DirX Identity support.

INF(TCK049): *string1* updated entry of class *string2* with DN '*string3*.

Explanation: The event controller changed the entry identified in the event and updated it in LDAP.

Action: None.

WAR(TCK050): Expected validation rule but obtained object with name = '*string1*' of class '*string2*'.

Explanation: The rule provider searched for validation rules (accept, cleanup, validate). The returned object is a storage object but not validation rule. It ignores this object.

Action: Check and correct the LDAP entry with the given DN.

ERR(TCK051): Skipping processing of validation rule with Name = '*string1*' after initialization failure: *string2*.

Explanation: The validation rule provider tried to read search filter, base and operation from found validation rule entry.

Action: Check the message details. Check the referenced operator and the filter definition.

WAR(TCK053): Exception *string1* when searching associated user for account with DN = '*string2*' and search filter = '*string3*': *string4*.

Explanation: The Account Event Controller searches the associated user for an account. It tries the next set of attributes.

Action: Check the message details and the search filter.

ERR(TCK054): Cannot read provisioning rules due to missing session.

Explanation: The provisioning rule provider was requested to read provisioning rules for the first time or after timeout, but did not get a session with LDAP access.

Action: Internal problem. Inform DirX Identity support.

ERR(TCK055): Exception *string1* when cloning session: '*string2*'.

Explanation: At the start of a new job the controller tries to clone an existing SvcSession object to the DirX Identity domain. The controller tries to create a new session.

Action: Check the message details. Check availability of the LDAP server.

INF(TCK056): Successfully cloned Session for domain *string1* and effective user with DN '*string2*'.

Explanation: The event controller cloned an existing session to Identity domain.

Action: None.

ERR(TCK057): Cannot read validation rules due to missing session.

Explanation: The validation rule provider was requested to read validation rules for the first time or after timeout, but did not get a session with LDAP access.

Action: Internal problem. Inform DirX Identity support.

INF(TCK058): Added *string1* rule to cache: *string2*.

Explanation: The validation rule provider read this rule and added it to its cache.

Action: None.

ERR(TCK059): Exception *string1* while searching validation rules using filter = '*string2*': *string3*.

Explanation: The validation rule provider searches validation rules, but did not get a result.

Action: Check the message details.

ERR(TCK060): *string1* while reading validation filter from group '*string2*': *string3*.

Explanation: The validation rule provider tries to get the validation filter for this group in order to check whether the imported or ignored group member matches the filter.

Action: Check the validation filter definition of the group.

INF(TCK061): Adding account '*string1*' to group '*string2*' as consequence of rule '*string5*'. Old state was: *string3*, new state = *string4*.

Explanation: The validation rule provider adds the account to the group because they match the rule subject and privilege filters respectively.

Action: None.

ERR(TCK062): *string1* while adding account '*string2*' to group '*string3*': *string4*.

Explanation: The validation rule provider could not add the account to the group or save the group. The problem is ignored.

Action: Check the message details and try to add the account manually.

ERR(TCK063): *string1* while reading user '*string2*' of account '*string3*': *string4*.

Explanation: The validation rule provider could not read the user with the given DN. The onAssignment rules for the account are applied without the user.

Action: Check the message details. Often the user is not needed for the operation.

INF(TCK064): Deleting account '*string1*' from group '*string2*' as consequence of rule '*string4*'. Old state was: *string3*.

Explanation: The validation rule provider removes the account from the group because it is a cleanup rule or they do not match any more the rule subject and privilege filters of a validation rule respectively.

Action: None.

ERR(TCK065): *string1* while removing account '*string2*' from group '*string3*': *string4*.

Explanation: The validation rule provider could not remove the account from the group or save the group. The problem is ignored.

Action: Check the message details and try to remove the account manually.

ERR(TCK066): *string1* when checking validation filter of group with DN = '*string2*' for account with DN = '*string3*' and validation filter = '*string4*': *string5*.

Explanation: The validation rule provider checked if the account matches the validation filter of the group. It assumes there is no match.

Action: Check the message details. Maybe you have to change the validation filter.

ERR(TCK067): *string1* when checking validation filter of obligations referencing the group with DN = '*string2*' for account with DN = '*string3*':_string4_.

Explanation: The validation rule provider checked if the account matches the validation filter of the obligations referencing the group. It assumes there is no match.

Action: Check the message details. Maybe you have to change the obligations.

WAR(TCK068): *string1* when searching for groups with search base = '*string2*' and filter = '*string3*' obtained from rule '*string4*': *string5*.

Explanation: The validation rule provider searches for groups matching the privilege filter(s) of the validation rule. It ignores the offended filter and continues with the next one.

Action: Check the message details and the filters.

WAR(TCK069): *string1* when casting obligationLink = '*string2*' of group with DN = '*string3*' to an obligation: *string4*.

Explanation: The validation rule provider instantiates the obligation links of the group as a SvcObligation object. It ignores the link and continues with the next one.

Action: Check the message details and the obligation link.

WAR(TCK070): *string1* when evaluating validation filter of obligation with DN = '*string2*': *string3*.

Explanation: The validation rule provider transforms the String value in the validation attribute of the obligation to a LDAP filter. It ignores the filter.

Action: Check the message details and the filter.

WAR(TCK071): Obligation link value '*string1*' of group with DN = '*string2*' is neither a String nor an array of Strings or links to Storage Objects.

Explanation: The validation rule provider tries to interpret the value as an obligation link. It ignores the link.

Action: Check the message details and the link.

WAR(TCK072): *string1* when reading obligation with DN = '*string2*' of group with DN = '*string3*': *string4*.

Explanation: The validation rule provider reads the obligation from LDAP. It ignores the link.

Action: Check the message details and the entry with the given DN.

WAR(TCK073): *string1* when reading groups where account with DN = '*string2*' is a member in state IMPORTED: *string3*.

Explanation: Before applying accept/cleanup rules the validation rule provider checks if the account is an imported member of some group. It does not apply any accept/cleanup rule on the account.

Action: Check the message details. Is the referencing in the target system correct?

INF(TCK074): Updating membership state of account '*string1*' in group '*string2*' to *string3* based on rule '*string4*'.

Explanation: By applying the accept/cleanup rule the validation rule provider changed the member state of the account from IMPORTED to IGNORE resp. DELETE.

Action: None.

WAR(TCK075): *string1* when updating membership state of account '*string2*' in group '*string3*': *string4*.

Explanation: The validation rule provider tries to change the membership state from IMPORTED to IGNORE resp. DELETE and save the group if it is necessary.

Action: Check the message details. Try to change the state manually if still necessary.

INF(TCK076): Associated user '*string2*' with account '*string1*'.

Explanation: The Account Event Controller succeeded in finding a user for the account matching the configured account attributes.

Action: None.

WAR(TCK077): *string1* when passing failed event for entry with DN '*string2*' to *string3*: *string4*.

Explanation: The event controller failed to process the change event. It tried to pass the event together with an error response to either the retry or error connector.

Action: Check the configuration for the retry or error connector. Check if the identifier section of the event is correct.

ERR(TCK078): *string1* while searching with baseDN = '*string2*' and filter = '*string3*': *string4*.

Explanation: The Storage Utilities performed a LDAP search resulting in this exception. The search internally switches to paged mode in case of sizelimit reached.

Action: Check the message details.

ERR(TCK079): *string1* when creating a session to Identity domain *string2*: *string3*.

Explanation: On start of the job the controller failed to clone an existing SvcSession or create a new one. The job is terminated.

Action: Check availability of the LDAP server, address and bind credentials.

INF(TCK080): Running consistency rule '*string1*' for Entry *string2* ...

Explanation: The Event Maintenance Controller applies the consisteny rule to the given entry.

Action: None.

ERR(TCK081): *string1* when reading LDAP entry with DN='*string2*' during link check of object with DN = '*string3*': *string4*.

Explanation: While processing a change event for the object, the controller tries to read all referenced LDAP entries and received an exception different to NO_SUCH_OBJECT.

Action: Check the LDAP entries involved.

INF(TCK082): *string1* reads provision rule with DN = *string2*.

Explanation: The Event Maintenance Controller reads the provisioning rules to be applied on changed users.

Action: None.

WAR(TCK083): *string1* received *string2* during processing of *string3* for Entry with id = '*string4*': *string5*.

Explanation: The controller cannot process the event any more and tries to put it into the error channel.

Action: Check the message details.

INF(TCK084): *string1* puts an event for Entry with DN = '*string2*' to the retry channel after a *string3*: *string4*.

Explanation: The controller cannot process the event, but puts it into the retry channel for later retry.

Action: Check the message details.

INF(TCK085): *string1* puts an event for Entry with DN = '*string2*' to the error channel after a *string3*: *string4*.

Explanation: The controller cannot process the event and puts it into the error channel.

Action: Check the message details.

ERR(TCK086): A unexpected exception has occurred with storage id=*string1*: *string2*.

Explanation: A sporadic problem has occurred.

Action: Repeat processing of the object.

INF(TCK087): A new LDAP session with ID=*string1* is created.

Explanation: The workflow uses this session.

Action: None.

INF(TCK089): The LDAP session with ID=*string1* is reused.

Explanation: The workflow uses this session.

Action: None.

INF(TCK090): The LDAP session with ID=*string1* is returned to the pool.

Explanation: The workflow releases this session.

Action: None.

WAR(TCK091): *string1* during close of the job controller '*string2*': *string3*.

Explanation: The close method of the parent class of the event maintenance controller returned with an exception. It is ignored.

Action: Check for more detailed log messages of the parent controller.

INF(TCK092): Failure removing privilege with DN = '*string1*' from user with DN = '*string2*' based on rule with DN = '*string3*'.

Explanation: The user has the privilege and it must be removed due to the deny rule, but this failed. Reason is not known here.

Action: Check other error messages before this one and check also the user and the privilege objects.

ERR(TCK093): *string1* cannot get a session to Identity domain *string2*.

Explanation: On start of the job the controller failed to clone an existing SvcSession or create a new one. The job is terminated.

Action: Check availability of the LDAP server, address and bind credentials.

16.5.30. Logging Messages (WEBADM)

This topic describes the logging messages returned by the IDSJ extension for Web Admin.

ERR(WEBADM001): Could not load LDAP bind profile '*string1*'.

Explanation: The Web Admin could not load the LDAP bind profile. The JMX

configuration of the primary C++-based Server will be ignored.

Action: Check the existence of the bind profile.

ERR(WEBADM002): No valid configuration object with search base '*string1*' and filter '*string2*' was found.

Explanation: The JMS system was unable to read described information about active message server. No or multiple entries was found.

Action: If the issue continues check if your JMS configuration is correctly configured, check activated service, host, port and SSL property of the JMS service.

ERR(WEBADM003): Connection to JMX port failed: '*string1*'.

Explanation: The Web Admin was unable to open a connection to the JMX port of the server.

Action: If the issue continues check if your JMX configuration is correctly configured, check activated service, host, port and SSL property of the JMS service.

WAR(WEBADM101): Could not get the LDAP connection to the host '*string1*' on port '*string2*'.

Explanation: The Web Admin could not read the JMX connection parameters from the configured LDAP server.

Action: Check the availability of the LDAP server, check the configuration of the LDAP parameters in the file install_path/ids-j-<domain>-S<number>/bin/bindcredentials.xml.

WAR(WEBADM102): Could not execute the LDAP search operation with base '*string1*', filter '*string2*' and requested attribute '*string3*'.

Explanation: The Web Admin could not read the described property from LDAP server.

Action: Check the availability of the LDAP server, check the configuration of the LDAP parameters in the file install_path/ids-j-<domain>-S<number>/bin/bindcredentials.xml, check the existence of the described entry.

16.6. Logging Messages for the DirX Identity Manager

GUI... Logging messages for the DirX Identity Manager.

16.6.1. Logging Messages (GUI)

This topic describes the logging messages returned by the DirX Identity Management Client.

ERR(GUI100): *string1*.

Explanation: Generic error message.

Action: Check the detailed message.

WAR(GUI200): *string1*.

Explanation: Generic warning message.

Action: Check the detailed message.

INF(GUI300): *string1*.

Explanation: Generic log message.

Action: Check the detailed message.

FTL(GUI500): *string1*.

Explanation: Generic fatal error message.

Action: Check the detailed message.

16.7. Logging Messages for the DirX Identity Services

APM... Logging messages for approval management.

AUD... Logging messages for DirX Identity Auditing.

CAG... Logging messages for the DirX Identity campaign generator.

CHM... Logging messages for the Change Management.

GID... Logging messages for the Siemens global UID generation service.

HIS... Logging messages for the history (audit) component.

LAN... Logging messages for the DirX Identity log file analyzer.

LMT... Logging messages for the DirX Identity log file merging tool.

LNC... Logging messages for the link checker utility.

MCO... Logging messages for the mail service.

POA... Logging messages for the policy service.

PSV... Logging messages for the provisioning Web service.

PWDMGMT... Logging messages for the password management service.

PWRST... Logging messages for the password reset notification service.

RBA... Logging messages for the provisioning service.

RPC... Logging messages for the rule processor.

RWF... Logging messages for the request workflow service.

SEC... Logging messages for the security manager.

SRT... Logging messages for the realtime workflow services.

SVC... Logging messages for the privilege resolution service.

SVO... Logging messages for the DirX Identity order processor and service components.

WDM... Logging messages for the Order Data Manager.

16.7.1. Logging Messages (APM)

This topic describes the logging messages returned by the Approval management.

ERR(APM500): Cannot resolve attribute approval workflows to start: Searching for attribute policies failed: *string1*.

> **Explanation**: The attribute policy cache cannot be read from the data store. No attribute approval workflows are started. Save of the object is aborted.
>
> **Action**: Check server and network connection.

ERR(APM501): Calculating approvals for junior role assignments of role DN=*string1* failed: *string2*.

> **Explanation**: The junior role assignment approvals for the role could not be calculated.
>
> **Action**: Check the message details. Correct the error and repeat the assignment.

ERR(APM502): Calculating approvals for permission assignments of role DN=*string1* failed: *string2*.

> **Explanation**: The permission assignment approvals for the role could not be calculated.
>
> **Action**: Check the message details. Correct the error and repeat the assignment.

ERR(APM503): Calculating approvals for group assignments of permission DN=*string1* failed: *string2*.

> **Explanation**: The group assignments to be approved of a permission could not be calculated.
>
> **Action**: Check the message details. Correct the error and repeat the assignment.

ERR(APM504): Error while creating order: *string1*.

> **Explanation**: Creating an order failed.
>
> **Action**: Check detailed message. Check installed jar files, especially dxmOrder.jar. Inform DirX Identity technical support about the problem.

ERR(APM505): Cannot read SendEvent Policy; LDAP search failed: *string1*.

> **Explanation**: The send event policy cache cannot be read from the data store. Update of the policy is retried after some waiting time.

Action: Check server and network connection.

ERR(APM506): *string1* when evaluating Send Event policy: *string2*.

Explanation: An exception occurred while the policy cache evaluated the structured attribute dxrEventCfg of the active Event Policies.

Action: Check the content of the event policy attributes.

ERR(APM507): Cannot read ObjectDelete Policy; LDAP search failed: *string1*.

Explanation: The ObjectDelete policy cache cannot be read from the data store. Update of the policy is retried after some waiting time.

Action: Check server and network connection.

ERR(APM508): *string1* when evaluating ObjectDelete policy: *string2*.

Explanation: An exception occurred while the policy cache evaluated the structured attribute dxrPolicyCfg of the active ObjectDelete Policies.

Action: Check the content of the object delete policy attributes.

ERR(APM509): Start of an attribute modification approval workflow failed for DN=*string1*.

Explanation: Start of an approval workflow failed.

Action: Unassign privilege, quit user, and assign privilege again.

ERR(APM510): Start of an delete object approval workflow failed for DN=*string1*.

Explanation: Start of an approval workflow failed.

Action: Unassign privilege, quit user, and assign privilege again.

ERR(APM511): Start of an approval workflow to *string1* assignment for resource DN=*string2* to subject DN=*string3* failed.

Explanation: Start of an approval workflow failed.

Action: Unassign privilege, quit user, and assign privilege again.

ERR(APM512): Remote exception during start of an approval workflow for assignment of privilege DN=*string1* to subject DN=*string2*: *string3*.

Explanation: During start of an approval workflow, a remote exception occurred, indicating an error while communicating with the approval engine.

Action: Check whether the Java based server is running. Check the approval workflow state.

ERR(APM513): Start of an approval workflow for assignment of privilege DN=*string1* to subject DN=*string2* failed: *string3*.

Explanation: Start of an approval workflow failed.

Action: Unassign privilege, quit user, and assign privilege again. Check log files of IdS-J

server.

ERR(APM514): Remote exception during start of an approval workflow for modifications of subject DN=*string1* failed: *string2*.

Explanation: During start of an approval workflow, a remote exception occurred, indicating an error while communicating with the approval engine.

Action: Check whether the Java based server is running. Check the approval workflow state.

ERR(APM515): Start of an approval workflow for modification of subject DN=*string1* failed: *string2*.

Explanation: Start of an approval workflow failed.

Action: Quit user, and perform user modifications again.

ERR(APM516): Cannot cancel request workflow uid=*string1*: *string2*.

Explanation: The request workflow with the given UID cannot be aborted.

Action: Check detailed message. Check workflow state in the workflow monitor. Cancel workflow in Web Center.

ERR(APM517): No matching *string1* workflow found for subject DN=*string2*.

Explanation: Either no active workflow is applicable for that type of subject and operation. Or you do not have the right to read/execute the workflow.

Action: Call your workflow administrator to create a suitable workflow and/or create the missing policies.

16.7.2. Logging Messages (AUD)

This topic describes the logging messages returned by the Storage Layer Auditing.

ERR(AUD000): Unexpected exception: '*string1*'.

Explanation: An exception occurred.

Action: Check the message text for details and perform appropriate actions.

WAR(AUD002): *string1*

Explanation: A recoverable error situation encountered.

Action: Check the message text for details and perform appropriate actions.

ERR(AUD003): *string1*

Explanation: A severe error occurred.

Action: Check the message text for details and perform appropriate actions. If the error situation persists inform DirX Identity technical support about the problem.

ERR(AUD400): Duplicate *string1* definition for attribute '*string2*' of audit configuration DN=*string3*.

Explanation: A duplicate definition exists in an audit configuration.

Action: Rename the attribute or remove one of the duplicates.

ERR(AUD500): Audit configuration DN=*string1* is empty. No audit information is created from this configuration.

Explanation: No attributes or assignments are configured for auditing in the audit configuration.

Action: Check and correct the configuration.

ERR(AUD501): Error in XML part of audit configuration DN=*string1*: *string2*.

Explanation: The content of the audit configuration has an incorrect XML structure.

Action: Correct the configuration.

ERR(AUD504): Validation error while creating the XML representation of an audit configuration: *string1*.

Explanation: The audit configuration has an invalid object structure. Maybe mandatory entries are missing.

Action: Check the message details. Correct the configuration.

ERR(AUD505): No auditing configured for subject DN=*string1* of assignment to resource DN=*string2*. The audit record is incomplete.

Explanation: The subject of an assignment has no auditing configured. Thus, the audit record for the assignment is incomplete.

Action: Add an audit configuration for this object type.

INF(AUD506): No auditing configured for resource DN=*string1* of assignment from subject DN=*string2*. Audit record is incomplete.

Explanation: The resource of an assignment has no auditing configured. Thus, the audit record for the assignment is incomplete.

Action: Add an audit configuration for this object type.

ERR(AUD507): UID is missing for subject DN=*string1* of assignment to resource DN=*string2*.

Explanation: The subject of an assignment has no UID. Thus, the audit record for the assignment uses UNDEFINED for that uid.

Action: Run audit migration from install_path/gui/migration/CreateDxrUids to fill all empty dxrUID attributes.

ERR(AUD508): UID is missing for resource DN=*string1* of assignment from subject DN=*string2*.

Explanation: The subject of an assignment has no UID. Thus, the audit record for the assignment uses UNDEFINED for that uid.

Action: Run audit migration from install_path/gui/migration/CreateDxrUids to fill all empty dxrUID attributes.

ERR(AUD509): Duplicate audit definition DN=*string1* for type=*string2*. Ignoring configuration.

Explanation: An audit configuration is processed whose type is configured already.

Action: Check and correct the dxrType attribute of the configuration.

ERR(AUD510): Error while creating the XML representation of the audit record for DN=*string1*: *string2*.

Explanation: The audit record has an invalid object structure.

Action: Check the message details and perform appropriate actions. If the error situation persists inform DirX Identity technical support about the problem.

ERR(AUD511): Mandatory attribute *string1* missing in audit configuration DN=*string2*.

Explanation: A mandatory attribute is missing in the audit configuration.

Action: Complete the configuration.

ERR(AUD512): Error reading the effective user. Incomplete audit record created for DN=*string1*.

Explanation: The effective (logged in) user cannot be determined. The active participant element of the audit record is incomplete.

Action: Check login state. If the error situation persists inform DirX Identity technical support about the problem.

ERR(AUD513): Cannot find audit configuration for the logged-in user DN=*string1*. Incomplete audit record created for DN=*string1*.

Explanation: There is no auditing configured matching the effective (logged in) user. The active participant element of the audit record is incomplete.

Action: Auditing is referenced via the object description name in the dxrType attribute of the configuration. Possibly the user has no dxrUser auxiliary object class.

ERR(AUD514): Cannot access resource of assignment from *string1* to *string2*: *string3*. Audit record is incomplete.

Explanation: The resource of an assignment cannot be loaded. Thus, the audit record for the assignment is incomplete.

Action: Check the LDAP server connection. Correct the audit record manually.

ERR(AUD515): Error saving rename history from DN=*string1* to DN=*string2*: *string3*.

Explanation: An error occurred during save of the object with the rename history

record.

Action: Check the LDAP server connection. Check the detailed message and perform appropriate actions.

ERR(AUD516): Create of history for '*string1*' of object DN='*string2*' failed: Error reading object referenced by '*string3*': '*string4*'.

Explanation: Could not determine referenced object when creating the history of a dxrGroupMember* attribute. No history is written.

Action: Check the LDAP server connection. Check the detailed message and perform appropriate actions.

ERR(AUD517): Establishing a marshaler for converting from XML string to Object or vice versa failed; error: '*string1*'.

Explanation: There are problems while creating the environment for XML handling.

Action: Check the error message. Contact the support team.

16.7.3. Logging Messages (CAG)

This topic describes the logging messages returned by the DirX Identity campaign generator.

INF(CAG201): Program '*string1*', Version '*string2*' started ***

Explanation: Start message of the campaign generator.

Action: None.

INF(CAG202): Program '*string1*' ended with *string2* error(s) and *string3* warning(s).

Explanation: Informs the user that the program has ended correctly. Shows an error statistics.

Action: None.

INF(CAG210): Processing *string1* DN=*string2*.

Explanation: Informs the user which privilege is being processed.

Action: None.

INF(CAG211): *string1* workflow instances from *string2* logfiles processed.

Explanation: Shows a statistics on the processed objects.

Action: None.

INF(CAG212): Reading certification workflow definition DN=*string1* from subject.

Explanation: A workflow definition has been read from the subject's dxrCertificationWorkflowLink.

Action: None.

INF(CAG213): Starting workflow DN='*string1*' for privilege '*string2*' and *string3* assigned users.

Explanation: A workflow is being started for the denoted privilege.

Action: None.

INF(CAG214): Processing complete. *string1* workflows started.

Explanation: A total number of <n> certification workflows has been started.

Action: None.

INF(CAG215): No certification workflow configured for *string1* DN=*string2*. Selection of the workflow is done by the Java Server's 'When Applicable' section.

Explanation: There is no requestworkflow configured for access certification. Thus, the workflow is selected by the Java Server.

Action: Configure a global workflow in the campaign generator's ini file. Configure an individual workflow at the privilege. Configure the 'When Applicable' section of you certification workflow.

INF(CAG216): Starting workflow for privilege '*string1*' and *string2* assigned users.

Explanation: A workflow is being started for the denoted privilege.

Action: None.

ERR(CAG400): No certification workflow configured for *string1* DN=*string2*: Skipping *string1*.

Explanation: There is no requestworkflow configured for access certification. Thus, no workflow is started for certification of the privilege.

Action: Configure a global workflow in the campaign generator's ini file. Configure an individual workflow at the privilege.

ERR(CAG401): Cannot create subject order for *string1* DN=*string2*: *string3*.

Explanation: An error occurred to create the subject modify order. Thus, no workflow is started for certification of the privilege.

Action: Check the detailed message. Check your installation. Remove error condition ad re-run workflow.

ERR(CAG402): Cannot create resource order for *string1* DN=*string2* / user DN=*string3*: *string4*.

Explanation: An error occurred to create the resource certification order. Thus, no workflow is started for certification of the privilege.

Action: Check the detailed message. Check your installation. Remove error condition ad re-run workflow.

ERR(CAG403): Searching for *string1*s failed for filter='*string2*', base='*string3*': *string4*.

Explanation: An error occurred while searching the privileges to be certified.

Action: Check the detailed message. Check your installation. Remove error condition ad re-run workflow.

ERR(CAG404): Reading next *string1* failed: *string2*. Check your paging policy parameters.

Explanation: An error occurred while requesting the next page of privileges from the server. Presumably a paging policy timeout has occurred.

Action: Check the detailed message. Increase paging policy timeout. Reduce page size.

ERR(CAG405): Could not start workflow *string1*.

Explanation: The request workflow displayed in the message could not be started.

Action: Check if the workflow exists and is active. Check for error messages in the Java Server log.

ERR(CAG406): Could not start workflow *string1*: *string2*.

Explanation: The request workflow displayed in the message could not be started.

Action: Check the message details. Check if the workflow exists and is active. Check for error messages in the Java Server log.

ERR(CAG407): Could not initialize workflow service.

Explanation: The request workflow service could not be initialized. Thus, no workflows are started.

Action: Check the preceding messages.

ERR(CAG408): Could not set expiration date to *string1* for workflow*string2*: *string3*.

Explanation: Setting the expiration date at the workflow instance failed.

Action: Check the detailed message. Set date manually.

ERR(CAG409): Could not set expiration date for workflow *string1*: Could not read workflow instance.

Explanation: Reading the workflow instance failed, no object was returned by the search. Thus, setting the expiration date at the workflow instance failed.

Action: Check existence of the workflow instance. Set date manually.

ERR(CAG410): Could not set expiration date for workflow *string1*: More than one workflow instance exists.

Explanation: Reading the workflow instance failed, more than one object was returned by the search. Thus, setting the expiration date at the workflow instance failed.

Action: Check the request workflow monitor entries. Remove duplicazes. Set date manually.

ERR(CAG411): Could not set dxrCertificationPending=true for *string1* DN=*string2*: *string3*.

Explanation: The access certification has been started but the privilege could not be marked. Thus, the next run of the campaign processor will start another workflow for this privilege.

Action: Mark privilege manually.

ERR(CAG412): An error has occurred while processing *string1*s: *string2*. Processing aborted.

Explanation: Processing of the denoted privilege type is aborted due to an unexpected error.

Action: Check the detailed message.

ERR(CAG413): An error has occurred: *string1*. Processing aborted.

Explanation: Processing is aborted due to an unexpected error.

Action: Check the detailed message.

ERR(CAG414): No active workflow found for certification.

Explanation: The DN of the workflow to be started is not configured. And the Java Server does not find an applicable workflow to start.

Action: Check if a workflow with a corresponding when applicable section exists and is active. Check for error messages in the Java Server log.

ERR(CAG415): Could not start workflow: *string1*.

Explanation: A request workflow for certification could not be started.

Action: Check the message details. Check if a matching workflow exists and is active. Check for error messages in the Java Server log.

ERR(CAG416): Error searching users with filter=*string1*, base=*string2*: *string3*. Cannot determine values of grouping attribute *string4*.

Explanation: Grouping of users according to an attribute is configured. The search for the users to determine the set of attribute values fails.

Action: Check the message details. Correct the error and run generator again.

ERR(CAG417): Searching for users failed for filter='*string1*', base='*string2*': *string3*.

Explanation: An error occurred while searching the privileges to be certified.

Action: Check the detailed message. Check your user filter. Check your installation. Remove error condition ad re-run workflow.

ERR(CAG418): Reading next user failed: *string1*. Check your paging policy parameters.

Explanation: An error occurred while requesting the next page of users from the server. Presumably a paging policy timeout has occurred.

Action: Check the detailed message. Increase paging policy timeout. Reduce page size.

WAR(CAG600): No users assigned to *string1* DN=*string2*. No access certification workflow started.

Explanation: A privilege that is prepared for access certification has no users assigned. No workflow started.

Action: Check the preparation process.

WAR(CAG601): Errors occurred in preparation of workflow for *string1* DN=*string2*. No access certification workflow started.

Explanation: The preparation of subject/resource orders failed. No workflow started.

Action: Check the preceding errors.

WAR(CAG602): No direct users assigned to *string1* DN=*string2*. No access certification workflow started for *string3* indirect assignments.

Explanation: A privilege that is prepared for access certification has only users via BOInheritance and RuleProcessing assigned. No workflow started.

Action: You may set the <privileges>.process.allindirect flag in the INI file to create workflows containing only indirect assignments.

WAR(CAG603): User *string1* has more than one value for grouping attribute *string2*. User will be contained in multiple certification workflows.

Explanation: Grouping of users by a link attribute is configured. The denoted user has more than one value for the attribute. Thus he will be part of multiple certification workflows.

Action: Check your configuration. Check the user's grouping attribute.

FTL(CAG901): Usage: Generator -c Configuration_File [options].

Explanation: The program was called with the wrong number of arguments.

Action: unlikely to occur. Call the campaign generator with correct number of arguments.

FTL(CAG902): Configuration file '*string1*' does not exist.

Explanation: The configuration file passed to the analyzer does not exist.

Action: Correct the file's pathname or create a new configuration file at the given pathname.

FTL(CAG903): Error(s) while processing configuration file '*string1*': *string2*.

Explanation: Error(s) occurred while processing the configuration file.

Action: Check the details, correct errors and re-run checker.

FTL(CAG907): No trace file defined in the agent.ini file.

Explanation: The definition for trace.filename is missing.

Action: Add a line trace.filename=pathname in logANT.ini where 'pathname' is the absolute pathname to the trace file.

FTL(CAG908): Cannot open trace file '*string1*' for writing: *string2*.

Explanation: The trace file defined in the agent's ini file cannot be opened for writing.

Action: Check pathname and create missing directories. Check file creation ACLs.

FTL(CAG909): Mandatory property '*string1*' is missing in the LogANT configuration file.

Explanation: A mandatory property is missing in the LogAnt ini file.

Action: Edit the configuration file, add missing properties.

FTL(CAG910): Illegal format *string1*='*string2*' in config file. Format 'P0Y0M0DT0H0M0S' expected.

Explanation: A time interval is defined in wrong format.

Action: Correct the format. Restart workflow.

FTL(CAG911): Initializing LDAP session failed: *string1*.

Explanation: Could not initialize session.

Action: Check the message details.

FTL(CAG912): Bind to LDAP server failed. Check your credentials.

Explanation: Bind failed.

Action: Check technical user / password.

FTL(CAG913): Authentication of the campaign initiator *string1* failed: *string2*.

Explanation: Bind to the campaign initiator failed.

Action: Check effective user / password.

16.7.4. Logging Messages (CHM)

This topic describes the logging messages returned by the Change management.

ERR(CHM200): Assignment of privilege DN=*string2* to subject DN=*string1* is not applied since it has to be approved in an own ticket.

Explanation: None.

Action: None.

ERR(CHM500): Cannot resolve attribute approval workflows to start: Searching for attribute policies failed: *string1*.

Explanation: The attribute policy cache cannot be read from the data store. No attribute approval workflows are started. Save of the object is aborted.

Action: Check server and network connection.

ERR(CHM501): Calculating approvals for junior role assignments of role DN=*string1* failed: *string2*.

Explanation: The junior role assignment approvals for the role could not be calculated.

Action: Check the message details. Correct the error and repeat the assignment.

ERR(CHM502): Calculating approvals for permission assignments of role DN=*string1* failed: *string2*.

Explanation: The permission assignment approvals for the role could not be calculated.

Action: Check the message details. Correct the error and repeat the assignment.

ERR(CHM503): Calculating approvals for group assignments of permission DN=*string1* failed: *string2*.

Explanation: The group assignments to be approved of a permission could not be calculated.

Action: Check the message details. Correct the error and repeat the assignment.

ERR(CHM504): Error while creating order: *string1*.

Explanation: Creating an order failed.

Action: Check detailed message. Check installed jar files, especially dxmOrder.jar. Inform DirX Identity technical support about the problem.

ERR(CHM505): Cannot read SendEvent Policy; LDAP search failed: *string1*.

Explanation: The send event policy cache cannot be read from the data store. Update of the policy is retried after some waiting time.

Action: Check server and network connection.

ERR(CHM506): *string1* when evaluating Send Event policy: *string2*.

Explanation: An exception occurred while the policy cache evaluated the structured attribute dxrEventCfg of the active Event Policies.

Action: Check the content of the event policy attributes.

ERR(CHM507): Cannot read ObjectDelete Policy; LDAP search failed: *string1*.

Explanation: The ObjectDelete policy cache cannot be read from the data store. Update of the policy is retried after some waiting time.

Action: Check server and network connection.

ERR(CHM508): *string1* when evaluating ObjectDelete policy: *string2*.

Explanation: An exception occurred while the policy cache evaluated the structured attribute dxrPolicyCfg of the active ObjectDelete Policies.

Action: Check the content of the object delete policy attributes.

ERR(CHM509): Start of an attribute modification approval workflow failed for DN=*string1*.

Explanation: Start of an approval workflow failed.

Action: Unassign privilege, quit user, and assign privilege again.

ERR(CHM510): Start of an delete object approval workflow failed for DN=*string1*.

Explanation: Start of an approval workflow failed.

Action: Unassign privilege, quit user, and assign privilege again.

ERR(CHM511): Start of an approval workflow to *string1* assignment for resource DN=*string2* to subject DN=*string3* failed.

Explanation: Start of an approval workflow failed.

Action: Unassign privilege, quit user, and assign privilege again.

ERR(CHM512): Remote exception during start of an approval workflow for assignment of privilege DN=*string1* to subject DN=*string2*: *string3*.

Explanation: During start of an approval workflow, a remote exception occurred, indicating an error while communicating with the approval engine.

Action: Check whether the Java based server is running. Check the approval workflow state.

ERR(CHM513): Start of an approval workflow for assignment of privilege DN=*string1* to subject DN=*string2* failed: *string3*.

Explanation: Start of an approval workflow failed.

Action: Unassign privilege, quit user, and assign privilege again. Check log files of IdS-J server.

ERR(CHM514): Remote exception during start of an approval workflow for modifications of subject DN=*string1* failed: *string2*.

Explanation: During start of an approval workflow, a remote exception occurred, indicating an error while communicating with the approval engine.

Action: Check whether the Java based server is running. Check the approval workflow state.

ERR(CHM515): Start of an approval workflow for modification of subject DN=*string1* failed: *string2*.

Explanation: Start of an approval workflow failed.

Action: Quit user, and perform user modifications again.

ERR(CHM516): Cannot cancel request workflow uid=*string1*: *string2*.

Explanation: The request workflow with the given UID cannot be aborted.

Action: Check detailed message. Check workflow state in the workflow monitor. Cancel workflow in Web Center.

ERR(CHM517): No matching *string1* workflow found for subject DN=*string2*.

Explanation: Either no active workflow is applicable for that type of subject and operation. Or you do not have the right to read/execute the workflow.

Action: Call your workflow administrator to create a suitable workflow and/or create the missing policies.

ERR(CHM520): Creating a ticket to delete object by workflow failed for DN=*string1*: *string2*.

Explanation: The ticket to delete an object by a delete workflow could not be created.

Action: Check detailed message.

ERR(CHM521): Creating a ticket to delete object failed for DN=*string1*: *string2*.

Explanation: The ticket to delete an object could not be created.

Action: Check detailed message.

ERR(CHM522): Creating a ticket to add object by workflow failed for DN=*string1*: *string2*.

Explanation: The ticket to add an object by a delete workflow could not be created.

Action: Check detailed message.

ERR(CHM523): Creating a ticket to add object failed for DN=*string1*: *string2*.

Explanation: The ticket to add an object could not be created.

Action: Check detailed message.

ERR(CHM524): Saving temporary object to be created failed for DN=*string1*: Cannot create ticket: *string2*.

Explanation: The ticket to add an object could not be created.

Action: Check detailed message.

ERR(CHM525): Starting workflow to create subject DN=*string1* failed: *string2*.

Explanation: The workflow to add an object could not be started.

Action: Check detailed message.

ERR(CHM526): Remote exception occurred while starting workflow to create subject DN=*string1*: *string2*.

Explanation: The workflow to add an object could not be started.

Action: Check detailed message.

ERR(CHM530): Searching for request workflow instance with cn='*string1*' failed: *string2*.

Explanation: Searching for the workflow instance for the given instance ID failed. The reference from ticket to workflow instance cannot be set.

Action: Check the message details.

ERR(CHM531): Searching for request workflow instance with cn='*string1*' failed: No match.

Explanation: No request workflow found for the given instance ID. The reference from ticket to workflow instance cannot be set.

Action: Check if there are competing processes to delete the workflow instance.

ERR(CHM532): Searching for request workflow instance with cn='*string1*' failed: Result is not unique.

Explanation: More than one request workflow found for the given instance ID. The reference from ticket to workflow instance cannot be set.

Action: Inform DirX Identity technical support about the problem.

ERR(CHM533): Collecting attribute changes by policies failed for DN=*string1*: *string2*.

Explanation: An exception occurred during evaluation of the attribute policies to determine the attribute changes to be approved.

Action: Check the detailed message.

ERR(CHM534): Collecting attribute changes to become valid in future failed for DN=*string1*: *string2*.

Explanation: An exception occurred during evaluation of the attribute changes that should become valid in future.

Action: Check the detailed message.

ERR(CHM535): Instantiation of a template for the order to be created failed for DN=*string1*: *string2*.

Explanation: An exception occurred when trying to create a template order. Ticket creation fails.

Action: Check detailed message. Check installed jar files, especially dxmOrder.jar. Inform DirX Identity technical support about the problem.

ERR(CHM536): An exception occurred when creating the subject order's string representation for DN=*string1*: *string2*.

Explanation: The xml representation of the subject order cannot be created. Ticket creation fails.

Action: Check detailed message. Check installed jar files, especially dxmOrder.jar. Inform DirX Identity technical support about the problem.

ERR(CHM537): An exception occurred when creating the resource order's string

representation for DN=*string1*: *string2*.

Explanation: The xml representation of the resource order cannot be created. Ticket creation fails.

Action: Check detailed message. Check installed jar files, especially dxmOrder.jar. Inform DirX Identity technical support about the problem.

ERR(CHM538): Creation of subject and resource orders from ticket DN=*string1* failed. Check preceding messages.

Explanation: The ticket cannot be processed since errors occurred during processing of the contained orders.

Action: Check preceding messages.

ERR(CHM539): Exception *string1* occurred during creation of subject and resource orders from ticket DN=*string2*: *string3*.

Explanation: The ticket cannot be processed since errors occurred during processing of the contained orders.

Action: Check preceding messages. Check detailed message.

ERR(CHM540): Applying subject and/or resource orders from ticket DN=*string1* failed: *string2*.

Explanation: The ticket cannot be processed since errors occurred when applying the orders.

Action: Check preceding messages. Check detailed message.

ERR(CHM541): Exception *string1* occurred during creation of subject and resource orders from ticket DN=*string2*: *string3*.

Explanation: The ticket cannot be processed since an exception occurred when applying the orders.

Action: Check preceding messages. Check detailed message.

ERR(CHM542): Calculating approvals for role assignments of user DN=*string1* failed: *string2*.

Explanation: The role assignment approvals for the user could not be calculated.

Action: Check the message details. Correct the error and repeat the assignment.

ERR(CHM543): Calculating approvals for permission assignments of user DN=*string1* failed: *string2*.

Explanation: The permission assignment approvals for the user could not be calculated.

Action: Check the message details. Correct the error and repeat the assignment.

ERR(CHM544): Calculating approvals for group assignments of user DN=*string1* failed: *string2*.

Explanation: The group assignment approvals for the user could not be calculated.

Action: Check the message details. Correct the error and repeat the assignment.

ERR(CHM545): Calculating orders for delegation assignments of user DN=*string1* failed: *string2*.

Explanation: The delegation assignment orders for the user could not be calculated.

Action: Check the message details. Correct the error and repeat the assignment.

ERR(CHM546): Calculating orders for granted delegation assignments of user DN=*string1* failed: *string2*.

Explanation: The granted delegation assignment orders for the user could not be calculated.

Action: Check the message details. Correct the error and repeat the assignment.

ERR(CHM547): Calculating orders for ' delegation to' assignments of user DN=*string1* failed: *string2*.

Explanation: The delegation to assignment orders for the user could not be calculated.

Action: Check the message details. Correct the error and repeat the assignment.

ERR(CHM548): Calculating approvals for assignment changes assignments of user DN=*string1* failed: *string2*.

Explanation: The role assignment changes for the user could not be calculated.

Action: Check the message details. Add/remove assignment manually.

ERR(CHM549): Calculating assignment changes for permission assignments of user DN=*string1* failed: *string2*.

Explanation: The permission assignment changes for the user could not be calculated.

Action: Check the message details. Add/remove assignment manually.

ERR(CHM550): Calculating assignment changes for group assignments of user DN=*string1* failed: *string2*.

Explanation: The group assignment changes for the user could not be calculated.

Action: Check the message details. Add/remove assignment manually.

WAR(CHM700): Cannot read privilege DN=*string1* to determine setting of approval flag: *string2*. Approval required assumed.

Explanation: The setting of the privileg’s dxrNeedsApproval flag cannot be read. For security reasons, an approval workflow is started.

Action: Check LDAP server and network connections.

16.7.5. Logging Messages (GID)

This topic describes the logging messages returned by the Siemens global UID generation service.

WAR(GID401): Activity caught a presumable temporary exception: *string1*.

Explanation: Check the detailed messages of the job's run. Activity is retrying.

Action: None.

WAR(GID402): Activity caught a presumable negligible exception: *string1*.

Explanation: Check the detailed messages of the job's run. Activity is retrying.

Action: None.

WAR(GID403): Activity caught a presumable negligible exception: *string1*.

Explanation: Check the detailed messages of the job's run. Activity is searching for the GID.

Action: None.

ERR(GID501): Mandatory parameters for the GidExService are missing: *string1*.

Explanation: Mandatory attributes are surname, givenname, mail, GidExRole, uid, pwd, and requestor.

Action: Correct the values.

ERR(GID502): Search for GidExService returned *string1* hits. Exact one was expected (Note: previously a createGID failed with exception entry already exists).

Explanation: The search found none or more hits then expected. No unique GID can be found.

Action: Check the values.

ERR(GID503): Activity caught an exception: *string1*.

Explanation: Check the detailed messages of the job's run.

Action: Action depends on the detailed messages.

ERR(GID504): Request parameter: *string1*.

Explanation: Check the detailed messages of the job's run.

Action: Action depends on the detailed messages.

16.7.6. Logging Messages (HIS)

This topic describes the logging messages returned by the History Agent.

INF(HIS201): Program '*string1*', Version '*string2*' of '*string3*' started at '*string4*'.

Explanation: Log version and start time.

Action: None.

INF(HIS202): Program '*string1*' terminated at '*string2*' with *string3* fatalError(s), *string4* error(s), *string5* warning(s).

Explanation: End time.

Action: None.

INF(HIS203): Successfully connected to LDAP server '*string1*'; user: '*string2*'.

Explanation: Connection to LDAP server successfully setup.

Action: None.

INF(HIS205): Search operation successful.

Explanation: Search operation successfully performed.

Action: None.

INF(HIS206): Deletion of *string1* history values from object '*string2*' was successful.

Explanation: The audit of the given object was extracted.

Action: None.

INF(HIS208): Unbind operation successful.

Explanation: Unbind operation successfully performed.

Action: None.

INF(HIS209): Statistics: Number of modified objects: *string1*, Number of unsuccessful modifications: *string2*.

Explanation: Statistic information about successful/unsuccessful operation.

Action: None.

INF(HIS210): Could not read the attribute 'keyOwnerPSE' of the object '*string1*'. Signing is not available.

Explanation: The related attribute could not be found. PSE for signing is not available.

Action: Check existence of that attribute.

INF(HIS211): Could not read the attribute 'userCertificate' of the object '*string1*'. Signing is not available.

Explanation: The related attribute could not be found. Certificate for signing is not available.

Action: Check existence of that attribute.

INF(HIS212): Searching for entries with existing audit trail using the search base DN '*string1*', the filter '*string2*', the scope '*string3*' and the size limit *string4*.

Explanation: The audit trail records are being extracted with described parameters.

Action: None.

FTL(HIS301): Error while connecting to LDAP server '*string1*' as user '*string2*'. Error message: '*string3*'.

Explanation: An error has occurred while connecting to the LDAP server.

Action: Check the error reason.

FTL(HIS303): Error while searching entries. Error message: '*string1*'.

Explanation: Searching entries in the LDAP directory failed.

Action: Check the error reason.

FTL(HIS304): Error while writing XML file. Error message: '*string1*'.

Explanation: An error has occurred while writing the history record into the XML file.

Action: Check the error reason.

FTL(HIS305): Error while opening XML file. Error message: '*string1*'.

Explanation: An error has occurred when opening the XML output file.

Action: Check the error reason.

FTL(HIS306): Exception received. Error message: '*string1*'.

Explanation: An exception has been received.

Action: Check the error reason.

FTL(HIS307): File directory could not be created.

Explanation: An error has occurred when creating a directory in the file system.

Action: Check the configuration parameters of the history agent.

FTL(HIS308): Importing private keys failed. Error message: '*string1*'.

Explanation: An error has occurred when importing the private keys (used for decrypting values).

Action: Check the error reason.

FTL(HIS309): Decrypting of '*string1*' failed.

Explanation: An error has occurred while decrypting an attribute value.

Action: Check the error reason.

FTL(HIS310): Unscrambling of '*string1*' failed.

Explanation: An error has occurred while unscrambling an attribute value.

Action: Check the error reason.

FTL(HIS311): Error while renaming file from '*string1*' to '*string2*'. Error message: '*string3*'.

Explanation: An error occurred while renaming the XML output file.

Action: Check the error reason.

ERR(HIS401): Error while modifying object '*string1*'. Error message: '*string2*'.

Explanation: An error has occurred while removing a history attribute value.

Action: Check the error reason.

ERR(HIS403): Error while performing LDAP unbind operation. Error message: '*string1*'.

Explanation: An error has occurred when unbind from the LDAP directory.

Action: Check the error reason.

FTL(HIS404): Error while extracting next entry from search result. Error message: '*string1*'.

Explanation: An error has occurred while extracting the next entry from the LDAP search result.

Action: Check the error reason.

ERR(HIS405): Error while extracting a XML audit record from search result with DN '*string1*'. Error message: '*string2*'.

Explanation: An error has occurred while extracting a XML audit record from the LDAP search result.

Action: Check the error reason.

ERR(HIS406): Error while connecting to Identity store. Could not get keys for signing. Signing is not available.

Explanation: The agent could not connect to the Identity store and get keys for signing.

Action: Check the bind credentials and configuration file.

ERR(HIS407): Error while retrieving certificate or PSE for signing from Identity store in domain '*string1*'. Signing is not available.

Explanation: The agent could not retrieve a valid certificate or PSE from Identity store.

Action: Check appropriate attributes for the related user.

ERR(HIS408): Error while retrieving certificate or PSE for signing from Identity store in domain '*string1*', null values detected. Signing is not available.

Explanation: The agent could not retrieve a valid certificate or PSE from Identity store.

Action: Check appropriate attributes for the related user.

ERR(HIS409): The agent could not resolve the environment variable 'DIRXIDENTITY_INST_PATH'.

Explanation: The agent could not get the environment variable 'DIRXIDENTITY_INST_PATH', the DirX Identity installation path after installation process.

Action: Check existence of that variable, define it correctly if needed.

ERR(HIS410): The agent could not read the file '*string1*', an IOException occurred:'*string1*'. Signing is not available.

Explanation: The specified file could not be read, PIN for PSE is not available.

Action: Check existence and readability of that file.

ERR(HIS411): Could not initialize crypt support for decrypting PIN for signing key. Signing is not available.

Explanation: An internal error has occurred, crypt support for decrypting PIN is not available.

Action: Contact DirX Identity support.

ERR(HIS412): Could not get the value of the 'signaturePin' property from the file '*string1*'. Signing is not available.

Explanation: The value of the 'signaturePin' is probably not defined.

Action: Check existence of the given property in the related property file.

ERR(HIS413): Could not decrypt the value of the 'signaturePin' from the property file '*string1*'. Signing is not available.

Explanation: The value of the 'signaturePin' is probably not defined or damaged.

Action: Check existence of the given property in the related property file.

ERR(HIS414): Could not read the object '*string1*', an Exception occurred: '*string2*'. Signing is not available.

Explanation: The related object could not be found.

Action: Check existence of that object, check the error reason.

ERR(HIS417): Could not create a valid certificate from the attribute 'userCertificate' of the object '*string1*'. Signing is not available.

Explanation: The related attribute is not valid. Certificate for signing is not available.

Action: Check existence and correctness of that attribute.

ERR(HIS418): Could not get PIN for PSE. Signing is not available.

Explanation: No PIN for PSE is available.

Action: Check the previously logged errors.

ERR(HIS419): Could not add PSE and PIN for signing to crypt support:'*string1*'. Signing is not available.

Explanation: Crypt support for signing could not be initialized.

Action: Check the error reason.

ERR(HIS420): Could not find LDAP entry '*string1*'. Signing is not available.

Explanation: The LDAP entry with attributes 'keyOwnerPSE' and 'userCertificate' was not found.

Action: Check the existence of that user.

ERR(HIS421): Could not extract the domain name from the bind credentials. Signing is not available.

Explanation: The domain name could not be resolved from the user name.

Action: Contact DirX Identity support.

ERR(HIS422): Processing of history record failed for DN=*string1*: No values signed/extracted.

Explanation: A history record was extracted from the object that could not be processed by the agent.

Action: Check certificated and privatized keys for signature. Verify that only one PSE is present, being consistent to the certificate for signing.

ERR(HIS423): Processing of history record failed for DN=*string1*: *string2*.

Explanation: A history record was extracted from the object that could not be processed by the agent.

Action: Check certificated and privatized keys for signature. Verify that only one PSE is present, being consistent to the certificate for signing.

ERR(HIS424): Could not find "password.properties" in an IDS-J server folder for the domain '*string1*'. Signing is not available.

Explanation: There is no file "password.properties" for the given domain in the IDS-J server subdirectories.

Action: Check the installation folder of the IDS-J server. If you don’t have an IDS-J server on your machine, you should create the subfolder '..ids-j-<domain-name>-S<server number>private' manually and copy 'password.properties' from the platform where the IDS-J server is running.

ERR(HIS425): Processing of history record failed for DN=*string1*: value=*string2* due to unmarhalling problems; nevertheless the value will be deleted in LDAP.

Explanation: A history record could not be unmarshalled (eventually due to invalid XML characters). The record will not be written to the ouput file, but will be deleted in LDAP.

Action: Check the history value.

16.7.7. Logging Messages (LAN)

This topic describes the logging messages returned by the DirX Identity log file analyzer.

INF(LAN201): Program '*string1*', Version '*string2*' of '*string3*' started ***

Explanation: Start message of the log file analyzer.

Action: None.

INF(LAN202): Program '*string1*' ended with *string2* error(s) and *string3* warning(s).

Explanation: Informs the user that the program has ended correctly. Shows an error statistics.

Action: None.

INF(LAN210): Processing logfile '*string1*'.

Explanation: Informs the user which file is being processed.

Action: None.

INF(LAN211): *string1* workflow instances from *string2* logfiles processed.

Explanation: Shows a statistics on the processed objects.

Action: None.

INF(LAN212): Output folder '*string1*' successfully removed.

Explanation: Inform the the output folder has been successfully cleared prior to processing.

Action: None.

ERR(LAN400): Cannot create filename for workflow instance uid='*string1*': Format error in start time '*string2*': *string3*.

Explanation: A format error has been detected preventing to create the correct output file name. The file is created under the name <uid>.log.

Action: Check if the input file is corrupt or has the wrong time format.

ERR(LAN401): Cannot append log information to existing file '*string1*': No write permission.

Explanation: Cannot append log information to an already existing file.

Action: Check the file access rights.

ERR(LAN402): Error while opening file '*string1*' for append: *string2*.

Explanation: Cannot append log information to an already existing file.

Action: Check the detailed message.

ERR(LAN403): Error while opening new file '*string1*': *string2*.

Explanation: Cannot append log information to an already existing file.

Action: Check the detailed message.

ERR(LAN404): An error occurred while writing the workflow instance files for input file '*string1*': *string2*.

Explanation: An unexpected error occurred.

Action: Check the detailed message.

ERR(LAN405): Error reading line '*string1*' of file '*string2*': *string3*.

Explanation: An error occurred during read of a log file. Processing of this file is aborted.

Action: Check the detailed message.

ERR(LAN406): Error processing line '*string1*' of file '*string2*': *string3*.

Explanation: An error occurred during processing of one logfile line. Processing of this file is aborted.

Action: Check the detailed message.

ERR(LAN407): Cannot open logfile *string1* for reading: *string2*. Skipping file.

Explanation: A logfile cannot be opened for reading. Processing of this file is skipped.

Action: Check protections and file path. Check the detailed message.

ERR(LAN408): Cannot open statistics file *string1* for writing: *string2*.

Explanation: The statistics cannot be opened for writing. Statistics information is lost.

Action: Check protections and file path. Check the detailed message.

ERR(LAN409): Error during write of statistics file *string1*: *string2*.

Explanation: The statistics file cannot be written completely. Statistics information is lost.

Action: Check protections and disk coverage. Check the detailed message.

ERR(LAN410): Error during read of statistics file *string1*: *string2*.

Explanation: The statistics file cannot be read completely. Statistics information is lost.

Action: Check protections. Check the detailed message.

ERR(LAN411): Error while calculating elapsed time from '*string1*' to '*string2*': *string3*.

Explanation: The elapsed time cannot be calculated.

Action: Check your date format. Configure a date format in the configuration file.

ERR(LAN412): Could not delete file '*string1*'.

Explanation: A file could not be deleted.

Action: Check if the file is used by another application. Delete file manually.

ERR(LAN413): Could not delete folder '*string1*'.

Explanation: A folder could not be deleted.

Action: Check if the folder contains a file that could not be deleted. Delete folder manually.

ERR(LAN414): Failed to delete incomplete file '*string1*'.

Explanation: An incomplete file could not be deleted.

Action: Check if the file is used by another application. Delete file manually.

ERR(LAN415): Failed to rename file '*string1*' to '*string2*'.

Explanation: An file could not be renamed.

Action: Check if the file is used by another application. Rename file manually.

ERR(LAN416): Error while writing data to file '*string1*': *string2*.

Explanation: An error occurred writing data into an output file. Processing of this file is aborted.

Action: Check the detailed message.

FTL(LAN901): Usage: LogANT <Configuration File>.

Explanation: The program was called with the wrong number of arguments.

Action: unlikely to occur. Call the checker with correct number of arguments.

FTL(LAN902): Configuration file '*string1*' does not exist.

Explanation: The configuration file passed to the analyzer does not exist.

Action: Correct the file's pathname or create a new configuration file at the given pathname.

FTL(LAN903): Error(s) while processing configuration file '*string1*': *string2*.

Explanation: Error(s) occurred while processing the configuration file.

Action: Check the details, correct errors and re-run checker.

FTL(LAN907): No trace file defined in the agent.ini file.

Explanation: The definition for trace.filename is missing.

Action: Add a line trace.filename=<pathname> in logANT.ini where <pathname is the absolute pathname to the trace file.

FTL(LAN908): Cannot open trace file '*string1*' for writing: *string2*.

Explanation: The trace file defined in the agent's ini file cannot be opened for writing.

Action: Check pathname and create missing directories. Check file creation ACLs.

FTL(LAN909): Mandatory property '*string1*' is missing in the LogANT configuration file.

Explanation: A mandatory property is missing in the LogAnt ini file.

Action: Edit the configuration file, add missing properties.

FTL(LAN910): The configured IDS-J log container '*string1*' does not exist or is not readable.

Explanation: The property inputPath points to a location that does not exist or is not a readable directory.

Action: Correct the inputPath, change file protections.

FTL(LAN911): The configured IDS-J log container '*string1*' does not contain any server log files for processing.

Explanation: No IDS-J server logfiles exist under the configured folder.

Action: Correct the inputPath definition.

FTL(LAN912): Could not remove output folder '*string1*'.

Explanation: The output folder could not be deleted.

Action: Check if the folder contains a file that could not be deleted. Delete folder manually.

FTL(LAN913): Wrong number of output files names defined in 'outputPath'. Available number: '*string1*', required number: '*string2*'.

Explanation: The number of output files names must be the same as in the definition of 'inputPath'.

Action: Correct the definition of 'outputPath'.

FTL(LAN914): Wrong number of statistics files names defined in 'statisticsFileName'. Available number: '*string1*', required number: '*string2*'.

Explanation: The number of statistics files names must be the same as in the definition of 'inputPath'.

Action: Correct the definition of 'statisticsFileName'.

FTL(LAN915): Wrong time stamp format: '*string1*', error: '*string2*'.

Explanation: The time stamp string can’t be evaluated correctly.

Action: Check the server log file for incorrect time stamps.

16.7.8. Logging Messages (LMT)

This topic describes the logging messages returned by the DirX Identity log file merging tool.

INF(LMT201): Program '*string1*', Version '*string2*' of '*string3*' started ***

Explanation: Start message of the log file analyzer.

Action: None.

INF(LMT202): Program '*string1*' ended with *string2* error(s) and *string3* warning(s).

Explanation: Informs the user that the program has ended correctly. Shows an error statistics.

Action: None.

INF(LMT213): The configured input path container '*string1*' does not contain any log files for processing.

Explanation: No log files exist under the configured folder.

Action: Correct the inputPath definition.

ERR(LMT403): Error while opening new file '*string1*': *string2*.

Explanation: Cannot append log information to an already existing file.

Action: Check the detailed message.

ERR(LMT405): Error reading line '*string1*' of file '*string2*': *string3*.

Explanation: An error occurred during read of a log file. Processing of this file is aborted.

Action: Check the detailed message.

ERR(LMT406): Error processing line '*string1*' of file '*string2*': *string3*.

Explanation: An error occurred during processing of one logfile line. Processing of this file is aborted.

Action: Check the detailed message.

ERR(LMT407): Cannot open logfile *string1* for reading: *string2*. Skipping file.

Explanation: A logfile cannot be opened for reading. Processing of this file is skipped.

Action: Check protections and file path. Check the detailed message.

ERR(LMT408): Cannot open statistics file *string1* for writing: *string2*.

Explanation: The statistics cannot be opened for writing. Statistics information is lost.

Action: Check protections and file path. Check the detailed message.

ERR(LMT409): Error during write of statistics file *string1*: *string2*.

Explanation: The statistics file cannot be written completely. Statistics information is lost.

Action: Check protections and disk coverage. Check the detailed message.

ERR(LMT411): Error while calculating elapsed time from '*string1*' to '*string2*': *string3*.

Explanation: The elapsed time cannot be calculated.

Action: Check your date format. Configure a date format in the configuration file.

ERR(LMT412): Could not delete file '*string1*'.

Explanation: A file could not be deleted.

Action: Check if the file is used by another application. Delete file manually.

ERR(LMT413): Could not delete folder '*string1*'.

Explanation: A folder could not be deleted.

Action: Check if the folder contains a file that could not be deleted. Delete folder manually.

ERR(LMT415): Failed to copy file '*string1*' to '*string2*'; error: *string3*.

Explanation: An file could not be copy.

Action: Check if the file is used by another application. Copy file manually.

ERR(LMT416): Error while writing data to file '*string1*': *string2*.

Explanation: An error occurred writing data into an output file. Processing of this file is aborted.

Action: Check the detailed message.

FTL(LMT901): Usage: LogMerge <Configuration File>.

Explanation: The program was called with the wrong number of arguments.

Action: unlikely to occur. Call the checker with correct number of arguments.

FTL(LMT902): Configuration file '*string1*' does not exist.

Explanation: The configuration file passed to the analyzer does not exist.

Action: Correct the file's pathname or create a new configuration file at the given pathname.

FTL(LMT903): Error(s) while processing configuration file '*string1*': *string2*.

Explanation: Error(s) occurred while processing the configuration file.

Action: Check the details, correct errors and re-run checker.

FTL(LMT907): No trace file defined in the agent.ini file.

Explanation: The definition for trace.filename is missing.

Action: Add a line trace.filename=<pathname> in logMerge.ini where <pathname is the absolute pathname to the trace file.

FTL(LMT908): Cannot open trace file '*string1*' for writing: *string2*.

Explanation: The trace file defined in the agent's ini file cannot be opened for writing.

Action: Check pathname and create missing directories. Check file creation ACLs.

FTL(LMT909): Mandatory property '*string1*' is missing in the LogMerge configuration file.

Explanation: A mandatory property is missing in the logMerge.ini file.

Action: Edit the configuration file, add missing properties.

FTL(LMT910): The configured input path container '*string1*' does not exist or is not readable.

Explanation: The property inputPath points to a location that does not exist or is not a readable directory.

Action: Correct the inputPath, change file protections.

16.7.9. Logging Messages (LNC)

This topic describes the logging messages returned by the DirX Identity link checker agent.

INF(LNC201): Program '*string1*', Version '*string2*' of '*string3*' started ***.

Explanation: Start message of the link checker agent.

Action: None.

INF(LNC202): Program '*string1*' ended with *string2* fatalError(s), *string3* error(s), and *string4* warning(s).

Explanation: Informs the user that the program has ended correctly. Shows an error statistics.

Action: None.

INF(LNC210): Number of referenced map items: *string1*.

Explanation: Shows the number of referenced map items.

Action: None.

INF(LNC212): *string1* unreferenced map items detected: *string2* unreferenced map items removed.

Explanation: The link checker reports the number of unreferenced and cleaned up map items.

Action: None.

ERR(LNC400): Link *string1* at *string2* DN=*string3* is broken: *string4*.

Explanation: A broken link has been detected.

Action: Correct link manually or run LinkChecker in cleanup mode.

ERR(LNC401): Save of *string1* DN=*string2* failed: *string3*. Cleanup not possible.

Explanation: Object could not be cleaned up due to save error.

Action: Correct broken link manually.

ERR(LNC402): LDAP error occurred during check of DN=*string1* for attribute *string2*, value=*string3*: *string4*.

Explanation: An unexpected LDAP-Error occurred while trying to read object from link.

Action: Correct broken link manually.

ERR(LNC403): An error occurred during check of DN=*string1* for attribute *string2*, value=*string3*: *string4*.

Explanation: An unexpected exception occurred while trying to read object from link.

Action: Check message details. Correct broken link manually.

ERR(LNC404): Paged search failed for objectClass *string1*: *string2*.

Explanation: The search run on an error for objects of the specified class.

Action: Check server and network connection. Check for typing errors in the configuration (wrong object class?).

ERR(LNC405): Error reading next page for objectClass *string1*: *string2*.

Explanation: Requesting the next page run on an error.

Action: Check detailed message. Increase paging policy timeout.

ERR(LNC407): Search of mapping scripts failed: *string1*. Cleanup of MapItems aborted.

Explanation: An error occurred during search of the mapping scripts.

Action: Check the detailed error message. Correct mapping manually.

ERR(LNC408): MapItem DN=*string1* is not referenced by any mapping script.

Explanation: The mapitem is not used in any mapping script. It is being deleted in cleanup mode.

Action: Run LinkChecker in cleanup mode to remove the unused map item.

ERR(LNC409): Cannot remove MapItem DN=*string1*: *string2*.

Explanation: Cleaning up an unused mapitem failed.

Action: Check the detailed message, correct the error and re-run the checker.

ERR(LNC410): *string1* unreferenced map items detected: *string2* unreferenced map items removed.

Explanation: The link checker reports the number of unreferenced and cleaned up map items.

Action: None.

ERR(LNC411): Mapping script references no map items: DN=*string1*.

Explanation: A mapping script was found that does not use any map items.

Action: Check the script.

ERR(LNC412): Removal of *string1* DN=*string2* failed: *string3*. Cleanup not possible.

Explanation: To cleanup an assignment object it has to be removed. Removal failed.

Action: Remove object in the data view manually.

FTL(LNC901): Usage: LinkChecker <Configuration File>.

Explanation: The program was called with the wrong number of arguments.

Action: unlikely to occur. Call the checker with correct number of arguments.

FTL(LNC902): Configuration file '*string1*' does not exist.

Explanation: The configuration file passed to the agent does not exist.

Action: Correct the file's pathname or create a new configuration file at the given pathname.

FTL(LNC903): Error(s) while processing configuration file '*string1*': *string2*.

Explanation: Error(s) occurred while processing the configuration file.

Action: Check the details, correct errors and re-run checker.

FTL(LNC905): Cannot initialize storage: *string1*.

Explanation: initialize of storage failed.

Action: Check the details, correct errors and re-run checker.

FTL(LNC907): No trace file defined in the agent.ini file.

Explanation: The definition for trace.filename is missing.

Action: Add a line trace.filename=<pathname> in agent.ini where <pathname is the absolute pathname to the trace file.

FTL(LNC908): Cannot open trace file '*string1*' for writing: *string2*.

Explanation: The trace file defined in the agent's ini file cannot be opened for writing.

Action: Check pathname and create missing directories. Check file creation ACLs.

16.7.10. Logging Messages (MCO)

This topic describes the logging messages returned by the Mail Connector for DirX Identity Connector Framework.

INF(MCO225): Connected to Smtp Server *string1* with port *string2* and user *string3* successfully.

Explanation: The connection to the smtp server has been established successfully.

Action: None.

INF(MCO231): Sent mail for entry: *string1* successfully.

Explanation: The current entry has been mailed successfully.

Action: None.

INF(MCO232): Encrypting mail for entry: *string1*.

Explanation: The current entry has been encrypted successfully.

Action: None.

INF(MCO233): Signing mail for entry: *string1*.

Explanation: The current entry has been signed successfully.

Action: None.

WAR(MCO410): Add entry *string1* failed: *string2*.

Explanation: Adding the entry failed.

Action: Check the error code.

WAR(MCO413): Cannot access version information from jar file.

Explanation: Accessing version information from manifest failed.

Action: Inform DirX Identity technical support about the problem.

WAR(MCO414): SMS message is too long.

Explanation: SMS message is too long. Must be less than 160 characters.

Action: Shorten the SMS message to less than 160 characters.

WAR(MCO415): The recipient: *string1* is not a valid mail address and not passed to the Smtp server.

Explanation: This recipient is not passed to the Smtp server.

Action: Check your configuration.

ERR(MCO570): Sending entry id:_string1_ failed: *string2*.

Explanation: Sending the entry failed.

Action: Check the error code.

ERR(MCO571): Modify requests are not supported by this connector.

Explanation: Modify requests are not supported by this connector.

Action: To send a mail submit an add request to the mail connector.

ERR(MCO572): Delete requests are not supported by this connector.

Explanation: Delete requests are not supported by this connector.

Action: To send a mail submit an add request to the mail connector.

ERR(MCO573): Search requests are not supported by this connector.

Explanation: Search requests are not supported by this connector.

Action: To send a mail submit an add request to the mail connector.

ERR(MCO574): No from attribute specified in request. Cannot send mail.

Explanation: An attribute named "from" in the request is mandatory.

Action: Add this attribute to the request with the value of the mail sender.

ERR(MCO575): No valid recipient specified in request. Cannot send mail.

Explanation: An attribute named "to" in the request is mandatory and must contain at least one valid mail address.

Action: Add this attribute to the request with the value(s) of the mail recipient(s).

ERR(MCO576): The following email addresses are invalid: *string1*

Explanation: Addresses to which this message could not be sent.

Action: Check the spelling of the email address.

ERR(MCO577): The message was successfully sent to the following email addresses: *string1*

Explanation: Addresses to which this message was sent successfully before the error occurred.

Action: None.

ERR(MCO578): The message could not be sent to the following email addresses: *string1*

Explanation: Addresses that are valid but to which this message was not sent.

Action: Check the log or inform the recipient about problems with the mailbox.

FTL(MCO910): Cannot determine address of smtp server, no connection defined for the connector.

Explanation: Add a <connection> tag to the definition of the connector.

Action: Correct the configuration file as described in explanation.

FTL(MCO911): Cannot bind to smtp server, no server defined in connection.

Explanation: Add a server="<server>" property to the connection tag with <server>

containing the name of the smtp server.

Action: Correct the configuration file as described in explanation.

FTL(MCO912): Cannot bind to smtp server, no user defined in connection.

Explanation: Add a user="<user>" property to the connection tag with <user> containing the user with the appropriate rights.

Action: Correct the configuration file as described in explanation.

FTL(MCO913): Cannot bind to smtp server, no password defined in connection.

Explanation: Add a password="<password>" property to the connection tag with <password> containing the users password.

Action: Correct the configuration file as described in explanation.

FTL(MCO914): Bind to the smtp server *string1* with port *string2* and user *string3* failed: *string4*.

Explanation: Binding to the smtp server failed.

Action: Check the connection parameters and if server is running.

FTL(MCO915): Disconnect from smtp server failed: *string1*.

Explanation: Disconnecting from the smtp server failed.

Action: Check the error code.

FTL(MCO916): Session to smtp server is lost or open method of Connector was not called.

Explanation: Check if open method of connector has been called.

Action: Check if server is running.

FTL(MCO917): Exception while accessing version information from jar file.

Explanation: Accessing version information from manifest failed.

Action: Inform DirX Identity technical support about the problem.

FTL(MCO933): Exception occurred with exception message: *string1*.

Explanation: An exception occurred with the given exception message.

Action: Check the exception message text.

16.7.11. Logging Messages (POA)

This topic describes the logging messages returned by the DirX Identity policy agent. Uses the rule processor to apply provisioning, consistency, and validation rules.

INF(POA201): Program '*string1*', Version '*string2*' of '*string3*' started ***.

Explanation: The policy agent has started.

Action: None.

INF(POA202): Program '*string1*' ended with *string2* fatalError(s), *string3* error(s), and *string4* warning(s).

Explanation: The policy agent has ended.

Action: None.

INF(POA203): Running in SIMULATE mode. Modifications are reported but not applied.

Explanation: Modifications are reported but not applied.

Action: None.

INF(POA204): Running in PROCESS mode. Modifications are applied.

Explanation: Modifications are applied.

Action: None.

INF(POA220): Rules processed in *string1* msec.

Explanation: The rules were processed in the given time.

Action: None.

INF(POA230): SIMULATE: Inserting account DN=*string1* to group DN=*string2* with state=IMPORTED.

Explanation: The rule processor runs in simulate mode. In real mode it would add the account to the group.

Action: None.

INF(POA231): SIMULATE: Deleting member DN=*string1* from group DN=*string2*.

Explanation: The rule processor runs in simulate mode. In real mode it would delete the account from the group.

Action: None.

INF(POA232): SIMULATE: Inserting account DN=*string1* to group DN=*string2* with state=DELETE.

Explanation: The rule processor runs in simulate mode. In real mode it would add the account to the group.

Action: None.

INF(POA240): Running in simulation mode, no changes applied to group DN=*string1*.

Explanation: The rule processor runs in simulate mode. Thus, the group under consideration is not modified.

Action: None.

INF(POA250): SIMULATE: Removing object DN=*string1* from the datastore.

Explanation: The rule processor runs in simulate mode. Thus, the object under consideration is not deleted.

Action: None.

INF(POA251): Object has been removed from the datastore: DN=*string1*.

Explanation: An object has been physically deleted by the rule processor.

Action: None.

INF(POA252): The object's internal states have been set to DELETED: DN=*string1*.

Explanation: The object's state has been set to DELETED.

Action: None.

INF(POA253): User DN=*string1* migrated for SoD checks: *string2* privileges added to dxrPrivilegesGrantedLink.

Explanation: The user has been successfully migrated for SoD checks.

Action: None.

INF(POA254): User DN=*string1* is ready for SoD checks: No privileges are assigned to user.

Explanation: The user has no privileges assigned. Thus, no SoD violations may occur at the user.

Action: None.

INF(POA255): SIMULATE: User DN=*string1* migrated for SoD checks: *string2* privileges added to dxrPrivilegesGrantedLink.

Explanation: The rule processor runs in simulate mode. The user has been successfully migrated for SoD checks but the changes are not applied.

Action: None.

INF(POA256): *string1*

Explanation: Generic message.

Action: None.

INF(POA306): Inserting account DN=*string1* to group DN=*string2* with state=IMPORTED.

Explanation: None.

Action: None.

INF(POA307): Deleting member DN=*string1* from group DN=*string2*.

Explanation: None.

Action: None.

INF(POA309): GrantPrivilege called for user DN=*string1*, privilege DN=*string2*.

Explanation: None.

Action: None.

INF(POA310): DenyPrivilege called for user DN=*string1*, privilege DN=*string2*.

Explanation: None.

Action: None.

INF(POA311): GrantPrivilege called for user DN=*string1*.

Explanation: None.

Action: None.

INF(POA312): DenyPrivilege called for user DN=*string1*.

Explanation: None.

Action: None.

INF(POA313): Removing privilege DN=*string1*.

Explanation: None.

Action: None.

INF(POA314): Validating group DN=*string1*.

Explanation: None.

Action: None.

INF(POA315): Inserting account DN=*string1* to group DN=*string2* with state=DELETE.

Explanation: account is set to DELETE in Group because it has lost the rights for this group.

Action: None.

INF(POA326): Imported account DN=*string1* is not affected by Accept/Cleanup rules.

Explanation: The policy engine searches for all IMPORTED memberships. For this account no accept or cleanup rules were found that would change the state to either IGNORE or DELETED.

Action: Set up the relevant rules so that this membership is affected or handle the membership manually. See the Provisioning Admin Guide (section Managing Group Memberships) for more information.

INF(POA328): Imported account DN=*string1* is accepted by rule DN=*string2*, priority=*string3*.

Explanation: None.

Action: None.

INF(POA329): Imported account DN=*string1* is cleaned up by rule DN=*string2*, priority=*string3*.

Explanation: None.

Action: None.

INF(POA330): Assignment to imported member *string1* is accepted by rule DN=*string2*, priority=*string3*.

Explanation: None.

Action: None.

INF(POA331): Assignment to imported member *string1* is cleaned up by rule DN=*string2*, priority=*string3*.

Explanation: None.

Action: None.

INF(POA332): Running in simulation mode, omitting changes for object DN=*string1*.

Explanation: The rule processor runs in simulate mode. Thus, the object is not modified.

Action: None.

INF(POA333): Running in process mode, applying changes to object DN=*string1*.

Explanation: The rule processor runs in process mode. Thus, the object is modified.

Action: None.

INF(POA334): The attribute *string1* contains an old value *string2* for account DN=*string3*. Value will be removed.

Explanation: The account password history contains an old value and will be removed.

Action: None.

INF(POA335): The attribute *string1* contained only old values for account DN=*string2*. All *string3* values will be removed.

Explanation: The account password history contained only old values and all of them will be removed.

Action: None.

INF(POA336): The attribute *string1* contained some old values for account DN=*string2*. *string3* values will be removed, *string4* will be kept.

Explanation: The account password history contained some old values and all of them will be removed.

Action: None.

INF(POA400): Segregation of duty checks are enabled, but role resolution is disabled. Enabling role resolution.

Explanation: Role resolution is a precondition for segregation of duty checks. Thus, role resolution is enabled, despite it is disabled by the setting of the assignOnly option.

Action: Set assignment mode to assignAndResolve or disable dxrSodActive at the domain root.

WAR(POA500): Cannot register granting rule, originating rule not found.

Explanation: The granting rule could not be retrieved. Thus, it is not registered in the user assignment details.

Action: None.

WAR(POA501): Skipping subject DN=*string1*: Subject is not a DirX Identity group.

Explanation: The subject in validate or accept/cleanup privilege is expected to be a DirX Identity group. Other objects are not processed.

Action: Check and correct OnValidation filter. Check if object descriptions are correctly loaded during startup.

WAR(POA502): ---------- rule >*string1*< has more than one filter, using first one !!!

Explanation: There is more than one account filter defined for a validate rule. Only the first one is processed.

Action: Remove all but one filter(s).

WAR(POA503): Cannot get dxrOnvalidation from group DN=*string1*.

Explanation: While validating a group, no OnValidation can be retrieved.

Action: Add validation filter.

WAR(POA504): Rule *string1* has no effect for group DN=*string2*.

Explanation: A validation rule has no effect on a group.

Action: Check and correct validation filter.

WAR(POA505): User DN=*string1* already has an assignment to privilege DN=*string2* being LOCKED. Privilege is not assigned.

Explanation: A new assignment is done after removal of the lock since locking indicates an approval.

Action: None.

WAR(POA506): Cannot write rule data to whyDetails, originating rule not found.

Explanation: The granting/denying rule could not be retrieved. Thus, it is not written to

the why/details in the audit record.

Action: None.

WAR(POA507): The time value *string1* extracted from attribute *string2* and account DN=*string3*. Could not be parsed. Omitting it.

Explanation: The value contained an invalid time. The value could not be parsed properly.

Action: Check the relevant value and correct its syntax. Run consistency rule again.

WAR(POA508): The value *string1* of attribute *string2* and account DN=*string3*. Has improper count of string tokens. Omitting it.

Explanation: The value is of an invalid format. The value could not be parsed properly.

Action: Check the relevant value and correct its syntax. Run consistency rule again.

ERR(POA700): Error while reading privilege DN=*string1*: *string2*.

Explanation: Error(s) occurred while reading privilege.

Action: Check details. Correct broken links.

ERR(POA701): Error while processing validate rule *string1*: Searching for accounts failed: Object.toString=*string2*.

Explanation: Trying to retrieve an account matching validation rule from search result failed.

Action: Check the message details to correct the error.

ERR(POA702): Error while processing validate rule *string1*: Searching for accounts failed: Got null value for object.

Explanation: Trying to retrieve an account matching validation rule from search result failed.

Action: Check the message details to correct the error.

ERR(POA703): Error while processing validate rule *string1*: Searching for accounts failed: Object DN=*string2* is not a DirX Identity account.

Explanation: The search returned an object being not a DirX Identity account.

Action: Check search filters/object descriptions.

ERR(POA704): Error while processing validate rule *string1*: Searching for accounts failed: Got null value for storageObject.

Explanation: Trying to retrieve an account matching validation rule from search result failed.

Action: Check the message details to correct the error.

ERR(POA705): Error while processing role *string1*: Searching for accounts failed: *string2*.

Explanation: An error occurred while searching accounts.

Action: Check the message details and correct the error.

ERR(POA706): Error while processing validate rule *string1*: Searching for obligations failed: Object.toString=*string2*.

Explanation: Trying to retrieve an obligation from search result failed.

Action: Check the message details to correct the error.

ERR(POA707): Error while processing validate rule *string1*: Searching for obligations failed: Got null value for object.

Explanation: Trying to retrieve an obligation from search result failed.

Action: Check the message details to correct the error.

ERR(POA708): Error while processing validate rule *string1*: Searching for obligations failed: Object DN=*string2* is not a DirX Identity obligation.

Explanation: The search returned an object being not a DirX Identity obligation.

Action: Check search filters/object descriptions.

ERR(POA709): Error while processing validate rule *string1*: Searching for obligations failed: Got null value for storageObject.

Explanation: Trying to retrieve an obligation from search result failed.

Action: Check the message details to correct the error.

ERR(POA710): Error while processing role *string1*: Searching for obligations failed: *string2*.

Explanation: An error occurred while searching obligations.

Action: Check the message details to correct the error.

ERR(POA711): Inserting account DN=*string1* to group DN=*string2* failed: *string3*.

Explanation: Creating an account to group assignment during validation failed.

Action: Check the message details to correct the error.

ERR(POA712): Error(s) occurred while searching imported accounts for group DN=*string1*: *string2*.

Explanation: The search for imported accounts of a group run into an error.

Action: Check the message details to correct the error.

ERR(POA713): GetObligationFilters for group DN=*string1*: Wrong class for dxrObligationLink.

Explanation: The Object referenced by the group's dxrObligationLink is not an obligation.

Action: Reassign an obligation to the group. Check if the object descriptions are

correctly loaded during startup.

ERR(POA714): Error in rule processor: Rule is neither accept nor cleanup rule DN=*string1*.

Explanation: RoleProvider combines ActionRunner with rule of wrong class. Should never occur.

Action: Call DirX Identity Support.

ERR(POA715): Reading of user DN=*string1* referenced by account DN=*string2* failed *string3* when applying onAssignment for group DN=*string4*.

Explanation: Account has lost rights for given group. While trying to process OnAssignment the search for the user failed.

Action: Check the message details to correct the error.

ERR(POA716): Deleting account DN=*string1* from group DN=*string2* failed: *string3*.

Explanation: Delete of an account to group assignment failed.

Action: Check the error condition supplied with the message.

ERR(POA717): Moving account DN=*string1* from group member *string2* to *string3* failed for DN=*string4*: *string5*.

Explanation: Modification of an account to group assignment failed.

Action: Check the error condition supplied with the message.

ERR(POA718): Moving member *string1* from group member *string2* to *string3* failed for DN=*string4*: *string5*.

Explanation: Modification of a group membership failed.

Action: Check the error condition supplied with the message.

ERR(POA720): Error initializing privilege DN=*string1* for reapproval: Privilege is not flagged requiring reapproval.

Explanation: A privilege has to be initialized for reapproval by a consistency rule. It is not processed since it is not flagged requiring reapproval.

Action: Correct the search filter of the rule: Only search for privileges with dxrNeedsReapproval=TRUE.

ERR(POA721): Error initializing privilege DN=*string1* for reapproval: Privilege is not flagged requiring approval.

Explanation: A privilege has to be initialized for reapproval by a consistency rule. It is not processed since it is not flagged requiring approval.

Action: Correct the settings of the privilege's 'requires approval' and 'requires reapproval' options.

ERR(POA722): Error initializing privilege DN=*string1* for reapproval: Privilege is of wrong

type *string2*.

Explanation: A privilege has to be initialized for reapproval by a consistency rule. It is not processed since it is not a role, permission or group.

Action: Correct the search filter of the rule: Only roles, permissions and groups are searched. Check that the object descriptions are read without errors at agent startup.

ERR(POA723): Error initializing privilege DN=*string1* for reapproval: Reading of affected users failed: *string2*.

Explanation: A privilege cannot be initialized for reapproval since an error occurred while reading the privilege's affected users.

Action: Check and correct the reason for the error (network/server connection, paging timeouts, ..). Rerun the consistency rule.

ERR(POA724): The object DN=*string1* returned by account search, filter='*string2*', peer TS DN='*string3*' is of wrong type: *string4*.

Explanation: The search for associated accounts in a peer TS returned objects of wrong type.

Action: Check the object description in the peer TS.

ERR(POA725): Search for referenced accounts in peer TS DN='*string1*' failed for filter='*string2*': *string3*.

Explanation: The search for associated accounts in a peer TS failed. The accept/cleanup rule cannot be evaluated for the imported member.

Action: Check the detailed message. Correct the error and run rule again.

ERR(POA726): Cannot read user for imported member *string1*: *string2*.

Explanation: In a virtual TS with states but without peer TS the member references a user. The user cannot be read, thus the accept/cleanup rule cannot be applied to the user.

Action: Check the detailed message. Correct the error and run rule again.

ERR(POA858): Error(s) occurred during save of assignments of user DN=*string1*: *string2*.

Explanation: Error(s) occurred during save of assignments.

Action: Correct errors and resolve user again.

ERR(POA859): Error(s) occurred during save of user DN=*string1*: *string2*.

Explanation: Error(s) occurred during save of user.

Action: Correct errors and resolve user again.

ERR(POA860): Error(s) occurred deleting object DN=*string1*: *string2*.

Explanation: Delete of the object failed.

Action: Check details of the error message.

ERR(POA861): Cannot migrate user for SoD: Save of user DN=*string1* failed: *string2*.

Explanation: Error(s) occurred during save of user. Thus, the user is not migrated for SoD.

Action: Correct errors and run SoD migration again or perform a role resolution on the user (includes SoD migration).

ERR(POA862): The subject in RunSODCheckForPolicy is not a SoD policy: DN=*string1*. Skipping subject.

Explanation: The rule's subject has to be of type SoD policy.

Action: Check search filter and correct object class to dxrSodPolicy.

ERR(POA863): Professional license required to use SoD checks.

Explanation: The SoD policy cannot be executed since the feature is not available with the business license.

Action: Upgrade to professional license.

ERR(POA864): SoD checks are not activated.

Explanation: The SoD policy cannot be executed since the feature is not activated.

Action: Set dxrSodActive flag at the identity domain.

ERR(POA865): Cannot start reapproval workflow for assignment DN=*string1*: Assignment is of wrong type.

Explanation: The subject being processed by the rule processor is not an assignment. Thus, no reapproval workflow is started for the subject.

Action: Check your search filter, it has to contain the objectClass dxrAssignment. Check if the object descriptions are read error-free on startup of the policy agent.

ERR(POA866): Reading of assignment DN=*string1* failed for workflow DN=*string2*: *string3*.

Explanation: The assignment related to an old approval workflow cannot be read. Migration of this workflow is aborted.

Action: Check error condition, remove error, and restart the migration rule.

ERR(POA867): Cannot create subject order for subject DN=*string1* of workflow DN=*string2*: *string3*.

Explanation: The info order for the subject cannot be created in approval workflow migration. Unlikely to occur.

Action: Check your installation. Check the detailed message. Call DirX Identity Support.

ERR(POA868): Cannot create resource order for resource DN=*string1* of workflow DN=*string2*: *string3*.

Explanation: The order for the resource cannot be created in approval workflow migration. Unlikely to occur.

Action: Check your installation. Check the detailed message. Call DirX Identity Support.

ERR(POA869): Cannot create string representation for subject order DN=*string1* of workflow DN=*string2*: *string3*.

Explanation: The info order for the subject cannot be converted into its xml representation. Unlikely to occur.

Action: Check your installation. Check the detailed message. Call DirX Identity Support.

ERR(POA870): Cannot create string representation for resource order DN=*string1* of workflow DN=*string2*: *string3*.

Explanation: The order for the resource cannot be converted into its xml representation. Unlikely to occur.

Action: Check your installation. Check the detailed message. Call DirX Identity Support.

ERR(POA871): Errors during save of migrated workflow DN=*string1*: *string2*.

Explanation: The migrated workflow instance cannot be saved.

Action: Check if all schema extensions are applied. Check the detailed message.

ERR(POA872): Errors during removal of old assignment DN=*string1* after migration of workflow DN=*string2*: *string3*.

Explanation: The assignment cannot be removed after successful migration of the approval workflow.

Action: Correct assignment manually in the data view.

ERR(POA873): Removal of privileges from user DN=*string1* failed: *string2*.

Explanation: An error occurred while removing a TBDEL user's privileges.

Action: Remove error condition and re-run consistency rule.

ERR(POA874): No privileges are removed from user DN=*string1*: User is in state *string2*, only users in state TBDEL are processed.

Explanation: The privileges are not removed since the user is not in a TBDEL state.

Action: Correct search filter of consistency rule.

ERR(POA875): Removal of privileges from user DN=*string1* failed: Error reading white list DN=*string2*, attribute=*string3*: *string4*.

Explanation: The privileges are not removed since a white list is provided as argument to the rule that cannot be read.

Action: Check for broken link in the white list reference. Run consistency rule again.

ERR(POA876): Removal of privileges from user DN=*string1* failed: Error reading black list

DN=*string2*, attribute=*string3*: *string4*.

Explanation: The privileges are not removed since a black list is provided as argument to the rule that cannot be read.

Action: Check for broken link in the black list reference. Run consistency rule again.

ERR(POA877): Error(s) occurred during save of object DN=*string1*: *string2*.

Explanation: Error(s) occurred during save of object.

Action: Correct errors. Run consistency rule again.

ERR(POA878): The attribute *string1* is not defined as a *string2* in object description for *string3*.

Explanation: The rule expects a fixed attribute value type. The type used for described attribute name and object type is missing or incorrectly defined.

Action: Check the relevant object descriptor and correct or add the definition for the described attribute. Run consistency rule again.

ERR(POA879): The argument *string1* could not be parsed as a valid value of a *string2*.

Explanation: The rule expects a correct syntax of a fixed argument. The passed value could not be converted properly.

Action: Check the relevant argument and correct its syntax. Run consistency rule again.

ERR(POA880): No privileges are removed from user DN=*string1*: user is in state *string2*, only users in state DISABLED are processed.

Explanation: The privileges are not removed since the user is not in a DISABLED state.

Action: Correct search filter of consistency rule.

WAR(POA881): User entry '*string1*' is currently locked and can’t be processed.

Explanation: Another application is currently working on the user object. Try later again.

Action: Repeat the operation at a later time.

ERR(POA882): Updating the group memberships for '*string1*' failed: *string2*.

Explanation: Update of group membership attributes failed.

Action: Check the error message.

INF(POA883): Removing the attribute '*string1*' from object '*string2*'.

Explanation: Due to some inconsistency (due to realtime workflow running in parallel) the group membership attributes need to updated.

Action: None.

ERR(POA884): Error(s) occurred during save of created email object DN=*string1*: *string2*.

Explanation: Error(s) occurred during save of email object.

Action: Correct errors. Run consistency rule again.

FTL(POA900): Error while initializing the rule context: *string1*.

Explanation: Initializing the rule context failed. Thus, the rule processor did not start.

Action: Check details of the error message.

FTL(POA901): Usage: PolicyAgent <Configuration File>.

Explanation: The program was called with the wrong number of arguments.

Action: Use the correct call.

FTL(POA902): Configuration file '*string1*' does not exist.

Explanation: The configuration file passed to the agent does not exist.

Action: Correct the configuration file pathname.

FTL(POA903): Error(s) while processing configuration file '*string1*': *string2*.

Explanation: Error(s) occurred while processing the configuration file.

Action: Check detailed message to correct the errors.

FTL(POA904): Cannot initialise rule storage: *string1*.

Explanation: Initialise rule storage failed.

Action: Check detailed message to correct the errors.

FTL(POA905): Cannot initialise user storage: *string1*.

Explanation: Initialise user storage failed.

Action: Check detailed message to correct the errors.

FTL(POA906): Cannot create rule provider: *string1*.

Explanation: Initialise rule provider failed.

Action: Check detailed message to correct the errors.

FTL(POA907): No trace file defined in the agent.ini file.

Explanation: trace.filename missing in agent.ini file.

Action: Add a line trace.filename=<pathname> in agent.ini where <pathname> is the absolute pathname to the trace file.

FTL(POA908): Cannot open trace file '*string1*' for writing: *string2*.

Explanation: The trace file defined in the agent's ini file cannot be opened for writing.

Action: Check pathname, create missing directories. Check if the file is in use by another

process.

16.7.12. Logging Messages (PSV)

This topic describes the logging messages returned by the provisioning services implementation.

ERR(PSV001): Undefined attribute name: '*string1*'.

Explanation: This message occurs as part of a response to a request containing an undefined attribute name.

Action: Correct the request and try again.

ERR(PSV002): Invalid attribute syntax. Attribute name: '*string1*', Details: *string2*.

Explanation: This message occurs as part of a response to a provisioning request containing attribute syntax errors.

Action: Correct the request with respect to the erroneous attribute *string1* and try again.

ERR(PSV003): Configuration only allows single value for attribute '*string1*'.

Explanation: This message occurs as part of a response to a request and indicates that multiple attribute values have been specified for an attribute where only a single value is allowed.

Action: Correct the request with respect to the erroneous attribute *string1* and try again.

ERR(PSV004): Operation '*string1*' failed for object '*string2*': *string3*

Explanation: This message indicates that the provisioning services failed to process a request due to serious problems. Here, *string1* indicates the request type, *string2* is the distinguished name of the related object, and *string3* indicates the reason.

Action: Check the message to see if the problem is obvious and can be solved by correcting the request, then solve the problem. Otherwise, inform DirX Identity technical support about the problem.

ERR(PSV005): Unsupported request: *string1*.

Explanation: This message indicates that the provisioning services failed to process a request because it is not supported. Here, *string1* indicates the request type.

Action: Do not initiate requests of the type indicated in the message.

ERR(PSV006): Cannot read object: '*string1*'.

Explanation: This message indicates that the provisioning services failed to process a request because fulfilling that request required reading a Directory object which cannot be accessed. Here, *string1* is the distinguished name of the object.

Action: Correct the request with respect to the erroneous distinguished name *string1* and try again.

ERR(PSV007): Object '*string1*' does not represent a user.

Explanation: This message indicates that the provisioning services failed to process a request regarding an object which is not a user.

Action: Correct the request with respect to the erroneous distinguished name *string1* and try again.

ERR(PSV008): Role parameter value list must not be empty.

Explanation: This message indicates that the provisioning services failed to process a modifyUserRoleAssignment request which results in an assignment of a parameterized role without role parameter values.

Action: Correct the request and try again.

ERR(PSV010): Modification of property '*string1*' of an assignment is not supported.

Explanation: This message occurs as part of a response to a modifyUserRoleAssignment request modifying attributes other than start or end date.

Action: Correct the request and try again.

ERR(PSV011): Missing scope specification in Search request.

Explanation: This message occurs as part of a response to a Search request (for example SearchUser) without scope specification.

Action: Correct the request and try again.

ERR(PSV012): Invalid scope specification '*string1*' in Search request.

Explanation: This message occurs as part of a response to a Search request with invalid scope specification.

Action: Correct the request and try again.

ERR(PSV013): Missing base specification in Search request.

Explanation: This message occurs as part of a response to a Search request where the distinguished name of the Search base is missing.

Action: Correct the request and try again.

ERR(PSV014): Invalid filter specification in Search request: '*string1*'

Explanation: This message occurs as part of a response to a Search request where filter is invalid.

Action: Correct the request and try again.

ERR(PSV015): Invalid iterator identifier '*string1*' in iterate or closeIterator request.

Explanation: This message occurs as part of a response to an iterate or closeIterator request with an invalid identifier. Either, the identifier '*string1*' has never been valid for subsequent Iterate requests or the identifier has expired and hence cleared internally.

Action: Correct the request and try again. The iterator identifier must match with the iterator identifier of a previous Search request.

ERR(PSV016): Iterator identifier '*string1*' has expired.

Explanation: This message indicates that the iterator identifier '*string1*' has expired and cannot be used for subsequent Iterate requests any more.

Action: None. As a fall-back, you can retry to collect your search result by executing a Search request followed by Iterate requests.

ERR(PSV017): Exception *string1* in class *string2* when processing request *string3*: *string4*.

Explanation: Spmlv2 request handler caught a runtime exception when unmarshalling the SPMLv2 request and processing it.

Action: Check dispatcher tables and sent request. Correct the request and try again.

ERR(PSV018): Operation *string1* not allowed for entry *string2*.

Explanation: The access policies defined in your DirX Identity configuration do not allow the requested operation for the requested entry.

Action: Create appropriate access policy which allows said operation.

ERR(PSV019): Capability with uri '*string1*' is not supported for entry '*string2*'.

Explanation: The SPMLv2 request contains a capability, which is not supported for said entry.

Action: Correct capability uri in the request so that it is suitable to the type of said entry and try again.

ERR(PSV020): *string1* when modifying capability with uri '*string2*' for '*string3*': *string4*.

Explanation: The handler could not process a capability within a modifyRequest. Here, *string1* indicates the Java exception type, *string3* is the object to be modified, and *string4* indicates the problem cause in human-readable form.

Action: Check the capability data in the request. Correct the request and try again.

ERR(PSV021): Cannot initialize Crypt Support: *string1*.

Explanation: On start-up the Provisioning Service tried to initialize a cryptographic support with private key and certificates.

Action: Check the message details! Check the passwords? Can the private key and the certificate for password encryption be read from LDAP?

ERR(PSV022): No attributes in *string1* with requestID = '*string2*'.

Explanation: The (Add)Request could not be processed. Attributes are needed.

Action: Make sure the client sends request with attributes.

WAR(PSV100): Unexpected element beneath MatchRule capability: *string1*.

Explanation: The <matchrule> element within the request contains an unsupported child element.

Action: Correct the <matchrule> element within the request and try again.

WAR(PSV101): Cannot find role parameter: uid = *string1*, role parameter DN = *string2*.

Explanation: The <matchrule> element does not contain both the uid and the distinguished name of the referenced role parameter and the given value is wrong.

Action: Send a request with either a correct distinguished name or dxrUID attribute of the role parameter.

WAR(PSV102): Cannot not delete or replace role parameter matchrule with uid = *string1* and role parameter DN = *string2*.

Explanation: The role does not contain a matchrule with the given UID.

Action: Check consistency of role parameter uid and distinguished name for each matchrule specified in the request. Correct the request and try again.

WAR(PSV103): Invalid operator string for matchrule in role '*string1*': *string2*.

Explanation: The role contains a matchrule with invalid operator string *string2*. It is ignored in responses.

Action: Correct the matchrule.

WAR(PSV104): Invalid type string for matchrule in role '*string1*': *string2*.

Explanation: The role contains a matchrule with invalid type *string2*. It is ignored in responses.

Action: Correct the matchrule.

WAR(PSV105): Attribute *string1* is not a configured member of entities associated with object description *string2*.

Explanation: The attribute is either requested in a search or to be added / modified, but is not listed in the listTargetsResponse. It is ignored in the response.

Action: Correct the listTargetsResponse so that the related attribute is supported or do not try to read or update the attribute.

ERR(PSV106): *string1* when setting capability with uri '*string2*' for object '*string3*': *string4*.

Explanation: The handler could not set a capability within an addRequest. Here, *string1* indicates the Java exception type, *string4* indicates the problem cause. The request is rejected.

Action: Check the capability data in the request. Correct the request and try again.

ERR(PSV107): *string1* when instantiating capability handler *string2*: *string3*.

Explanation: The request handler failed in instantiating the capability handler with the class name taken from the capability handler table. Here, *string1* indicates the Java

exception type, *string3* indicates the problem cause.

Action: This problem should not occur under normal circumstances. Inform DirX Identity technical support about the problem.

WAR(PSV108): No attributes configured for entity type *string1* associated with object description *string2*.

Explanation: The request handler could not return any attributes in the response since none are configured for the given object description.

Action: Inform DirX Identity support to check the object description name to attributes table!

WAR(PSV109): No Entries defined for target *string1*.

Explanation: The listTargetsResponse does not contain any entries for the given target.

Action: Change the configured listTargetsResponse: either remove the target or add entries.

ERR(PSV110): Expected Reference in Capability with URI = *string1*, received class of type = *string2*.

Explanation: The request contained a reference capability with a non-reference element beneath it. The handler rejects the request.

Action: Provide SPMLv2 compatible reference capabilities.

ERR(PSV111): Dispatcher received *string1* from request handler *string2*: *string3*.

Explanation: The request dispatcher of the provisioning services returns an error response due to an internal handler problem. This should not occur under normal circumstances.

Action: Check error logs and inform DirX Identity support about the problem.

ERR(PSV112): *string1* does not support target identifier *string2*. Request id = *string3*.

Explanation: The said target identifier is not supported with the sent request. Here, *string1* indicates the request type, *string3* is the request identifier of the erroneous request.

Action: Check the configured listTargetsResponse file, correct the request and try again.

WAR(PSV113): Ignore modification element *string1* for object *string2*.

Explanation: A modifyRequest contains a non-modification element beneath <modification>. This element is ignored.

Action: Check the sent request and remove or correct the ignored element.

ERR(PSV114): Exception *string1* evaluating DSML modification element *string2* for object *string3*: *string4*.

Explanation: The provisioning services processed encountered a problem during

processing of attribute modifications. Here, *string1* indicates the Java exception type, *string4* indicates the problem cause.

Action: Check the error message and correct the attribute modifications in the request.

ERR(PSV115): Exception *string1* processing listTargetsRequest: *string2*.

Explanation: The request handler failed in reading the listTargetsResponse from file. Here, *string1* indicates the Java exception type, *string2* indicates the problem cause.

Action: Check the error message and check whether the problem cause is syntactical incorrectness of the listTargetsResponse file or if this file is even missing, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

ERR(PSV116): Exception *string1* when validating configuration for target *string2* with handler: *string3*.

Explanation: The provisioning services read and validate the configured listTargetsResponse from file during start-up. The error message states that this validation fails. Either, said handler for a given target cannot be found or the validating handler throws an exception.

Action: Check the listTargetsResponse in the file or the handler configuration.

ERR(PSV117): Missing reference identifier in *string1*.

Explanation: The user-to-company reference could not be stored in the user attribute.

Action: Check the request!

ERR(PSV118): Missing attribute dxrType for new target system.

Explanation: In order to find a template target system we need at least the type.

Action: Assure that addRequest for new target system contains a value for attribute dxrType!

ERR(PSV119): Cannot find a template target system with dxrType = '*string1*' and optional cluster = '*string2*'.

Explanation: The search for a template target system with given type didn't return exactly one result.

Action: Check there is exactly one template target system in the customer extensions or in the system domain for either type and cluster or only type!

ERR(PSV120): Caught *string1* when returning template target system with dxrType = '*string2*', dxrCluster = '*string3*': *string4*.

Explanation: The exception occurred when searching for a template target system and trying to return the result.

Action: Check if the matching template target system in the system domain has proper attributes!

ERR(PSV121): Caught *string1* when unmarshalling capability '*string2*': *string3*.

Explanation: The XML content of the capability does not match its namespace.

Action: Correct the XML content of the capability!

ERR(PSV122): Cannot find template folder *string1* for new target system '*string2*'.

Explanation: While trying to create a new target system, sub-folders for accounts and /or groups cannot be found beneath template target system.

Action: Make sure that the template target system in DirXmetaRole system domain contains folders "Accounts and Groups" or "Accounts" and "Groups"!

ERR(PSV123): Cannot find target folder *string1* for configuration of new target system '*string2*'.

Explanation: While trying to create a new target system, the configuration sub-folder should be created beneath the given container.

Action: Make sure that the target folder for the target system configuration exists or correct the attribute 'dxmTSConfigurationLink' in the addRequest!

ERR(PSV124): Exception '*string1*' when copying sub-folders of template system '*string2*' for new target system '*string3*': *string4*.

Explanation: While trying to create a new target system, could not create the sub-folders for Accounts, Groups and Configuration.

Action: Make sure that the template target system has appropriate sub-folders!

ERR(PSV125): Won't delete container subtree '*string1*' without flag 'recursive' set to true.

Explanation: The container to be deleted is not empty and the deleteRequest hasn't set the recursive flag.

Action: Set the flag "recursive" to true, if you want to delete a container!

ERR(PSV126): Caught Exception *string1* when deleting target system or target system container with DN = '*string2*': *string3*.

Explanation: The web service tried to delete the tree in LDAP store.

Action: Check the message details! Do you have enough access rights? Do the objects still exist?

ERR(PSV127): Wrong element <*string1*> in capability with URI '*string2*'.

Explanation: The SPMLv2 add request for a target system contains a capability with wrong content.

Action: Make sure that client sends a correct capability.

WAR(PSV128): Exception *string1* while parsing anyObject children in deleteRequest with ID = '*string2*': *string3*.

Explanation: The SPMLv2 delete request for a target system contains not-well-formed children.

Action: Make sure the client sends a correct request.

WAR(PSV129): Exception *string1* while copying target system with ID = '*string2*' to tombstone '*string3*': *string4*.

Explanation: Before the target system is deleted, the whole sub-tree is requested to be copied to the tombstone DN within the same LDAP server.

Action: Check the message details. Maybe the tombstone does not exist or access is denied?

ERR(PSV130): Cannot find superior entry for '*string1*'.

Explanation: Before copying the target system to the tombstone the target system containers beneath the tombstone cannot be created or found.

Action: Check the DN of the searched entry. Has the service enough access rights?

ERR(PSV131): Exception *string1* when reading the superior entry for target system '*string2*': *string3*.

Explanation: Before copying the target system to the tombstone the target system parent containers cannot be found.

Action: Check the message details. Is the DN of the searched entry correct? Has the service enough access rights?

ERR(PSV132): Runtime Exception *string1* when creating the superior entry for target system '*string2*': *string3*.

Explanation: Before copying the target system to the tombstone the target system parent containers cannot be created.

Action: Check the message details. Has the service enough access rights?

ERR(PSV133): Account handler '*string1*' expected account with DN = '*string2*', but is of class '*string3*'.

Explanation: The account handler was passed an object, which is not of class SvcTSAccount.

Action: Make sure the request contains a correct DN referencing an account.

ERR(PSV134): Unsupported reference data child: *string1*.

Explanation: Reference data for an account group membership may only contain <accountGroupMembership> child elements.

Action: Make sure the request contains correct reference data.

INF(PSV135): Set TS-State of group with DN = '*string1*' to DELETED.

Explanation: Because the group state was not DELETED, the service could not physically delete it, but set the TS-State to DELETED.

Action: None.

INF(PSV136): Set TS-State of account with DN = '*string1*' to DELETED.

Explanation: Because the account state was not DELETED, the service could not physically delete it, but set the TS-State to DELETED.

Action: None.

ERR(PSV137): Caught Exception *string1* when deleting leaf or container entry with DN = '*string2*': *string3*.

Explanation: The web service tried to delete the entry or the sub-tree in LDAP store.

Action: Check the message details! Do you have enough access rights? Do the objects still exist?

ERR(PSV138): Caught Exception *string1* when deleting account or account container with DN = '*string2*': *string3*.

Explanation: The web service tried to delete the entry or the sub-tree in LDAP store.

Action: Check the message details! Do you have enough access rights? Do the objects still exist?

ERR(PSV139): Exception *string1* when processing setPasswordRequest for entry with DN = '*string2*': *string3*.

Explanation: The web service could not change the password of the user or account.

Action: Check the message details! Do you have enough access rights? Does the object exist? Can the private key and the certificate for password encryption be read from LDAP?

ERR(PSV140): Found no matching object description for new entry with DN = '*string1*' and object class '*string2*'.

Explanation: The web service tried to create an entry, but could not find a matching object description.

Action: Check if there is an object description for the target system! Check the attributes in the addRequest!

ERR(PSV141): Cannot set password for entry with DN = '*string1*' due to missing CryptSupport.

Explanation: The web service needs to encrypt the password, but cannot obtain the necessary support.

Action: Check your service configuration! Make sure you supplied the correct pin.

ERR(PSV142): Cannot find a password policy entry for Entry with DN = '*string1*'.

Explanation: For password validation the web service needs a password policy to check against.

Action: Make sure the entry has a valid password policy link or the domain has at least a default policy.

ERR(PSV143): *string1* while reading/transforming binary value of attribute *string2* in Entry '*string3*': *string4*.

Explanation: The Search requested a binary attribute to be returned in ASN.1/BER encoding by adding ';binary' to the type. The decoding of the attribute failed.

Action: Check the attribute value(s) in LDAP: probably they are wrong.

ERR(PSV144): Ignoring BER encoded value because of wrong start tag '*string1*'.

Explanation: The Service tried to decode a binary attribute with a ASN.1/BER encoded value. The value didn't have the correct start byte 0x04. The value is ignored.

Action: Check the attribute value(s) in LDAP or provided by your client.

ERR(PSV145): Handler '*string1*' expected entry with DN = '*string2*' of Java class '*string3*' but was class '*string4*'.

Explanation: The request handler read the Storage object with the given DN from LDAP, but it was not the expected Java class.

Action: Make sure the request contains a correct DN and that the appropriate object description maps to the correct Java class.

ERR(PSV146): *string1* when deleting Entry with DN = '*string2*': *string3*.

Explanation: The service tried to delete the LDAP entry with the given DN, but the Java Exception was thrown.

Action: Check the access policies, the DN of the entry and the details of the exception message.

ERR(PSV147): *string1* when finding account for password change using filter '*string2*': *string3*.

Explanation: The account for a password change is identified by account and target system identifiers. The exception occured when searching the matching accounts and comparing with the target system attributes.

Action: Check the details of the error message and the content of the potential matching accounts or target system entries.

ERR(PSV200): *string1* when loading configuration from domain '*string2*': *string3*.

Explanation: The Web Service cannot load configuration from DirX Identity domain.

Action: Check domain configuration: Object descriptions, etc.

ERR(PSV201): Loading configuration of storage '*string1*' FAILED, error='*string2*'

Explanation: The provisioning services failed in binding to the Identity Store in order to load their configuration.

Action: Check the error log files to identify the problem cause. If the problem cause is deployment different from the instructions in the user documentation or temporary unavailability of the Identity Store, then fix the problem. Otherwise, inform DirX Identity

technical support about the problem.

ERR(PSV202): Internal cleaner thread problem: '*string1*'

Explanation: The internal cleaner thread of the provisioning services reports a problem during execution of its tasks. This should not happen under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

ERR(PSV203): Internal problem: Unsuitable object of class '*string1*' encountered while processing a privilege assignment to a user.

Explanation: This should not happen under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

ERR(PSV204): The parameter pagesize must be a positive integer.

Explanation: This message indicates that an invalid page size has been specified in a search request .

Action: Correct the request and try again.

ERR(PSV205): Invalid reference identifier for an assignment to be created: must be specified as distinguished name.

Explanation: An invalid reference identifier has been specified in a request of type AddUserRequest or AddUserRoleAssignmentRequest.

Action: Correct the request so that the related reference identifier is specified using the <provcore:dn> tag and try again.

ERR(PSV206): Cannot modify privilege assignment of user with DN = '*string2*' addressed by reference identifier '*string1*'. Either it does not exist or it is ambiguous.

Explanation: A privilege assignment with the specified identifier cannot be modified. Either it does not exist, or it is a DN and there are multiple assignments of that privilege.

Action: Provide an existing unique identifier and try again. This may include the necessity to re-read the related object again.

ERR(PSV208): Cannot modify role parameters of assignment '*string1*'. Role Parameter must be specified by unique identifier.

Explanation: A role assignment modification fails because a role parameter has not been specified by its unique identifier. Here, *string1* is the reference identifier as specified in the request.

Action: Correct the request and try again.

ERR(PSV209): Cannot assign role '*string1*' to user '*string2*'. Role Parameter must be specified by unique identifier.

Explanation: A role assignment creation fails because a role parameter has not been specified by its unique identifier.

Action: Provide the unique identifier of the role parameter and try again.

ERR(PSV210): Role parameter '*string1*' does not exist in assignment '*string2*'.

Explanation: A role parameter modification has been defined for an assignment where this role parameter has not been used. Here, *string1* is the role parameter and *string2* the reference identifier of the erroneous assignment.

Action: Correct the request and try again.

ERR(PSV211): Cannot delete not-existing value '*string1*' of role parameter '*string2*' in assignment addressed by reference identifier '*string3*'.

Explanation: The said role parameter modification cannot be performed because the related values are not present in the identity store. Here, *string3* is the reference identifier of the erroneous assignment as specified in the request.

Action: Correct the request and try again. This may include the necessity to re-read the related object again.

ERR(PSV212): Missing values for modification, modification type '*string1*', role parameter '*string2*', assignment reference identifier '*string3*'.

Explanation: A role parameter modification cannot be performed because the related values are missing in the request. Here, *string3* is the reference identifier of the related assignment as specified in the request.

Action: Correct the request and try again.

ERR(PSV213): Missing values for modification, modification type '*string1*', Property '*string2*', assignment reference identifier '*string3*'.

Explanation: A modification cannot be performed because the related values are missing in the request. Here, *string3* is the reference identifier of the related assignment as specified in the request.

Action: Correct the request and try again.

ERR(PSV214): Unsupported modification type '*string1*'.

Explanation: The modification type is not supported. This should not occur under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

ERR(PSV215): Value '*string1*' of property '*string2*' does not exist in assignment addressed by reference identifier '*string3*'.

Explanation: A modification cannot be performed because the related values are not present in the identity store. Here, *string3* is the reference identifier of the erroneous assignment as specified in the request.

Action: Correct the request and try again. This may include the necessity to re-read the related object again.

ERR(PSV216): Attribute value '*string1*' of Attribute '*string2*' does not exist.

Explanation: A provisioning request, for example ModifyUser, fails because attribute value *string1* does not exist for attribute *string2*.

Action: Correct the request and try again.

ERR(PSV217): Missing values for attribute modification, modification type '*string1*', Attribute '*string2*'.

Explanation: An attribute modification cannot be performed because the related values are missing in the request.

Action: Correct the request and try again.

ERR(PSV218): Unsupported attribute modification type '*string1*'.

Explanation: The said attribute modification type is not supported. This should not occur under normal circumstances.

Action: Inform DirX Identity technical support about the problem.

ERR(PSV219): Invalid value '*string1*' for parameter '*string2*'.

Explanation: A given value for said parameter is invalid.

Action: Correct the request and try again.

ERR(PSV220): Loading of bind parameters for technical bind FAILED.

Explanation: The provisioning services could not find the bind credentials for the technical bind in the configuration file.

Action: Make sure you provided an element <technicalbind> with all necessary parameters in the configuration file.

INF(PSV221): Imported certificate of subject '*string1*' to CryptSupport.

Explanation: The provisioning services successfully added one or more certificates that could be used for decrypting passwords.

Action: None.

INF(PSV222): Added PSE(s) to CryptSupport.

Explanation: The provisioning services successfully added one or more PSEs that could be used for encrypting passwords.

Action: None.

ERR(PSV223): Dispatcher received *string1* from user hook *string2* before handling request: *string3*.

Explanation: The user hook configured for this request could not process the request.

Action: Check the detailed error message and fix the user hook.

ERR(PSV224): Dispatcher received *string1* from user hook *string2* after handling request: *string3*.

Explanation: The user hook configured for this request could not process the request.

Action: Check the detailed error message and fix the user hook.

ERR(PSV225): Dispatcher expected *string1*, but received *string2* from user hook *string3*.

Explanation: The user hook returned a wrong request type. The request is ignored.

Action: Fix the user hook.

ERR(PSV226): Dispatcher expected *string1*, but received *string2* from user hook *string3*.

Explanation: The user hook returned a wrong response type after the request was processed successfully. The dispatcher creates an appropriate error response.

Action: Fix the user hook.

ERR(PSV250): Cannot compute Pso Identifier(DN) of the object to be added.

Explanation: An SPML V2 request is incomplete regarding the information identifying the DN of the object to be handled. A correct request must either contain a PSO Identifier to represent a DN, or a container identifier (parent DN) and a default value for the naming attribute (mostly cn).

Action: Correct the request or the object description.

ERR(PSV251): Handler '*string1*' received request without Pso Identifier.

Explanation: The PSO Identifier is missing in an SPML V2 request.

Action: Correct the request and try again.

ERR(PSV252): Unsupported execution mode: '*string1*'.

Explanation: The execution mode *string1* is not supported in a request.

Action: Correct the request and try again.

ERR(PSV253): Saving object '*string1*' failed due to exception '*string2*': *string3*.

Explanation: LDAP update of the requested operation failed due to the given exception.

Action: Analyze the error message.

ERR(PSV254): Problems while trying to authenticate: *string1*.

Explanation: Before processing the request authentication of the user failed.

Action: Analyze the error message.

ERR(PSV255): Exception while unmarshalling and processing a SOAP request: *string1*.

Explanation: Either the SOAP request cannot be unmarshalled or some hard exception occurred during processing.

Action: Check previous error messages or try to capture or log the SOAP request and validate it.

ERR(PSV256): Exception while importing PSE for en-/decryption: *string1*.

Explanation: Password resolver tried to initiate the crypto support and read the certificate.

Action: Check pin and certificate store.

ERR(PSV257): Exception while decrypting password associated with key '*string1*': *string2*.

Explanation: Password resolver tried to read encrypted password from properties file and decrypt it.

Action: Check key in password.properties file and encrypted string.

ERR(PSV258): Exception while reading *string1*. element from paged result: *string2*.

Explanation: Basic Search Request handler collected search result from LDAP page result. Probably a LDAP error.

Action: Analyze the error message.

INF(PSV259): Ignoring entry from search result of request with ID = *string1* since subject is not allowed to read it.

Explanation: The effective user is not allowed to read all entries of the search result.

Action: Update the access policies for the user, if you need to see all elements matching the filter.

ERR(PSV260): Configured ReferenceHandler does not implement IReferenceHandler: *string1*.

Explanation: The configured reference handler class does not implement the required interface.

Action: Correct the class, if you have configured and customized it or inform DirX Identity Support.

ERR(PSV261): Unexpected type of object '*string1*'. Expected: '*string2*', Actual: '*string3*'.

Explanation: An unexpected object type has been encountered while reading the object with distinguished name *string1* from the provisioning store. Here, *string2* is the expected type in terms of a Java class, *string3* is the actual type in terms of a Java class.

Action: If the problem cause is obvious from the message then fix the problem in your client application. Otherwise, especially if *string3* represents an exception, check whether the problem cause is syntactical incorrect customization of object descriptions in your provisioning store, then fix the problem. Otherwise, inform DirX Identity technical support about the problem.

ERR(PSV262): Required object class '*string1*' missing in addRequest with requestID = '*string2*': *string3*.

Explanation: The SPML Service expected the object class for the object to be created.

Action: Provide all necessary object classes for the new object, especially the one cited in *string1*.

ERR(PSV263): Reference of type '*string2*' for user '*string1*' does not contain the privilege DN.

Explanation: A SPMLv2 reference needs a toPSOID.

Action: Send a SPMLv2 request with a correct reference capability.

ERR(PSV264): Exception '*string1*' while evaluating assignment attribute *string2* for user '*string3*': *string4*.

Explanation: An assignment start or end date has been provided, but couldn't be transformed into a valid time.

Action: Correct the date and try again.

ERR(PSV265): Exception '*string1*' while assigning privilege *string2* to user '*string3*': *string4*.

Explanation: The SPML service tried to assign the privilege with optional assignment attributes, but caught the exception from the Service layer.

Action: Check the message details.

ERR(PSV266): Could not delete value *string2* in assignment attribute *string1* of privilege assignment '*string4*' for user '*string3*'.

Explanation: No existing attribute value matches the value to be deleted. The request is rejected.

Action: Provide an existing value or nothing to delete all existing values.

WAR(PSV267): Ignoring assignment attribute *string1* while modifying assignment of privilege '*string2*' for user '*string3*'.

Explanation: The given attribute name is not supported for assignment modifications. The modification of this attribute is ignored.

Action: Check the attribute name.

ERR(PSV268): Exception '*string1*' while modifying assignment attribute *string2* of privilege assignment '*string3*' to user '*string4*': *string5*.

Explanation: The SPML service tried to perform a modification for the assignment attribute. The request is canceled.

Action: Check the message details.

ERR(PSV269): Expected class '*string1*' but received '*string2*' while modifying role parameters of privilege assignment '*string3*' to user '*string4*'.

Explanation: The received class is not able to handle role assignments with parameters.

Action: Internal error: Inform DirX Identity technical support about the problem.

ERR(PSV270): Exception '*string1*' while deleting privilege assignment '*string2*' for user '*string3*': *string4*.

Explanation: The SPML service tried to delete a user-privilege assignment and caught an exception from the service layer. The request is canceled.

Action: Check the message details.

ERR(PSV271): Could not delete assignment of privilege with DN '*string1*' and assignment UID '*string2*' for user '*string3*'.

Explanation: The Service layer could not remove the assignment. The request is canceled.

Action: Check for any log messages of the service layer.

WAR(PSV272): Expected class '*string1*' but received '*string2*' while reading assignments for user '*string3*'.

Explanation: The SPML service reads privilege assignments from the Service layer object. The received class does not represent a user-to-privilege assignment. The object is ignored.

Action: Check for log messages of the service layer.

WAR(PSV273): Unexpected element of class '*string1*' in Permission MatchRule capability for node with DN='*string2*': *string3*.

Explanation: The <permMatchrules> element within the request contains an unsupported child element. It is ignored.

Action: Correct the <permMatchrules> element within the request and try again.

ERR(PSV274): Permission match rule capability for permission with DN= '*string1*' does not contain any rule: *string2*.

Explanation: A permission match rule needs at least one rule. The request is rejected.

Action: Correct the <permMatchrules> element within the request and try again.

ERR(PSV275): PRequest to set permission match rule capability for permission with DN= '*string1*' contains more than one set of rules: *string2*.

Explanation: Only one <permMatchrules> element is allowed in the capability. All but the first are ignored.

Action: Send only one <permMatchrules> element within a capability element.

ERR(PSV276): *string1* when deleting children of '*string2*': *string3*.

Explanation: The delete request for the given identifier contained the recursive flag set to true. Therefore the service first tried to delete the children of the entry. This failed. The request is rejected.

Action: Check the subtree entries and the details of the message.

ERR(PSV277): *string1* when deleting entry '*string2*': *string3*.

Explanation: The service tries to delete the given entry, but fails. The request is rejected.

Action: Check the entry and the details of the message. Has the entry children, which should be deleted first? If so, use the recursive flag!

INF(PSV301): Loading configuration using bind profile host=*string1*:_string2_, user='*string3*', password=***, domain='*string4*', ssl=*string5*.

Explanation: The provisioning services start loading their configuration from the Identity Store. This message displays the bind profile used for this purpose.

Action: None.

INF(PSV302): Loading configuration OK.

Explanation: The provisioning services succeeded in loading their configuration from the Identity Store.

Action: None.

INF(PSV303): Loading provisioning services (Version *string1*).

Explanation: This message indicates that the provisioning services are loaded. The String *string1* indicates the version.

Action: None.

INF(PSV304): Loading provisioning services OK.

Explanation: The provisioning services have been successfully loaded.

Action: None.

INF(PSV305): Cleaner thread started.

Explanation: A thread which performs cleanup tasks has been successfully activated.

Action: None.

INF(PSV306): Loading Target Response file(s) '*string1*'.

Explanation: The provisioning services have successfully loaded their SPML V2 targets from said configuration file and are ready for supporting the SPML V2 operation listTargets.

Action: None.

INF(PSV307): Received SPMLv2 *string1* with id: *string2*.

Explanation: The provisioning services are going to process the received request.

Action: None.

INF(PSV308): Create configuration for new target system with DN = '*string1*' beneath folder with DN = '*string2*'.

Explanation: The configuration folder for the new target system does not yet exist and is created beneath the given folder.

Action: None.

INF(PSV309): Value of Initialization parameter '*string1*' is: '*string2*'.

Explanation: The This parameter lists the files containing the listTarget responses.

Action: None.

ERR(PSV310): No list targets responses available.

Explanation: The service tried to read list target responses from files configured in the target2Response map, but didn't get any.

Action: Check the file name configuration in the application context.

ERR(PSV311): No matching authenticator found; authentication failed!

Explanation: The SOAP request did not contain header or variable authentication information which can be evaluated by one of the configured authenticators.

Action: Make sure the client sends valid authentication info with the request and/or adapt the authenticator configuration.

ERR(PSV312): Authentication failed for user with DN='*string1*': *string2*.

Explanation: The configured authenticator could not authenticate the user to LDAP.

Action: Check user/account-DN and password or whatever the authentication token sent by the client contains.

ERR(PSV313): Exception while cloning SvcSession and checking connection: *string1*.

Explanation: For each new request an existing session to the identity domain is cloned and initialized. A re-connect occurs in case the LDAP connection was lost.

Action: Probably the LDAP server or the network is temporary not available.

ERR(PSV314): *string1* when initializing authenticator *string2*: *string3*.

Explanation: For the incoming HTTP request we first try to authenticate using the authenticator used for the previous request. We ignore the exception and try all configured ones.

Action: Check the exception details. Inform DirX Identity Support.

ERR(PSV315): *string1* when checking authenticated domain connection: *string2*.

Explanation: In order to be able to authenticate an incoming HTTP request the Service wants to make sure that the existing LDAP connection is authenticated. If not, it tries a re-bind, but gets this exception. Authentication fails.

Action: Check the exception details. Check that the network and the LDAP server are available.

ERR(PSV316): *string1* when modifying attribute *string2* for entry '*string3*' with operation *string4*: *string5*.

Explanation: The service tried to perform a modification for the entry and got an exception. The request fails.

Action: Check the exception details and the existing attributes of the entry to be changed.

ERR(PSV317): *string1* while initializing domain flags: *string2*.

Explanation: After having created a session to the identity domain evaluating the domain parameters and feature flags failed. The request is rejected.

Action: Check the message details and the attributes of the domain root entry.

ERR(PSV318): Cannot bind to domain *string1* on '*string2*:_string3_' as '*string4* with ssl = '*string5*'; ErrorCode = *string6*.

Explanation: The Web Service cannot bind to DirX Identity domain.

Action: Check address and bind credentials. Is LDAP server available?

ERR(PSV319): *string1* when creating a session to Identity domain *string2*: *string3*.

Explanation: On receipt of a SOAP request the Web Service failed to create a new Session. The request is rejected.

Action: Check availability of the LDAP server, address and bind credentials.

WAR(PSV320): *string1* when return session to '*string2*' to pool: *string3*.

Explanation: After having processed a SOAP request the Web Service disposes and disconnects the session to the DirX Identity domain. The problem is ignored.

Action: Check if the LDAP server is still available.

ERR(PSV321): *string1* when reading UID of role parameter with DN='*string2*' for assignment of role '*string3*' to user '*string4*': *string5*.

Explanation: The web service got a DN for the role parameter, but needs the UID. Reading the role parameter entry from LDAP failed. The request is rejected.

Action: Check the DN of the role parameter.

ERR(PSV322): *string1* when searching the template for a new target system with DN='*string2*': *string3*.

Explanation: When creating a new target system the web service searches for a template, whose DN is given in the request. The service ignores the template and tries to find a matching target system based on the other criteria.

Action: Check the DN of the template.

INF(PSV323): Cannot find template target system with dxrType = '*string1*', dxrCluster = '*string2*'. Number found entries was: *string3*.

Explanation: The search for a template target system with given type and cluster didn't return exactly one result.

Action: Check there is exactly one template target system in the system domain for these parameters!

ERR(PSV324): No Password Manager available when processing set password.

Explanation: The web service needs to encrypt the password and send it as password change event, but cannot obtain the necessary support.

Action: Check the password.properties of your service configuration! Make sure you supplied the correct pin.

ERR(PSV325): *string1* when reading LDAP entry '*string2*': *string3*.

Explanation: The web service tried to read the LDAP entry with all attributes.

Action: Check the connection to the LDAP store.

ERR(PSV326): Cannot obtain a session to Identity domain.

Explanation: The Web Service cannot bind to DirX Identity domain. The Service returns an error response to the client.

Action: Check address and bind credentials. Is LDAP server available?

ERR(PSV327): *string1* when processing SOAP request: *string2*.

Explanation: After unmarshalling an incoming SOAP request the Web Service dispatched processing to an appropriate handler, but it returned with the exception. The Web Service returns an exception to the client.

Action: Analyze the stack trace and check for previous error messages of the specific handler.

ERR(PSV328): *string1* when marshalling the SPML response: *string2*.

Explanation: The web service obtained a SPMLv2 response from the specific handler and tried to marshall it to a string and produce the SOAP response. The Web Service returns an exception to the client.

Action: Analyze the stack trace. Check the entry in LDAP for special attribute values: e.g. wrong binary definition.

WAR(PSV329): Cannot find Account with DN='*string1*' or identifying attributes = '*string2*'.

Explanation: The Web Service used the DN or the list of identifying attributes in the PSO ID of the SPMLv2 request to find an account.

Action: Correct the data in the request.

WAR(PSV330): *string1* when changing password for entry with DN='*string2*': *string3*.

Explanation: The Web Service failed to change the password for the user or account.

Action: Check the associated password policies and the password history.

WAR(PSV331): *string1* when checking challenge/response data for entry with DN='*string2*': *string3*.

Explanation: The Web Service failed to check the challenge/response data for the user.

Action: Check the request data.

WAR(PSV332): *string1* when reading the challenges for entry with DN='*string2*': *string3*.

Explanation: The Web Service failed to read the challenges for the user.

Action: Check the request data.

WAR(PSV333): *string1* when reading the password policy for entry with DN='*string2*': *string3*.

Explanation: The Web Service failed to read the password policy for the user or account.

Action: Check the request data.

ERR(PSV334): Exception *string1* when processing a getChallenges request for entry with DN = '*string2*': *string3*.

Explanation: The web service failed to read the challenges of the user or account.

Action: Check the message details! Do you have enough access rights? Does the object exist?

ERR(PSV335): Exception *string1* when processing a getPasswordPolicy request for entry with DN = '*string2*': *string3*.

Explanation: The web service failed to read the password policy of the user or account.

Action: Check the message details! Do you have enough access rights? Does the object exist?

ERR(PSV336): Exception *string1* when processing a checkChallengeResponse request for entry with DN = '*string2*': *string3*.

Explanation: The web service failed to check the challenge/response pairs of the user or account.

Action: Check the message details! Is the request properly authenticated? Do you have enough access rights? Does the object exist?

ERR(PSV337): Exception *string1* when processing a status request for correlation ID = '*string2*': *string3*.

Explanation: The web service failed to read the password change state.

Action: Check the message details! Is the correlation ID correct?

INF(PSV338): HTTP request header *string1*: *string2*.

Explanation: The name and value of an HTTP request header.

Action: None.

INF(PSV339): HTTP remote address: *string1*.

Explanation: The IP address of the client that sent the request.

Action: None.

ERR(PSV340): Exception *string1* when processing a sendOtp request for entry with DN = '*string2*': *string3*.

Explanation: The web service failed to generate a one time password for the account.

Action: Check the message details! Do you have enough access rights? Does the account exist?

ERR(PSV341): Invalid type *string1* of a node used for *string2*, only types derived from *string3* are supported.

Explanation: The privilege2privilege reference handler was incorrectly configured for a wrong object type.

Action: Check the configuration, use the privilege2privilege handler only for privileges.

ERR(PSV342): Invalid attribute name *string1* used for *string2*, only attributes dxrRoleLink, dxrPermissionLink and dxrRoleLink are supported.

Explanation: The privilege2privilege reference handler was incorrectly configured for a wrong attribute name.

Action: Check the configuration, use the privilege2privilege handler only for defined attributes.

ERR(PSV343): Exception *string1* when creating a new instance of the authenticator *string2*: *string3*.

Explanation: The configured authenticator could not be instantiated.

Action: Check the message details! Is the configuration correct?

ERR(PSV344): Invalid object or object type was used (*string1*) for authentication using value *string2*, only object types based on SvcUser, SvcPersona and SvcFunctionalUser are supported.

Explanation: The authentication failed because no valid user object was found in the provisioning configuration.

Action: Use correct user object of proper type for authentication.

ERR(PSV345): The authenticator *string1* could not be initialized, an exception has occurred.

Explanation: The authenticator could not be initialized, no authentication can succeed using the named authenticator.

Action: Check the message details! Check the configuration you provided for the named authenticator.

ERR(PSV346): The authenticator *string1* could not parse HTTP headers, incorrect number of header tokens detected.

Explanation: The authenticator could not parse the authorization message data due to an invalid format.

Action: Inform DirX Identity support.

ERR(PSV347): The authenticator *string1* could not find any certificate for the subject *string2*, known certificates are *string3*.

Explanation: The authenticator could not find the required certificate in the configured truststore.

Action: Check existence of the certificate for the given subject in the configured truststore.

ERR(PSV348): The authenticator *string1* could not authenticate the user because the expected UID *string2* does not fit to the message UID *string3*.

Explanation: The authenticator could not verify authorization header because of incorrect UID.

Action: Inform DirX Identity support.

ERR(PSV349): The authenticator *string1* cannot verify authentication data using key type *string2*.

Explanation: The authenticator could not verify authorization header because of incorrect key type.

Action: Inform DirX Identity support.

ERR(PSV350): The authenticator *string1* cannot get value for *string2*, null value detected.

Explanation: The authenticator could not initialize the cryptography because of invalid configuration.

Action: Is the configuration correct? Check the documentation for more information about the named authenticator class.

ERR(PSV351): The authenticator *string1* cannot read the trust store *string2*.

Explanation: The authenticator could not read the configured truststore.

Action: Is the configuration correct? Does the configured trust store exists? It must be located within .../provisioningServlet/WEB-INF directory.

INF(PSV352): Send SMS event properties: '*string1*' body: '*string2*'.

Explanation: A notification event with the given parameters is sent.

Action: None.

FTL(PSV601): Loading Targets Configuration file '*string1*' failed: *string2*.

Explanation: The provisioning services failed in loading their SPML V2 targets from said configuration file, '*string2*' indicates the problem cause. Loading this configuration file is essential for support of the SPML V2 operation listTargets.

Action: Check the error message in order to check whether the configuration file is not readable or inconsistent, then fix the problem. Restart the servlet container (Tomcat or IDS-J)you deployed the provisioning services into.

16.7.13. Logging Messages (PWDMGMT)

This topic describes the logging messages returned by the Password management module.

INF(PWDMGMT201): The mail for: *string1* is not sent, because mapMailAddress is set to dummy.

Explanation: If the configuration attribute "mapMailAddress" is set to dummy, no mail is sent.

Action: If you want the mail to be sent to the original recipient clear the mapMailAddress attribute.

INF(PWDMGMT202): The mail for: *string1* is sent to *string2* as configured in mapMailAddress.

Explanation: If the configuration attribute "mapMailAddress" is set, each mail is sent to this address.

Action: If you want the mail to be sent to the original recipient clear the mapMailAddress attribute.

INF(PWDMGMT203): The users attribute dxrPwdChangedTime is empty. Hence expirationDate and daysToExpire cannot be calculated.

Explanation: If "expirationDate" and/or "daysToExpire" are used in the body text, dxrPwdChangedTime is needed to calculate them.

Action: Ignore it or check why the users dxrPwdChangedTime attribute is not set.

INF(PWDMGMT204): The attribute dxrPwdChangedTime is not in the attribute list. Hence expirationDate and daysToExpire cannot be calculated.

Explanation: If "expirationDate" and/or "daysToExpire" are used in the body text, dxrPwdChangedTime is needed to calculate them.

Action: Add the dxrPwdChangedTime attribute to the requested attributes list.

INF(PWDMGMT210): The password policy: *string1* has infinite password age. Hence no users with this policy must be notified regarding password expiration.

Explanation: No users with this policy must be searched for regarding password expiration.

Action: None.

INF(PWDMGMT230): Password for account DN=*string1* was changed successfully.

Explanation: Account password has successfully been changed.

Action: None.

INF(PWDMGMT231): Password for account DN=*string1* already exists.

Explanation: Account password already exists.

Action: None.

INF(PWDMGMT232): Successfully submitted request to initiate sync to ts for account: *string1*.

Explanation: Submitting request to sync account to ts succeeded.

Action: None.

INF(PWDMGMT233): No expired accounts were found. Nothing more to do.

Explanation: No expired accounts were found. Nothing more to do.

Action: None.

INF(PWDMGMT234): Total number of expired accounts: *string1*. Processing them....

Explanation: Expired accounts were found. For each of them password will be updated and request for password synchronization workflow will be sent.

Action: None.

INF(PWDMGMT235): Password for account DN=*string1* was generated successfully.

Explanation: Account password has successfully been generated.

Action: None.

INF(PWDMGMT236): The password is not synchronized for account DN=*string1* because the TS configuration prevents password sync.

Explanation: You may enable password synchronization to the connected directory by unchecking the Prevent Password Sync check box in the TS configuration.

Action: None.

INF(PWDMGMT237): The password is not synchronized for account DN=*string1* because TS state is *string2*.

Explanation: No password synchronization is done to a connected directory's account that does not exist or is in a DELETE state.

Action: None.

INF(PWDMGMT238): Sync to ts for account DN=*string1* is not allowed because of either

target system prevent sync or account TS state.

Explanation: Either the target system is set to prevent sync or 'tsstate' of account is NONE or DELETED.

Action: Check those settings.

INF(PWDMGMT239): Start searching for expired accounts...

Explanation: Searching for expired accounts.

Action: None.

INF(PWDMGMT240): Searching for expired accounts linked to policy: *string1*.

Explanation: Searching for expired accounts which have the specified policy set.

Action: None.

INF(PWDMGMT241): Searching for expired accounts with default policy.

Explanation: Searching for expired accounts with the default policy set.

Action: None.

INF(PWDMGMT242): The policy list of domain *string1* is empty. No notifications for expired passwords need to be sent.

Explanation: Password expiring dates of users can only be determined with a password policy.

Action: If you want to send password expiring mails (default) check your domain policy tree.

INF(PWDMGMT243): No expired user passwords for policy '*string1*' were found. Nothing more to do.

Explanation: No expired accounts were found. Nothing more to do.

Action: None.

WAR(PWDMGMT401): The policy list of domain *string1* is empty.

Explanation: Password expiring dates of users can only be determined with a password policy.

Action: If you want to send password expiring mails (default) check your domain policy tree.

WAR(PWDMGMT402): The password expiration activity was interrupted.

Explanation: The password expiration activity was interrupted by the java server due to reached timeout or server shutdown.

Action: Check your timeout settings and run the workflow again.

WAR(PWDMGMT403): The attribute "dxrPwdExpiryNotified" is not in the attribute list or

was empty yet for this user.

Explanation: If this attribute is not in the list, it cannot be incremented but only set to 1.

Action: Add this attribute to the requested attributes list if not already done.

ERR(PWDMGMT501): Search for users with LDAP Connector failed. Custom error message: *string1*.

Explanation: Searching for users with the specified search request file failed.

Action: Check the error message.

ERR(PWDMGMT502): Cannot submit mail request: *string1* because of missing attributes.

Explanation: The mail connector needs at least "to" and "from" attributes.

Action: Check the configuration of the PwExpirationNotification workflow.

ERR(PWDMGMT503): Performing mail request: *string1* failed. Custom error message: *string2*.

Explanation: The mail connector could not perform the mail request.

Action: Check the error message.

ERR(PWDMGMT504): The attribute "dxrPwdExpiryNotified" is not in the attribute list.

Explanation: If this attribute is not in the list, it cannot be incremented but only set to 1.

Action: Add this attribute to the requested attributes list.

ERR(PWDMGMT505): Setting the pwdExpiryNotified flag of user '*string1*' failed. Custom error message: *string2*.

Explanation: The LDAP connector could not update the pwdExpiryNotified flag of the user after sending him mail.

Action: Check the error message.

ERR(PWDMGMT530): Calculating the target system dn from account dn: *string1* failed. Account has wrong syntax.

Explanation: No target system dn could be calculated for given account.

Action: Check the dn of the account.

ERR(PWDMGMT531): Searching for account related target system with base: *string1* returns 0 results.

Explanation: No target system could be found with the given dn.

Action: Check the dn of the target system.

ERR(PWDMGMT532): Searching for account related target system with base: *string1* returns more than one result.

Explanation: More than one target system was found with the given dn.

Action: Check the dn of the target system.

ERR(PWDMGMT533): Searching for account related target system with base: *string1* failed. Error: *string2*.

Explanation: Searching for the given target system failed.

Action: Check the error message of the search.

ERR(PWDMGMT534): Searching for account related target system with base: *string1* returns null.

Explanation: The search result for the given target system is empty.

Action: Check the dn of the target system.

ERR(PWDMGMT535): Searching for account related target system failed. Exception: *string1*.

Explanation: Searching the ts name in the LDAP directory failed.

Action: Check the given error reason.

ERR(PWDMGMT537): Submitting request to initiate sync to ts for account: *string1* failed. Error: *string2*.

Explanation: Submitting request to sync account to ts failed.

Action: Check the error message.

ERR(PWDMGMT538): No policy or default policy exists for account: *string1*.

Explanation: Policy for account is missing.

Action: Check the account.

ERR(PWDMGMT539): Domain is not defined.

Explanation: There is no domain name defined in the job configuration.

Action: Check the job configuration.

ERR(PWDMGMT540): Scope is not defined.

Explanation: There is no scope defined in the job configuration.

Action: Check the job configuration.

ERR(PWDMGMT541): Topic prefix is not defined.

Explanation: There is no topic prefix defined in the job configuration.

Action: Check the job configuration.

ERR(PWDMGMT542): Expire before n days is not defined.

Explanation: There is no expire before defined in the job configuration.

Action: Check the job configuration.

ERR(PWDMGMT543): The incoming request does not contain an account identifier.

Explanation: No identifier is contained in the request.

Action: Check the request.

ERR(PWDMGMT544): Searching the account: *string1* returns 0 results.

Explanation: No account could be found with the given dn.

Action: Check the dn of the account.

ERR(PWDMGMT545): Searching the account: *string1* returns more than one result.

Explanation: More than one account was found with the given dn.

Action: Check the dn of the account.

ERR(PWDMGMT546): Searching the account: *string1* failed. Error: *string2*.

Explanation: Searching for the given account failed.

Action: Check the error message of the search.

ERR(PWDMGMT547): Searching the account: *string1* returns null.

Explanation: The search result for the given account is empty.

Action: Check the dn of the account.

ERR(PWDMGMT548): Searching account failed. Exception: *string1*.

Explanation: Searching the account in the LDAP directory failed.

Action: Check the given error reason.

ERR(PWDMGMT549): Converting the configured filter from LDAP to spml failed. Exception: *string1*.

Explanation: Exception in class FilterLdap2Spml .

Action: Check the exception message.

ERR(PWDMGMT550): No search base is defined in the workflow definition.

Explanation: No search base is defined in the workflow definition.

Action: Add search base to definition.

ERR(PWDMGMT560): Connector *string1* does not exist.

Explanation: The given connector name has not been found in the workflow configuration.

Action: Check the workflow configuration.

ERR(PWDMGMT561): Connector *string1* does not implement SPMLv1 requests.

Explanation: The given connector needs to implement DxmRequestor interface with methods addRequest(), modifyRequest(), etc.

Action: Take a connector class that implements this SPMLv1 requests!

ERR(PWDMGMT562): Unscrambling of password failed for account DN=*string1*: *string2*.

Explanation: Unscrambling of a scrambled password failed. Unlikely to occur as long as passwords are processed with DirX Identity algorithms only.

Action: Change password manually.

ERR(PWDMGMT563): Encryption of password failed for account DN=*string1*: *string2*.

Explanation: Encryption of a password failed.

Action: Check certificate/private key pair. Change password manually.

FTL(PWDMGMT901): No LDAP Connector is specified in the configuration file.

Explanation: An LDAP Connector is needed to search and update the directory.

Action: Add a <connector> definition for the LDAP Connector to the configuration file.

FTL(PWDMGMT902): No Mail Connector is specified in the configuration file.

Explanation: A Mail Connector is needed to inform the users whose passwords will expire soon.

Action: Add a <connector> definition for the Mail Connector to the configuration file.

FTL(PWDMGMT903): No search request file is specified in the configuration file.

Explanation: Search request parameters must be specified in a spml search request file.

Action: Add a <property name="searchReqfile"> with value=<filename> to the PwdExpire controller section.

FTL(PWDMGMT904): No "from" property is specified in the configuration file.

Explanation: The "from" property is mandatory for sending mails.

Action: Add a <property name="from"> with value=<valid mail address> to the PwdExpire controller section.

FTL(PWDMGMT905): No search base is specified in search request file.

Explanation: A search base must be specified in a spml search request.

Action: Add a search base to the search request file.

FTL(PWDMGMT906): Exception in function getUsersAndSendMail. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(PWDMGMT907): Exception in function getSearchReqFromFile. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(PWDMGMT908): Exception in function buildFilter. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(PWDMGMT909): Exception in function getMailAddReqFromSRE. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(PWDMGMT910): Exception in function getExpiryNotifiedModReq. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(PWDMGMT911): Exception in function getExpirationDate. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(PWDMGMT912): Exception in function getDaysToExpire. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(PWDMGMT920): No domain name is specified in the controller properties.

Explanation: A domain name must be specified in the controller properties.

Action: Add a domain name to the controller configuration.

FTL(PWDMGMT930): Add request is not implemented.

Explanation: An add request is not supported by the connector.

Action: Check why you received an Add request.

FTL(PWDMGMT931): Searching for expired accounts failed. Exception: *string1*. Workflow aborted.

Explanation: Searching for expired accounts in the LDAP directory failed.

Action: Check the given error reason.

FTL(PWDMGMT932): Exception in function generatePassword. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(PWDMGMT933): Exception in function updatePwdOfAccount. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(PWDMGMT934): Exception in function getTargetSystemOfAccount. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(PWDMGMT935): Exception in function getAccountAttrs. Exception message: *string1*.

Explanation: An exception occurred in the described function.

Action: Check the exception message.

FTL(PWDMGMT950): Initialization of CryptSupport library failed.

Explanation: Loading the crypt support library (used for decrypting passwords) failed.

Action: Check the error reason.

FTL(PWDMGMT951): Initialization of PwdManagement failed.

Explanation: Setup of the password management environment failed.

Action: Check the error reason.

FTL(PWDMGMT952): Creation of SPML filter for searching accounts failed; error: *string1*.

Explanation: Creation of the SPML filter (using password policy info) for searching accounts failed.

Action: Check the error reason.

16.7.14. Logging Messages (PWRST)

This topic describes the logging messages returned by the Password reset notification module.

INF(PWRST200): There are no accounts to be reset under base=*string1*.

Explanation: The search for accounts to be reset did return 0 objects.

Action: Check if the search parameters are configured correctly.

INF(PWRST201): There are *string1* accounts found to be reset under base=*string2*.

Explanation: The search for accounts to be reset returned more than 0 objects.

Action: None.

INF(PWRST202): Password for account DN=*string1* was changed successfully.

Explanation: Account password has successfully been changed.

Action: None.

INF(PWRST203): Password for account DN=*string1* already exists.

Explanation: Account password already exists.

Action: None.

WAR(PWRST400): The password reset activity was interrupted.

Explanation: The password reset activity was interrupted by the Java server due to reached timeout or server shutdown.

Action: Check your timeout settings and run the workflow again.

ERR(PWRST500): Password reset for account DN=*string1* failed. Error during generation of new password: *string2*.

Explanation: Cannot reset password of account since password generation failed.

Action: Check the detailed error message and correct the error. Run workflow again.

ERR(PWRST501): Password reset for personal account DN=*string1* failed: The account has no user assigned.

Explanation: Cannot reset password of personal account since the account has an empty dxrUserLink.

Action: Correct your LDAP filter for searching personal accounts.

ERR(PWRST502): Password reset for personal account DN=*string1* failed: Cannot read assigned user DN=*string2*: *string3*.

Explanation: Cannot reset password of personal account since the assigned user cannot be read.

Action: Check for a broken dxrUserLink. Correct or remove link.

ERR(PWRST503): Password reset for personal account DN=*string1* failed: The assigned user DN=*string2* has no dxmPassword.

Explanation: Cannot reset password of personal account since the assigned user has no password stored in dxmPassword.

Action: Change the user's password in Web Center.

ERR(PWRST504): Resetting password of account DN=*string1* failed: *string2*.

Explanation: Password reset for the denoted account failed.

Action: Check message details. Change the password manually in Web Center.

ERR(PWRST505): Cannot determine the related TS data for account DN=*string1*: *string2*.

Explanation: The target system for an account cannot be determined. Unlikely to occur in a scheduled workflow.

Action: Check message details. Check target system structure.

ERR(PWRST506): No search base is defined in the workflow definition.

Explanation: No search base is defined in the workflow definition.

Action: Add search base to definition.

ERR(PWRST507): Converting the configured filter from LDAP to SPML failed. Exception: *string1*.

Explanation: Exception in class FilterLdap2Spml.

Action: Check the exception message.

ERR(PWRST508): Topic prefix is not defined.

Explanation: There is no topic prefix defined in the job configuration.

Action: Check the job configuration.

FTL(PWRST900): No search filter is configured for account search.

Explanation: No search filter is configured for account search.

Action: Configure a valid LDAP filter in the join activity's controller tab.

FTL(PWRST901): Syntax error in account filter *string1*: *string2*.

Explanation: The LDAP filter for account search is syntactically incorrect.

Action: Check the message details and correct the filter.

FTL(PWRST902): Searching for accounts to be reset returned errors: *string1*. Workflow aborted.

Explanation: The LDAP search for accounts to be reset returned an error.

Action: Check the message details. Remove the error condition and run workflow again.

FTL(PWRST903): Modify request is not implemented.

Explanation: A modify request is not supported by the connector.

Action: Check why you received an Modify request.

16.7.15. Logging Messages (RBA)

This topic describes the logging messages returned by the DirX Identity Service Agent.

INF(RBA100): *string1*

Explanation: Generic message.

Action: None.

INF(RBA201): Program "*string1*", Version "*string2*" started ***

Explanation: Start message.

Action: None.

INF(RBA202): Program "*string1*" ended with *string2* fatalError(s), *string3* error(s), and *string4* warning(s).

Explanation: End message.

Action: None.

INF(RBA203): *string1*

Explanation: Usage Message.

Action: Analyze input arguments.

INF(RBA204): Successfully connected to LDAP server "*string1*".

Explanation: Successfully connected to LDAP server.

Action: None.

INF(RBA205): All rights removed of user DN=*string1*.

Explanation: All rights of given user were removed.

Action: None.

INF(RBA206): User DN=*string1* deleted from the data store.

Explanation: User was deleted from the data store.

Action: None.

INF(RBA207): Processing of user DN=*string1* ended with *string2* error(s), *string3* warning(s) at *string4*.

Explanation: Summary message for given user.

Action: None.

INF(RBA208): Changes in group resolution for user DN=*string1* successfully stored.

Explanation: Success message of user group resolution.

Action: None.

INF(RBA209): No changes detected in group resolution for user DN=*string1*.

Explanation: No changes detected in group resolution for given user.

Action: None.

INF(RBA210): Processing of user DN=*string1* with state TBD started at *string2*.

Explanation: Start message for handling of given user with state DELETED.

Action: None.

INF(RBA211): User DN=*string1* not modified, rights are already removed. User is waiting for deletion.

Explanation: User not modified because rights are already removed. User is waiting for deletion.

Action: None.

INF(RBA212): processModifiedUsers: Counter = *string1*.

Explanation: Given number of users to analyze were successfully handled.

Action: None.

INF(RBA214): processDeletedUsers: Counter = *string1*.

Explanation: Given number of users to be deleted were successfully handled.

Action: None.

ERR(RBA240): Processing of user DN=*string1* ended with error(s) at *string2*.

Explanation: A user to be deleted could not be processed successfully.

Action: Check user.

ERR(RBA241): Error(s) occurred while writing status=*string1* for user DN=*string2*.

Explanation: Update of users status attribute failed. Obsolete?

Action: None.

ERR(RBA242): Save of user DN=*string1* failed: *string2*.

Explanation: Save of user failed with given error message.

Action: Analyze error message.

FTL(RBA250): No values are given for options: *string1*.

Explanation: Arguments are missing in command line of agent.

Action: Check command line parameters.

FTL(RBA251): Cannot open trace file "*string1*" for writing.

Explanation: Given trace file could not be opened.

Action: Check trace file.

FTL(RBA252): Cannot open report file "*string1*" for writing.

Explanation: Given report file could not be opened.

Action: Check report file.

ERR(RBA300): BrokenReference junior role "*string1*" not existing.

Explanation: A role has a broken link to its junior role.

Action: Check the dxrRolelink attribute.

ERR(RBA301): BrokenReference permission "*string1*" not existing.

Explanation: A role has a broken link to its permission.

Action: Check dxrPermissionLink attribute.

ERR(RBA302): BrokenReference targetsytem of account "*string1*" not found (not of class TS).

Explanation: Evaluating the target system of the given account returns an object which seems not to be a valid target system.

Action: Check the targetsytem structure where the account belongs to.

ERR(RBA303): BrokenReference targetsytem of account "*string1*" not found.

Explanation: Cannot evaluate the target system of given account.

Action: Check account.

ERR(RBA304): BrokenReference admin "*string1*" not found.

Explanation: The target system has invalid admin link.

Action: Check the dxrAdminLink attribute of the target system.

ERR(RBA305): BrokenReference user "*string1*" not found.

Explanation: The account has an invalid user link.

Action: Check the dxrUserLink attribute of the account.

INF(RBA306): Account "*string1*" in state "*string2*" with end date "*string3*" deleted/expired.

Explanation: End date of account reached. Depending on state account is deleted or set to state DELETED.

Action: None.

ERR(RBA307): Invalid permission parameter "*string1*" .

Explanation: Given permission parameter of Group is not valid.

Action: Check given permission parameter of group. See proposal lists for valid values.

WAR(RBA308): Multiple primary users: account "*string1*" is also a primary for user "*string2*".

Explanation: More than one account of a specific target system references the same user. More than one of these accounts are marked as primary accounts.

Action: Check dxrIsPrimary attribute of accounts.

INF(RBA309): Inconsistent: user assigned but flagged as not needed.

Explanation: The account has a user assigned but is flagged as managed in target system only.

Action: Check account.

INF(RBA310): Group "*string1*" in state "*string2*" with end date "*string3*" deleted/expired.

Explanation: End date of group reached. Depending on state group will be deleted or set to state DELETED.

Action: None.

INF(RBA311): At least one error detected; please check dxrError attribute.

Explanation: More information is available in the dxrError attribute.

Action: Check messages in dxrError attribute.

ERR(RBA312): BrokenReference group "*string1*" not existing.

Explanation: The user has broken group link.

Action: Check the dxrAssignedGroups attribute of user.

ERR(RBA313): BrokenReference role "*string1*" not existing.

Explanation: The user has a broken role link.

Action: Check dxrRoleLink attribute of user.

ERR(RBA314): BrokenReference "*string1*" : "*string2*" not existing.

Explanation: The user has a broken link.

Action: Check given link attribute of user.

INF(RBA315): Account "*string1*" not element of TS.

Explanation: The given account is referenced by this group, but could not be found in the target system.

Action: Check the reference in group. Check account.

ERR(RBA316): Junior role "*string1*" is not a role.

Explanation: A role has a junior role defined. The dxrRoleLink attribute does not point to a role.

Action: Check dxrRoleLink attribute.

ERR(RBA317): Assigned permission "*string1*" is not a permission.

Explanation: The dxrPermissionLink attribute does not point to a permission.

Action: Check dxrPermissionLink attribute.

INF(RBA318): No junior role and no permission assigned.

Explanation: The role is not valid.

Action: Check role.

ERR(RBA319): Invalid value "*string1*" for property "*string2*".

Explanation: An invalid value for dxrState or dxrTSState attribute was detected.

Action: Check state attribute.

ERR(RBA320): BrokenReference obligation "*string1*" not existing.

Explanation: Group has a broken obligation link.

Action: Check dxrObligationLink attribute of group.

ERR(RBA321): BrokenReference obligation "*string1*" is not of type obligation.

Explanation: The dxrObligationLink attribute does not point to an obligation.

Action: Check dxrObligationLink attribute.

INF(RBA322): Account "*string1*" not element of TS; moved to delete.

Explanation: An account in groupmemberstate OK or ADD is missing. It is moved to DELETE state.

Action: None.

INF(RBA323): Inconsistent: no user assigned and not TS local.

Explanation: The account has no user assigned and is not flagged as managed in target system only.

Action: Check account.

INF(RBA350): Cannot read dxrLDAPSizeLimit of Domain "*string1*".

Explanation: The dxrLDAPSizeLimit is missing in given domain.

Action: None.

WAR(RBA360): At least one error detected; please check dxrError attribute

Explanation: More information is available in the dxrError attribute.

Action: Check messages in dxrError attribute.

ERR(RBA361): User to group resolution failed for user DN=*string1*: *string2*.

Explanation: An unexpected exception occurred during user to group resolution.

Action: Check the detailed message. Correct the error condition and resolve user manually in Identity Manager.

WAR(RBA362): No LDAP lock could be set for user '*string1*'. User will be skipped.

Explanation: There are problems when setting the LDAP lock.

Action: Check the message.

ERR(RBA400): *string1*

Explanation: Generic message.

Action: None.

INF(RBA500): User to group resolution started for user DN=*string1* at *string2*.

Explanation: Start message for user to group resolution.

Action: None.

FTL(RBA999): *string1*

Explanation: Generic message.

Action: Read the message carefully. Try to solve the problem. If you cannot solve it, inform DirX Identity support about the problem.

16.7.16. Logging Messages (RPC)

This topic describes the logging messages returned by the DirX Identity rule processor. Includes rule processing algorithms and action processing.

INF(RPC101): Processing rule cluster for privilege '*string1*'.

Explanation: The rules are clustered according to the privileges to be processed. The rule processor informs about the privilege under consideration.

Action: None.

INF(RPC102): Processing rule cluster for accept/cleanup rules.

Explanation: The rule processor processes accept and cleanup rules.

Action: None.

INF(RPC103): Skipping rule *string1* in state TBDEL.

Explanation: In TBDEL state, only grant rules are processed for auto-revocation. All other rules are skipped.

Action: None.

INF(RPC110): Processing rule: *string1*.

Explanation: The rule being processed.

Action: None.

INF(RPC111): Subject *string1* matches exclude filter. Skipping ...

Explanation: A subject is not being processed since it matches the exclude filter.

Action: None.

INF(RPC120): SIMULATE: Assigning user DN=*string1* to account DN=*string2*.

Explanation: The rule processor simulates assigning a user to an account.

Action: None.

INF(RPC121): Assigning user DN=*string1* to account DN=*string2*.

Explanation: A user is assigned to an account.

Action: None.

INF(RPC130): SIMULATE: Adding property '*string1*' with value '*string2*' to subject DN=*string3*.

Explanation: The rule processor simulates adding a property to a subject.

Action: None.

INF(RPC131): Adding property '*string1*' with value '*string2*' to subject DN=*string3*.

Explanation: A new property value is created and the property is added to a subject.

Action: None.

INF(RPC140): SIMULATE: Adding user's attribute '*string1*' with value '*string2*' to list '*string3*', user DN=*string4*).

Explanation: The rule processor simulates adding a property to a distribution list.

Action: None.

INF(RPC141): Adding user's attribute '*string1*' with value '*string2*' to list '*string3*', user DN=*string4*.

Explanation: A property is added to a distribution list.

Action: None.

INF(RPC142): Value not added: User's attribute '*string1*' with value '*string2*' already exists in list '*string3*', user DN=*string4*).

Explanation: A property is not added to a distribution list, since it already exists in the list.

Action: None.

INF(RPC143): The user's attribute '*string1*' is empty, user DN=*string2* not added to list '*string3*'.

Explanation: The user is not added to a distribution list, since the attribute to be added to the list is empty.

Action: Enter a value for the user's attribute.

INF(RPC150): SIMULATE: Running command "*string1*" for subject with DN=*string2*.

Explanation: The rule processor simulates execution of a command.

Action: None.

INF(RPC151): Running command "*string1*" for subject with DN=*string2*.

Explanation: A command (script or executable) is executed on the subject.

Action: None.

INF(RPC155): SIMULATE: Running Java operation *string1*.*string2* on object with DN=*string3*.

Explanation: A Java operation is applied to the subject.

Action: None.

INF(RPC158): Running Java script method *string1* on subject with DN=*string2*.

Explanation: A Java script action is applied to the subject.

Action: None.

INF(RPC160): Type | priority | DN of the rules being applied.

Explanation: Table header of the rules being invoked.

Action: None.

INF(RPC161): Grant | *string1* | *string2*.

Explanation: Priority and name of a grant rule being invoked.

Action: None.

INF(RPC162): Deny | *string1* | *string2*.

Explanation: Priority and name of a deny rule being invoked.

Action: None.

INF(RPC163): Accept | *string1* | *string2*.

Explanation: Priority and name of an accept rule being invoked.

Action: None.

INF(RPC164): Cleanup | *string1* | *string2*.

Explanation: Priority and name of an cleanup rule being invoked.

Action: None.

INF(RPC165): Granting privilege by applying the following provisioning rules:

Explanation: List of rules being applied to grant the privilege.

Action: None.

INF(RPC166): Revoking privilege by applying the following provisioning rules:

Explanation: List of rules being applied to revoke the privilege.

Action: None.

INF(RPC167): Accepting/cleaning up privileges by applying the following provisioning rules:

Explanation: List of rules being applied for accept/cleanup of a privilege.

Action: None.

INF(RPC168): Processing validation rule: *string1*.

Explanation: The validation rule being processed.

Action: None.

INF(RPC169): Processing deny rule: *string1*.

Explanation: The deny rule being processed.

Action: None.

INF(RPC180): *string1*.

Explanation: The message shows a message that the son process (started by the action runner) sent to stdout.

Action: check the detailed message.

INF(RPC181): Asynchronous process successfully started. Muting output/error channels.

Explanation: An asynchronous process has been started, running independently from now on.

Action: None.

INF(RPC182): Command returns with exit code=0.

Explanation: The command returns 0, indicating a successful execution.

Action: None.

INF(RPC183): *string1*.

Explanation: A log message output from a Java script.

Action: None.

WAR(RPC190): *string1*.

Explanation: A warning output from a Java script.

Action: Check the message details.

ERR(RPC191): Error while saving object DN=*string1*: *string2*.

Explanation: An error occurred during save of the object.

Action: Check the detailed message. Correct error and run policy agent again

INF(RPC200): *string1* rule(s) processed in *string2* ms.

Explanation: Log how many rules have been processed.

Action: None.

INF(RPC201): Rule*string1* | Type*string2* | Subjects | Errors | Warnings |

Explanation: Header of rule statistics.

Action: None.

INF(RPC202): *string1* | *string2* | *string3* | *string4* | *string5* |

Explanation: Line of rule statistics.

Action: None.

WAR(RPC500): Privilege filters are not supported for grant rules. Filter not processed for rule DN=*string1*.

Explanation: A grant rule has a privilege filter which is ignored during further processing.

Action: Delete privilege filter from the rule.

WAR(RPC501): Cannot replace attribute *string1* in expression *string2*: Attribute does not exist at subject DN=*string3*.

Explanation: A consistency has no effect on a subject since the filter expression cannot be evaluated.

Action: Add the attribute to the subject or use a modified expression.

WAR(RPC502): Cannot build join filter for searching associated user for account dn=*string1*: Replacement of attributes failed for filter expression *string2*.

Explanation: A consistency rule using the assocAccount2User method has no effect on a subject since the join filter cannot be evaluated.

Action: Add the missing attributes to the subject or use a modified join filter.

WAR(RPC503): Attribute value for attribute *string1* not set at subject *string2*. Attributes from naming rule *string3* are missing on the subject.

Explanation: Some of the attributes required to evaluate the naming rule are missing on the subject.

Action: Add the missing attributes to the subject or use a modified naming rule.

WAR(RPC504): Fixed parameter '*string1*' supplied by rule is not an input parameter of action DN=*string2*. Parameter ignored.

Explanation: A consistency rule supplies a parameter not being defined at the related action.

Action: Remove the parameter from the rule or modify the action.

WAR(RPC505): Fixed parameter '*string1*' supplied by rule has context subject in action DN=*string2*. Parameter ignored.

Explanation: A consistency rule supplies a parameter whose context is subject.

Action: Change the parameter's context or remove it from the rule.

ERR(RPC700): Error while unmarshalling AssignmentDetails: *string1*.

Explanation: Errors occurred while processing the user's dxrAssignmentDetails attribute.

Action: Edit the content in the data view. Check correctness of its xml structure. Remove attribute.

ERR(RPC701): Validation error while unmarshalling AssignmentDetails: *string1*.

Explanation: Validation errors occurred while unmarshalling the xml description into Java objects. Check the detailed message.

Action: Edit the content in the data view. Check correctness of its xml structure. Remove attribute.

ERR(RPC702): Marshalling error while cloning AssignmentDetails: *string1*.

Explanation: Error while marshalling/unmarshalling while cloning the assignment details. Check the detailed message.

Action: Edit the content in the data view. Check correctness of its xml structure. Remove attribute.

ERR(RPC703): Validation error while cloning AssignmentDetails: *string1*.

Explanation: Validation error while marshalling/unmarshalling while cloning the assignment details. Check the detailed message.

Action: Edit the content in the data view. Check correctness of its xml structure. Remove attribute.

ERR(RPC704): Error while marshalling AssignmentDetails: *string1*.

Explanation: Errors occurred while marshalling Java objects into an xml description. Check the detailed message.

Action: Edit the content in the data view. Check correctness of its xml structure. Remove attribute.

ERR(RPC705): Validation error while marshalling AssignmentDetails: *string1*.

Explanation: Validation errors occurred while marshalling Java objects into an xml description. Check the detailed message.

Action: Edit the content in the data view. Check correctness of its xml structure. Remove attribute.

ERR(RPC710): Initialization of provisioning rule *string1* failed: *string2*. Skipping processing of rule.

Explanation: Initialization of a provisioning rule fails.

Action: Check the error message and correct error(s).

ERR(RPC711): Object with DN=*string1* is not a rule, class is *string2*. Check search filter and object mapping.

Explanation: The search of the rules to be applied returns an object of wrong class (not a rule).

Action: Check the rule filter in the agent's startup file. Check if the object descriptions are loaded error free during startup.

ERR(RPC712): Cannot read privilege DN=*string1*, *string2*: Rules for privilege not processed.

Explanation: A privilege cannot be read from the datastore. The related rules are not processed for that privilege.

Action: Check and correct the broken privilege link. Check server connection. Assign another privilege or remove role.

ERR(RPC720): Initialization failed for rule '*string1*': *string2*.

Explanation: The rule is not processed since it could not be initialized.

Action: Check the error message and correct error(s).

ERR(RPC721): Skipping processing of rule due to preparation error(s): *string1*.

Explanation: The rule is not processed since it could not be initialized.

Action: Check the detailed error message and correct the error(s).

ERR(RPC722): Error while running action on rule *string1*: *string2*.

Explanation: An error occurred while running a rule's action.

Action: Check the detailed error message and correct the error(s).

ERR(RPC723): Error while processing rule *string1*: *string2*.

Explanation: An error occurred while processing a rule.

Action: Check the detailed error message and correct the error(s).

ERR(RPC730): Cannot create account for user DN=*string1* in the data store: Value range of naming rules exceeded.

Explanation: The value range for account names is exceeded.

Action: Create account manually.

ERR(RPC731): Cannot create account for user DN=*string1* in the data store: *string2*.

Explanation: An error occurred during account creation.

Action: Remove error condition/Create account manually.

ERR(RPC740): Cannot associate accountDN=*string1* to user: Search result is not unique for filter='*string2*' under base='*string3*'.

Explanation: The automatic association of an unassigned account to a user failed: More than one user found.

Action: Assign user to account manually by setting the account's dxrUserLink.

INF(RPC741): Cannot associate accountDN=*string1* to user: Search result is empty for filter='*string2*' under base='*string3*'.

Explanation: The automatic association of an unassigned account to a user failed: No user found.

Action: Assign user to account manually or mark the account being a local account.

ERR(RPC742): Cannot associate accountDN=*string1* to user: Search result is not unique for filter='*string2*' under base='*string3*': *string4* users found.

Explanation: The automatic association of an unassigned account to a user failed: More than one user found.

Action: Assign user to account manually by setting the account's dxrUserLink.

ERR(RPC857): Initialization of masters failed: *string1*.

Explanation: Processing of master properties is not possible due to initialization errors. Unlikely to occur.

Action: Check error text details and correct the error.

ERR(RPC880): Error in naming rule for attribute *string1* of class *string2*: *string3*.

Explanation: A naming rule is incorrect.

Action: Correct the attribute's naming rule in the object descriptions.

ERR(RPC881): Error while executing javamethod *string1* for attribute *string2* of class *string3*: *string4*.

Explanation: An error occurs during run of a Java method defined in a naming rule.

Action: Check the detailed message and correct the error.

ERR(RPC890): *string1*.

Explanation: The message shows a message that the son process (started by the action runner) sent to stderr.

Action: Consult the manual of the external tool that was started by the action runner.

ERR(RPC891): The command '*string1*' executed on subjectDN=*string2* has been interrupted: *string3*.

Explanation: Processing of an external tool has been interrupted.

Action: Check the detailed message.

ERR(RPC892): Error(s) occurred while processing command '*string1*' on subjectDN=*string2*:

Explanation: The following lines show the messages being written from the executable to stderr.

Action: Consult the manual of the external tool that was started by the action runner.

ERR(RPC893): Command '*string1*' returns error status, exit code=*string2*.

Explanation: The command returns a non-zero exit status, indicating errors/warnings have occurred.

Action: Consult the manual of the external tool that was started by the action runner.

ERR(RPC894): Running Java script *string1.string2* on subjectDN=*string3* returns error: *string4*.

Explanation: An error occurred during Java script-Execution.

Action: Check the detailed message. Correct syntax error(s) in the Java script.

ERR(RPC895): *string1*.

Explanation: An error message output from a Java script.

Action: Check the detailed message.

16.7.17. Logging Messages (RWF)

This topic describes the logging messages returned by the RequestWorkflow Configuration.

ERR(RWF338): Cannot create request workflow factory. Instantiation failed: *string1*.

Explanation: Creating the request workflow factory failed.

Action: Check your installation, the request workflow jars have to be present in the lib directories.

ERR(RWF340): Creating request workflow service failed for endpoint=*string1*, user=*string2*: *string3*.

Explanation: The request workflow service could not be created on the Java based identity server. Thus, no request workflow is started.

Action: Check server and network connection to Java based identity server.

ERR(RWF350): Reading IDS-J SOAP configuration from connectivity LDAP server failed.

Explanation: The request workflow endpoint could not be created from the IDS-J server configuration. Thus, no request workflow is started.

Action: Check server and network connection to Java based identity server. Check preceding messages and correct configuration errors.

ERR(RWF500): Cannot determine host:port of DirX Identity connectivity DB: Cannot read domain root of provisioning domain *string1*: *string2*.

Explanation: The domain root object cannot be read. Thus, host and port of the LDAP server holding the related connectivity configuration cannot be determined. As a consequence, no request workflow is started.

Action: Check server and network connection.

ERR(RWF501): The host where the DirX Identity connectivity DB is installed is not configured at the domain root of provisioning domain *string1*.

Explanation: The dxrMetahubServer attribute is empty at the domain root. As a consequence, no request workflow is started.

Action: Enter the host where the DirX Identity connectivity DB is installed in the domain root's dxrMetahubServer attribute.

ERR(RWF502): The port of the LDAP server where the DirX Identity connectivity DB is installed is not configured at the domain root of provisioning domain *string1*.

Explanation: The dxrMetahubPort attribute is empty at the domain root. As a consequence, no request workflow is started.

Action: Enter the port of the LDAP server where the DirX Identity connectivity DB is installed in the domain root's dxrMetahubPort attribute.

ERR(RWF503): Cannot connect to the LDAP server *string1*:_string2_ as user *string3*, where the DirX Identity connectivity DB is installed: *string4*.

Explanation: Connection to the LDAP server with the DirX Identity connectivity DB fails. Thus, the message server configuration cannot be determined and no request workflow is started.

Action: Check the detailed message. Check for server connection and bind credentials.

ERR(RWF504): No IDS-J server configured in connectivity database.

Explanation: The IDS-J server configuration cannot be read. Thus, host and port of the

IDS-J server cannot be determined. As a consequence, no request workflow is started.

Action: Run the Configuration Wizard to configure the IDS-J server. Check LDAP server and network connection.

WAR(RWF505): More than one IDS-J server configured in connectivity database. Reading request workflow configuration data from DN=*string1*.

Explanation: More than one IDS-J server is present in the configuration database. The request workflow service configuration is read from the first server returned by the search.

Action: Check configuration of all IDS-J server objects. The domain property has to be unique. Correct via Identity Manager in Expert View.

ERR(RWF506): Cannot read IDS-J server from connectivity database: *string1*.

Explanation: The LDAP search for IDS-J server configuration returned error(s). Thus, no request workflow is started.

Action: Check detailed message and remove error(s). Check server and network connections.

ERR(RWF507): No service configured for IDS-J server DN=*string1*.

Explanation: The link to the service configuration cannot be read from the IDS-J server configuration. As a consequence, no request workflow is started.

Action: Configure IDS-J server and its related service/system properly.

ERR(RWF508): No service found in search with filter=*string2* under base=*string3* in DirX Identity confdb.

Explanation: The service referenced by the message service configuration's dxmService-DN is not found. Thus, host and port of the message service cannot be determined.

Action: Correct the dxmService-DN link of the message service configuration.

ERR(RWF509): Server ports dxmDataPort/dxmSecurePort are missing or empty in the service related to the IDS-J server.

Explanation: The dxmAddress/dxmSecurePort attributes are empty. As a consequence, no request workflow is started.

Action: Edit the service related to the message server. In case you use SSL, enter the secure port to dxmSecurePort and set ssl true in the specific attributes. Otherwise, enter the port to dxmDataPort.

ERR(RWF510): No system found in search with filter=*string2* under base=*string3* in DirX Identity confdb.

Explanation: The system referenced by the service configuration's dxmSystem-DN is not found. Thus, host of the IDS-J service cannot be determined. No request workflow is started.

Action: Correct the dxmSystem-DN link of the service configuration.

ERR(RWF511): Searching for attributes *string1* with filter=*string2* under base=*string3* on DirX Identity connectivity server failed: *string4*.

Explanation: Reading of attributes from the DirX Identity connectivity DB fails. Thus, the IDS-J server configuration is incomplete and no request workflow is started.

Action: Check the detailed message. Check for server and network connection.

ERR(RWF512): IDS-J server host can neither be read from the system nor from the service.

Explanation: The dxmAddress attribute is empty, or the related configuration object cannot be read. As a consequence, no request workflow is started.

Action: Edit the system related to the IDS-J server. Enter the hostname in dxmAddress.

ERR(RWF513): The system is not referenced by the IDS-J server configuration.

Explanation: The system link is not set at the IDS-J service. Therefore, the IDS-J server host cannot be read an no request workflows are started.

Action: Add a system to the IDS-J service. Enter the hostname in dxmAddress.

WAR(RWF514): A temporary error occurred while trying to read IDS-J server SOAP configuration: *string1*. Retrying.

Explanation: Reading the IDS-J server configuration ran on a temporary error. After waiting 20s, this is retried, until the retry limit is exceeded.

Action: Check LDAP server and network connections.

ERR(RWF515): Error reading activities or subactivities failed; processing of activities is incomplete; error: *string1*.

Explanation: Retrival of activities or subactivities failed for the given reason.

Action: Check the error messages.

ERR(RWF81): *string1*

Explanation: Check the detailed message.

Action: None.

WAR(RWF82): *string1*

Explanation: Check the detailed message.

Action: None.

ERR(RWF971): *string1* while unmarshalling request to string: *string2*.

Explanation: The request workflow service client request could not be converted to a readable XML string.

Action: Check the error message.

ERR(RWF972): *string1* while executing buildCreateInstanceRequest: request: *string2*, error: *string3*.

Explanation: The request workflow service client request could not successfully create the 'CreateInstance' request.

Action: Check the error message.

ERR(RWF973): *string1* while converting subject order to string: error: *string2*.

Explanation: The request workflow service client request could not successfully create the 'CreateInstance' request. Conversion of the subject order (for analyzing the problem) failed.

Action: Check the error message.

ERR(RWF974): *string1* while converting subject order to string: error: *string2*.

Explanation: The request workflow service client request could not successfully create the 'CreateInstance' request. Conversion of the resource order (for analyzing the problem) failed.

Action: Check the error message.

INF(RWF975): Subject order (simple string): *string1*.

Explanation: The request workflow service client request could not successfully create the 'CreateInstance' request. The subject order is shown for analyzing the problem.

Action: None.

INF(RWF976): Resource order (simple string): *string1*.

Explanation: The request workflow service client request could not successfully create the 'CreateInstance' request. The resource order is shown for analyzing the problem.

Action: None.

ERR(RWF977): *string1* caught: *string2*

Explanation: The request workflow service client request could not successfully create the request workflow instance due to an exception.

Action: Remote exception

ERR(RWF978): *string1* finding workflows matching request workflow definition filter: operation: *string2*, subject type: *string3*, resource type: *string4*; error: *string5*.

Explanation: The workflow service tried to find workflow definitions matching the given filter. The request is rejected.

Action: Check the message details and the filter.

ERR(RWF979): The workflow '*string1*' (assigned at privilege '*string2*') is not active and therefore will not be started.

Explanation: The workflow service tried to find workflow definitions matching the given filter. The request is rejected.

Action: Check the workflow links at the given privilege.

16.7.18. Logging Messages (SEC)

This topic describes the logging messages returned by the DirX Identity security processing.

WAR(SEC501): Cannot convert ACL filter *string1* to LDAP filter: *string2*.

Explanation: The conversion of an Acl Filter to an LDAP Filter is not possible. All subjects are searched and the filtered (performance-issue).

Action: Check detailed message. Avoid expressions like *$(subject.attr)*. Do not combine $(subject.dn) with other filter expressions in one rule.

WAR(SEC502): Reading LDAP attributes from Access Policy resource filters and rules failed: *string1*.

Explanation: Reading the LDAP attributes from the access policies failed. Thus, performance of access policy evaluation may be reduced.

Action: Check detailed message.

ERR(SEC700): Invalid resource: *string1 string2*.

Explanation: Error when searching the referenced object.

Action: Check detailed message, network- and server connections.

ERR(SEC701): Exception while reading access policies: *string1*.

Explanation: Error when searching an access policy.

Action: Check detailed message, network- and server connections.

ERR(SEC702): Exception while reading base nodes: *string1*.

Explanation: An exception occurred. See the message text.

Action: Check detailed message, network- and server connections.

ERR(SEC703): Exception while reading resource: '*string1*'; '*string2*'.

Explanation: An exception occurred. See the message text.

Action: Check detailed message, network- and server connections.

ERR(SEC704): Exception while reading subject: '*string1*'; '*string2*'.

Explanation: An exception occurred. See the message text.

Action: Check detailed message, network- and server connections.

ERR(SEC705): Exception while checking rule: '*string1*'; '*string2*'.

Explanation: An exception occurred. See the message text.

Action: Check detailed message, network- and server connections.

ERR(SEC706): Exception while reading users: *string1*.

Explanation: An exception occurred. See the message text.

Action: Check detailed message, network- and server connections.

ERR(SEC707): Exception while reading delegation assignments: *string1*.

Explanation: An exception occurred. See the message text.

Action: Check detailed message, network- and server connections.

ERR(SEC708): Exception while reading groups: *string1*.

Explanation: An exception occurred. See the message text.

Action: Check detailed message, network- and server connections.

ERR(SEC709): Exception in AccessPolicy constructor: *string1 string2*.

Explanation: An exception occurred. See the message text.

Action: Check detailed message, network- and server connections.

ERR(SEC710): Exception while creating delegation assignment: *string1*.

Explanation: An exception occurred. See the message text.

Action: Check detailed message, network- and server connections.

ERR(SEC711): Exception while reading access rights: *string1*.

Explanation: An exception occurred. See the message text.

Action: Check detailed message, network- and server connections.

ERR(SEC712): Exception while reading users: *string1*.

Explanation: An exception occurred. See the message text.

Action: Check detailed message, network- and server connections.

ERR(SEC713): Cannot determine approvers for order id=*string1*, resource=*string2*: Processing of order failed: *string3*.

Explanation: Processing of an order failed. No object could be created to derive the approvers from.

Action: Check detailed message.

ERR(SEC714): Cannot determine approvers for order. Order is of wrong type *string1*. It has to be an XML string or a DOM node.

Explanation: Processing of an order failed. No object could be created to derive the

approvers from.

Action: Check detailed message.

ERR(SEC715): Cannot determine approvers for order. Reading of order failed: *string1*.

Explanation: Reading of an order failed. No object could be created to derive the approvers from.

Action: Check detailed message.

ERR(SEC716): Cannot determine approvers for order id=*string1*: Reference to resource is missing in order *string2*.

Explanation: The DN of the referenced privilege is missing in the order for an assignment. In case the order is created by DirX Identity client, this is unlikely to occur.

Action: Check and correct the source of the order. Ensure it writes the correct order format.

ERR(SEC717): Exception while reading users: *string1*, filter= *string2*.

Explanation: An exception occurred. See the message text.

Action: Check detailed message, network- and server connections.

ERR(SEC718): Cannot read effective user: *string1*.

Explanation: An exception occurred while reading the effective user. Unlikely to occur.

Action: Check detailed message, network- and server connections.

ERR(SEC719): Cannot read resource DN=*string1*: *string2*.

Explanation: An exception occurred while reading a resource. Thus, the resource cannot be supplied by the resource provider.

Action: Check detailed message, network- and server connections. Correct broken link(s).

ERR(SEC720): Syntax error in resulting filter *string1* for resource DN=*string2*: *string3*.

Explanation: The combined filter for resource search and access control rules is incorrect. Thus, the resource is not provided for selection.

Action: Check and correct filter syntax.

ERR(SEC721): Cannot read proposal link root DN=*string1* of role parameter DN=*string2* from the datastore: *string3*.

Explanation: The object referenced by the dxrProposalLinkRoot attribute of the role parameter cannot be read. Thus, no hierarchical role parameters are available for selection.

Action: Check the message details. Correct the error and restart the application.

ERR(SEC722): Cannot read children of node DN=*string1* during search of role parameter

root nodes for parameter DN=*string2*: *string3*.

Explanation: The children of a container or not assignable parameter object cannot be read. Presumably the list of hierarchical role parameter root nodes is incomplete.

Action: Check the message details. Correct the error and restart the application.

ERR(SEC723): Cannot evaluate access policies for node DN=*string1* during search of role parameter root nodes for parameter DN=*string2*: *string3*.

Explanation: Error occurred during evaluation of the access policy for a role parameter value. Presumably the list of hierarchical role parameter root nodes is incomplete.

Action: Check the message details. Correct the error and restart the application.

WAR(SEC724): Check for valid resources in subtree DN=*string1* failed during search of role parameter root nodes for parameter DN=*string2*: *string3*.

Explanation: This warning occurs e.g. in case a contains rule is defined for a DN attribute. If the message is displayed, the subtree is processed even though no valid object may exist.

Action: Check the message details. Check you access policy, if beginsWith/contains/endsWith rules are used for DN type attributes. This may result in a lower performance.

ERR(SEC725): Malformed rule filter *string1* in Access Policy *string2*: Cannot extract subject attribute.

Explanation: The access policy's rule filter is malformed.

Action: Correct the rule filter and restart the application.

ERR(SEC726): Broken reference chain while evaluating access policy *string1*: *string2*=*string3* is broken for DN=*string4*.

Explanation: A reference chain defined in the access policy is broken..

Action: Correct the broken link reported in the message..

WAR(SEC727): Delegation assignments result in a loop; erroneous delegator name is '*string1*'.

Explanation: There is a loop in the delegation definitions. One approver delegates to someone and vice versa or using a third or fourth approver name.

Action: Check the delegation assignments and correct them so that no loop exists.

16.7.19. Logging Messages (SRT)

This topic describes the logging messages returned by the DirX Identity realtime workflow services.

ERR(SRT500): Cannot determine host:port of DirX Identity Connectivity Configuration: Cannot read domain root of provisioning domain *string1*: *string2*.

Explanation: The domain root object cannot be read. Thus, host and port of the LDAP server holding the related connectivity configuration cannot be determined. As a consequence, no realtime provisioning workflow is started.

Action: Check server and network connection.

ERR(SRT501): The host where the DirX Identity Connectivity Configuration is installed is not configured at the domain root of provisioning domain *string1*.

Explanation: The dxrMetahubServer attribute is empty at the domain root. As a consequence, no realtime provisioning workflow is started.

Action: Enter the host where the DirX Identity Connectivity Configuration is installed in the domain root's dxrMetahubServer attribute.

ERR(SRT502): The port of the LDAP server where the DirX Identity Connectivity Configuration is installed is not configured at the domain root of provisioning domain *string1*.

Explanation: The dxrMetahubPort attribute is empty at the domain root. As a consequence, no realtime provisioning workflow is started.

Action: Enter the port of the LDAP server where the DirX Identity Connectivity Configuration is installed in the domain root's dxrMetahubPort attribute.

ERR(SRT503): Cannot connect to the LDAP server *string1*:_string2_ as user *string3*, where the DirX Identity Connectivity Configuration is installed: *string4*.

Explanation: Connection to the LDAP server with the DirX Identity Connectivity Configuration fails. Thus, the message server configuration cannot be determined and no realtime provisioning workflow is started.

Action: Check the detailed message. Check for server connection and bind credentials.

ERR(SRT504): Cannot read specific attributes of DirX Connectivity server root node: *string1*.

Explanation: Reading of specific attributes from the DirX Identity Connectivity Configuration root fails. Thus, the messaging product cannot be determined and no realtime provisioning workflow is started.

Action: Check the detailed message. Check for server and network connection.

ERR(SRT505): Searching for attributes *string1* with filter=*string2* under base=*string3* on DirX Identity connectivity server failed: *string4*.

Explanation: Reading of attributes from the DirX Identity Connectivity Configuration fails. Thus, the message server configuration is incomplete and no realtime provisioning workflow is started.

Action: Check the detailed message. Check for server and network connection.

ERR(SRT506): The dxmService-DN attribute is missing at the message service configuration.

Explanation: The link to the service is missing. Thus, address and port cannot be

retrieved. As a consequence, no realtime workflow is started.

Action: Repair the corrupt message service by adding a service link in the expert view.

ERR(SRT507): No event stream is configured at the MQ series message server configuration.

Explanation: The dxmStreamEvent attribute is empty. As a consequence, no realtime workflow is started.

Action: Edit the message server's dxmStreamEvent in the data view.

ERR(SRT508): No queue manager is configured at the MQ series message server configuration.

Explanation: The dxmQueueManager attribute is empty. As a consequence, no realtime workflow is started.

Action: Edit the message server's dxmQueueManager in the data view.

ERR(SRT509): Server address dxmAddress is missing in the service related to the message server.

Explanation: The dxmAddress attribute is empty. As a consequence, no realtime workflow is started.

Action: Edit the service related to the message server. Enter the message server host in dxmAddress.

ERR(SRT510): Server ports dxmDataPort/dxmSecurePort are missing or empty in the service related to the message server.

Explanation: The dxmAddress/dxmSecurePort attributes are empty. As a consequence, no realtime workflow is started.

Action: Edit the service related to the message server. In case you use SSL, enter the secure port to dxmSecurePort. Otherwise, enter the port to dxmDataPort.

WAR(SRT511): Cannot send request id=*string1* to message service *string2*:_string3_. *string4*.

Explanation: Sending a JMS message to initiate a realtime workflow failed. As a consequence, no realtime workflow is started.

Action: Check the detailed message. Check message server and network connection.

WAR(SRT512): Cannot create request publisher: *string1*.

Explanation: Creating the request publisher failed. As a consequence, no realtime workflow is started.

Action: Check the detailed message. Check message server and network connection.

ERR(SRT513): Cannot extract the topic prefix from the topic with alias *string1*.

Explanation: The topic prefix for realtime provisioning is not configured. Thus, no request is sent.

Action: Create a topic entry in the connectivity configuration. Its topicName has to be the alias reported above, as value dxm.request.provisionToTS is recommended.

ERR(SRT514): Searching for realtime provisioning topic with filter=*string2* under base=*string3* in DirX Identity confdb failed. No matching entry exists.

Explanation: The topic for realtime provisioning is not configured. Thus, no request is sent.

Action: Create a topic entry in the connectivity configuration, with topiName=TOPIC_PROVISION_TO_TS. As value dxm.request.provisionToTS is recommended.

ERR(SRT515): Searching for realtime provisioning topic with filter=*string2* under base=*string3* in DirX Identity confdb failed. Search result is not unique.

Explanation: More than one topic is configured for topicName=TOPIC_PROVISION_TO_TS.

Action: Remove all but one configurations for that alias.

ERR(SRT516): No message service configured for realtime provisioning found in search with filter=*string2* under base=*string3* in DirX Identity confdb.

Explanation: No message service exists supporting the realtime provisioning topic.

Action: Add TOPIC_PROVISION_TO_TS to the dxmSupportedTopics attribute of the message service where realtime provisioning is configuration.

ERR(SRT517): More than one message service configured for realtime provisioning found in search with filter=*string2* under base=*string3* in DirX Identity confdb.

Explanation: More than one message service is configured for dxmSupportedTopics=TOPIC_PROVISION_TO_TS.

Action: Remove TOPIC_PROVISION_TO_TS from dxmSupportedTopics of all but one message server configurations.

ERR(SRT518): No service found in search with filter=*string2* under base=*string3* in DirX Identity confdb.

Explanation: The service referenced by the message service configuration's dxmService-DN is not found. Thus, host and port of the message service cannot be determined.

Action: Correct the dxmService-DN link of the message service configuration.

WAR(SRT519): A temporary error occurred while trying to read IDS-J server SOAP configuration: *string1*. Retrying.

Explanation: Reading the IDS-J server configuration ran on a temporary error. After waiting 20s, this is retried, until the retry limit is exceeded.

Action: Check LDAP server and network connections.

ERR(SRT520): Cannot retrieve message server configuration for topic alias *string1*.

Explanation: The configuration of the message service cannot be retrieved from the connectivity confDB.

Action: Check the message service configuration. Check that the topic alias is registered with the message service.

ERR(SRT521): Cannot create JMS message publisher for topic alias *string1*.

Explanation: Creating the JMS publisher failed. Maybe connection to the message service failed or the service is down.

Action: Check the message service configuration. Check message service and network connections.

INF(SRT522): Published *string1* for entry with DN="*string2*" and topic prefix=*string3* and type=*string4*.

Explanation: The JMS publisher successfully sent a JMS message containing a change event.

Action: None.

ERR(SRT523): Error while publishing *string1* for entry with DN="*string2*" and topic prefix=*string3* and type=*string4*.

Explanation: The JMS publisher could not sent a JMS message containing a change event.

Action: Check the log file for additional errors when sending messages. Check the ActiveMQ logs, too.

ERR(SRT524): No publisher could be created for publishing *string1* for entry with DN="*string2*" and topic prefix=*string3* and type=*string4*.

Explanation: No JMS publisher could be created for sending a JMS message containing a change event.

Action: Check the log file for additional errors when sending messages. Check the ActiveMQ logs, too.

16.7.20. Logging Messages (SVC)

This topic describes the logging messages returned by the DirX Identity Provisioning Services.

INF(SVC103): *string1*.

Explanation: Generic info message.

Action: None.

INF(SVC110): Running in SIMULATE mode: New *string1* with DN=*string2* is created but not saved. List of attributes:

Explanation: Informs the user that creation of a new user/persona/functional user is

simulated.

Action: None.

INF(SVC111): *string1=string2*.

Explanation: Lists an attribute name/value pair.

Action: None.

INF(SVC112): Running in SIMULATE mode: New *string1* with DN=*string2* is created but workflow DN=*string3* is not started. List of attributes:

Explanation: Informs the user that creation of a new user/persona/functional user by workflow is simulated.

Action: None.

INF(SVC113): Running in SIMULATE mode: Account *string1* is modified:

Explanation: Informs the user that modification of an account is simulated.

Action: None.

INF(SVC114): '*string1*': '*string2*' -> '*string3*'.

Explanation: Lists an attribute modification.

Action: None.

ERR(SVC120): Role DN=*string1* is no longer valid after deleting *string2* DN=*string3*.

Explanation: Deleting a role or a permission invalidates another role, since it has no more permissions/junior roles assigned.

Action: Check, if deleting the role or permission is required. Edit/remove the role that becomes invalid.

ERR(SVC121): Cannot remove PermissionAssignment *string1* from user DN=*string2*.

Explanation: A permission is being deleted. Removal of the assignment to that permission failed for the user.

Action: Edit user and remove permission assignment manually.

ERR(SVC122): Cannot remove RoleAssignment *string1* from user DN=*string2*.

Explanation: A role is being deleted. Removal of the assignment to that role failed for the user.

Action: Edit user and remove role assignment manually.

ERR(SVC123): Cannot remove GroupAssignment *string1* from user DN=*string2*.

Explanation: A group is being deleted. Removal of the assignment to that group failed for the user.

Action: Edit user and remove group assignment manually.

ERR(SVC125): Permission DN=*string1* is no longer valid after delete group DN=*string2*.

Explanation: Deleting the group results in a permission being no longer valid since it has no groups assigned.

Action: Check, if deleting the group is required. Edit/remove the permission that becomes invalid.

INF(SVC220): SIMULATE: Applying onRevocation actions on account DN=*string1*.

Explanation: The application of onRevocation actions is simulated only.

Action: None.

INF(SVC221): OnRevocation actions successfully applied on account DN=*string1*.

Explanation: The onRevocation actions have been applied.

Action: None.

ERR(SVC300): Referenced property *string1* does not exist for *string2*, DN=*string3*.

Explanation: A change in reference is not possible since the object has an empty referenced property.

Action: Fill referenced properties with unique values and re-run migration.

ERR(SVC301): Referenced property *string1* is not unique for *string2* DN=*string3*: It also exists for DN=*string4*.

Explanation: A change in reference is not possible since the object has no unique referenced property.

Action: Fill referenced properties with unique values and re-run migration.

ERR(SVC302): 100 or more errors detected. Check aborted.

Explanation: The check has run on more than 100 errors.

Action: Correct the errors and re-run the migration.

ERR(SVC303): LDAP error(s) detected during check of referenced property: *string1*. Check aborted.

Explanation: An LDAP error has occurred.

Action: Check details, correct the errors and re-run migration.

ERR(SVC304): The *string1* DN=*string2* is mapped to the wrong Java class. Check object descriptions in TS.

Explanation: Presumably due to an error in the object descriptions an account is incorrectly mapped.

Action: Correct object descriptions and re-run migration.

ERR(SVC305): Error(s) during save of *string1* DN=*string2*: *string3*. Object not migrated.

Explanation: An error occurred during save of an account/group object. Thus, the references are not updated.

Action: Analyze the error condition given in the detailed message. Restore server, remove error condition and re-run migration.

ERR(SVC306): Cannot read assigned groups of account DN=*string1*: *string2*. Account not migrated.

Explanation: An error occurred during read of the groups assigned to the account.

Action: Analyze the error condition given in the detailed message. Restore server, remove error condition and re-run migration.

ERR(SVC307): LDAP error(s) detected during migration: *string1*.

Explanation: An LDAP error has occurred.

Action: Analyze the error condition given in the detailed message. Restore server, remove error condition and re-run migration.

ERR(SVC308): Exception detected during migration: *string1*.

Explanation: An exception has occurred during migration.

Action: Analyze the error condition given in the detailed message. Restore server, remove error condition and re-run migration.

ERR(SVC320): Search for referenced objects failed for DN=*string1*: Filter='*string2*', searchbase='*string3*': *string4*.

Explanation: Error when searching the referenced objects.

Action: Check detailed message, correct the error condition, and re-run function.

ERR(SVC321): Read of referenced object DN=*string1* failed for *string2* DN=*string3*: *string4*.

Explanation: Error when trying to read the referenced object.

Action: Check for broken reference. Correct broken reference manually and re-run function.

ERR(SVC322): Search for referenced object failed for *string1* DN=*string2*: Filter='*string2*', searchbase='*string3*': Broken reference, search result is empty.

Explanation: Error when trying to read the referenced object.

Action: Check for broken reference. Correct broken reference manually and re-run function.

ERR(SVC323): Search for referenced object failed for DN=*string1*: Filter='*string2*', searchbase='*string3*': Search result is not unique, DN=*string4*, DN=*string5*...

Explanation: The search result for a referenced account or group object is not unique.

Action: Check and correct the assign attributes of the objects returned by the search (must be unique).

ERR(SVC324): Cannot access cn of assigned group DN=*string1*. Check if it is deleted from the datastore.

Explanation: This error may occur in case the group is in the application's cache but was meanwhile deleted from the datastore.

Action: Refresh the user.

ERR(SVC325): Cannot access cn of newly created account for group DN=*string1*. Check naming rules and user attributes.

Explanation: The newly created account does not have a cn.

Action: Check naming rules and create an account manually for the user.

ERR(SVC326): Cannot access cn of target system of group DN=*string1*.

Explanation: The common name of a target system cannot be read.

Action: Check and correct object descriptions for the target system. Check for exceptions in the Manager log during startup.

ERR(SVC327): CopyPrivileges: Assignment to DN=*string1* not copied from user DN=*string2* due to error: *string3*.

Explanation: The source user for copyPrivileges has an assignment with an error (presumably a broken link).

Action: Fix error at the source-user and re-run copyPrivileges.

ERR(SVC328): Professional license required to use Persona feature.

Explanation: The Persona feature cannot be enabled since the feature is not available with the business license.

Action: Upgrade to professional license.

ERR(SVC329): Professional license required to use Functional User feature.

Explanation: The Functional User feature cannot be enabled since the feature is not available with the business license.

Action: Upgrade to professional license.

ERR(SVC330): Creating an object from the template DN=*string1* by object description *string2* failed: *string3*.

Explanation: An error occurred while generating a new user, persona or functional user from a template object.

Action: Check the object description that is used for the conversion.

ERR(SVC331): Cannot read associated *string1* for user DN=*string2*: *string3*.

Explanation: An error occurred while reading the associated persona/functional user for the user.

Action: Check detailed message, network- and server connections.

ERR(SVC332): Cannot read associated user DN=*string1* for persona DN=*string2*: *string3*.

Explanation: An error occurred while reading the associated user for the persona.

Action: Check detailed message, check for broken owner link.

ERR(SVC333): Sending of modify event failed for account DN=*string1*. No password expiration workflow started.

Explanation: A user was removed from the account's dxrUsedBy attribute, but sending of a SPML request to the message server failed. Thus, no password expiration workflow is started for the modified account.

Action: Check message server and message server configuration. Change password manually.

ERR(SVC334): Sending of add request failed for new group DN=*string1*. No realtime workflow started.

Explanation: Sending of a SPML request to the message server failed. Thus, no realtime workflow is started for the new group.

Action: Check the preceding messages.

ERR(SVC335): Sending of add request failed for new account DN=*string1*. No realtime workflow started.

Explanation: Sending of a SPML request to the message server failed. Thus, no realtime workflow is started for the new account.

Action: Check the preceding messages.

ERR(SVC336): Sending of modify request failed for modified group DN=*string1*. No realtime workflow started.

Explanation: Sending of a SPML request to the message server failed. Thus, no realtime workflow is started for the modified group.

Action: Check the preceding messages.

ERR(SVC337): Sending of modify request failed for new account DN=*string1*. No realtime workflow started.

Explanation: Sending of a SPML request to the message server failed. Thus, no realtime workflow is started for the modified account.

Action: Check the preceding messages.

ERR(SVC338): Cannot create request workflow factory. Instantiation failed: *string1*.

Explanation: Creating the request workflow factory failed.

Action: Check your installation, the request workflow jars have to be present in the lib directories.

ERR(SVC339): Authenticated bind required to create request workflows.

Explanation: The user or application trying to create a request workflow is not authenticated.

Action: Bind to the LDAP server with user and password.

ERR(SVC340): Creating request workflow service failed for endpoint=*string1*, user=*string2*: *string3*.

Explanation: The request workflow service could not be created on the Java based identity server. Thus, no request workflow is started.

Action: Check server and network connection to Java based identity server.

ERR(SVC341): The privilege DN=*string1* to be assigned to user DN=*string2* cannot be read: *string3*. Assignment of privilege aborted.

Explanation: Assignment of the privilege to a user is aborted since the privilege cannot be read from the datastore.

Action: Check detailed message. Check server and network connection.

ERR(SVC342): The object DN=*string1* to be assigned to user DN=*string2* is not a privilege: *string3*. Assignment of privilege aborted.

Explanation: The object to be assigned to the user is not a privilege. Assignment aborted.

Action: Check whether the object descriptions are read without errors at tool startup. Check if the link to the assigned object points to an existing role, permission or group.

ERR(SVC343): Cannot create assignment from subject DN=*string1* to resource DN=*string2*: Error reading subject: *string3*.

Explanation: Assignment of a resource to a subject is aborted since the subject cannot be read from the datastore.

Action: Check detailed message. Check server and network connection.

ERR(SVC344): Cannot create assignment from subject DN=*string1* to resource DN=*string2*: Error reading resource: *string3*.

Explanation: Assignment of a resource to a subject is aborted since the resource cannot be read from the datastore.

Action: Check detailed message. Check server and network connection.

ERR(SVC345): Cannot create assignment from subject DN=*string1* to resource DN=*string2*: Assignments between subjects/resources of the given types are not supported.

Explanation: The assignment is not created since assignments between the types of

subjects/resources involved are not supported. With DirX Identity standard clients, this should not occur.

Action: Check detailed message. Check and correct the source creating the assignment. Check if the object descriptions are read-in without errors during startup.

ERR(SVC346): Failed to initialize assignment from user DN=*string1* to privilege DN=*string2* for re-approval: Error during save: *string3*.

Explanation: The assignment is not initialized for re-approval since an error occurred during save.

Action: Check detailed message. Check network- and server connections. Correct the error and re-run the consistency rule.

ERR(SVC347): Less than two conflicting privileges assigned to the policy. Assign additional privileges or set policy inactive.

Explanation: The SoD policy has less than two conflicting privileges assigned and is flagged active.

Action: Assign additional privileges or set policy inactive.

ERR(SVC348): Professional license required to use SoD checks.

Explanation: The SoD policy cannot be edited since the feature is not available with the business license.

Action: Upgrade to professional license.

ERR(SVC349): The assigned object DN=*string1* is not a privilege (role/permission/group).

Explanation: Only roles, permissions and groups are allowed here.

Action: Edit the conflicting privileges link. Remove link to the denoted object.

WAR(SVC350): Cannot evaluate matchrule '*string1*'. No value for parameter '*string2*' assigned!.

Explanation: The role parameter value is missing in the role assignment.

Action: Edit user, and enter a value for the role parameter.

ERR(SVC351): No role parameter value provided. Rule is '*string1*'.

Explanation: No value defined for the role match rule.

Action: Edit role and enter a match rule value.

ERR(SVC352): Unknown matchrule type '*string1*' in '*string2*'.

Explanation: The role match rule defines an unknown object type.

Action: Edit role and enter a correct object type for the match rule.

ERR(SVC353): Unknown operator '*string1*'.

Explanation: The operator defined in the role match rule is unknown.

Action: Edit role match rule, enter a correct operator.

ERR(SVC354): *string1* '*string2*' [*string3*] is not comparable!.

Explanation: Evaluation of the role parameter match rule fails since the role-param-value or the value is not comparable.

Action: Choose comparable Java classes for role-param-value and value.

WAR(SVC355): *string1*: No value!.

Explanation: The role parameter has no value assigned.

Action: Enter a correct value for the role parameter.

ERR(SVC356): *string1*: Illegal role parameter type '*string2*'!.

Explanation: A role parameter has an illegal type.

Action: Edit role parameter with the Identity Manager and choose a correct type from the proposal list.

WAR(SVC357): *string1*: Value '*string2*' out of range!.

Explanation: The role parameter has a value being out of range.

Action: Enter a correct value for the role parameter.

WAR(SVC358): Cannot evaluate matchrule '*string1*'of role *string2*: Role parameter *string3* does not exist.

Explanation: A role parameter does no longer exist. The user is flagged dxrIsInconsistent. During role resolution, the dependent privileges are removed.

Action: Check if removal of the parameter is OK. Remove assignment or create parameter again.

WAR(SVC359): Cannot evaluate matchrule '*string1*' of role *string2*: Role parameter value *string3* of parameter *string4* is invalid.

Explanation: The role parameter has a an invalid value. Thus, the user is marked inconsistent. The privileges affected by the value are removed in role resolution.

Action: Check if the missing value is correct. Otherwise, add value. Remove invalid value and/or assignment.

ERR(SVC360): Start of approval workflow for assignment of privilege DN=*string1* to subject DN=*string2* failed. Cannot lock assignment: *string3*.

Explanation: Unlikely to occur. Prior to starting the approval workflow an assignment has to be locked. Saving of the assignment failed.

Action: Check server connection and detailed message. Remove assignment and assign it again.

ERR(SVC361): Remote exception during start of approval workflow for assignment of privilege DN=*string1* to subject DN=*string2*: *string3*.

Explanation: During start of approval workflow, a remote exception has been thrown, indicating an error during communication to the approval engine.

Action: Check if the Java based identity server is running. Check approval workflow state.

ERR(SVC362): Start of approval workflow for assignment of privilege DN=*string1* to subject DN=*string2* failed: *string3*.

Explanation: Start of an approval workflow failed.

Action: Unassign privilege, quit user, and assign privilege again. Check log files of IDS-J server.

ERR(SVC363): Save of user DN=*string1* failed: *string2*. Approval workflow to subject DN=*string3* was NOT started.

Explanation: Start of an approval workflow failed.

Action: Unassign privilege, quit user, and assign privilege again.

ERR(SVC364): Illegal value '*string1*' of role parameter '*string2*' detected for user DN=*string3*.

Explanation: An assignment of a role with role parameter has inconsistent no role parameter values.

Action: Correct role assignment/assigned parameters.

ERR(SVC365): Role parameter *string1* for user DN=*string2* has no parameter value.

Explanation: An assignment of a role with role parameter has no role parameter value.

Action: Correct role assignment/assigned parameters.

WAR(SVC366): Exception *string1* occurred while checking constraints of role parameter *string2*: *string3*.

Explanation: The role parameter check ran on an exception.

Action: Correct role assignment/assigned parameters.

ERR(SVC367): No Role-Parameter values defined for role *string1* assigned to user DN=*string2*.

Explanation: The role has a role parameter but the assignment has no role parameter values.

Action: Correct role assignment/assigned parameters.

ERR(SVC370): Reading of object assigned to group DN=*string1* failed: *string2*. Cannot change state of imported member.

Explanation: The account referenced by the group via dxrGroupMemberImported

cannot be read. Thus, changing the state of the assignment failed.

Action: Check detailed message, server and network correction. Correct error and edit assignment again. Remove the entry manually from dxrGroupMemberImported in the data view.

ERR(SVC371): Saving of account DN=*string1* assigned to group DN=*string2* failed: *string3*. Cannot change state of imported member.

Explanation: The account referenced by the group via dxrGroupMemberImported cannot be saved. Thus, changing the state of the assignment failed.

Action: Check detailed message, server and network correction. Correct error and edit assignment again.

ERR(SVC372): Can't create order cn=*string1* for subject DN=*string2* in the data store: *string3*.

Explanation: Saving a new order to the data store failed.

Action: Check network and server connections. Check if the name space is exhausted. In that case, change the naming rule for the Order's attributes (superior, cn).

ERR(SVC373): Reading of property *string1* from master failed: *string1*.

Explanation: The property is defined as master property in the object description. The services fail to read its value from the master. Presumably the link given in the link attribute of the master definition is broken.

Action: Check and correct the link to the master object. Check detailed message and network and server connections.

WAR(SVC374): Illegal value '*string1*' given for assign.InitialSearch in dxi.cfg. Using default assign.InitialSearch=false.

Explanation: Entering assign.InitialSearch=true forces the services to perform an initial search before opening the assignment dialog or after pressing the edit button of the Identity Manager.

Action: Enter correct value true or false for option assign.InitialSearch in dxi.cfg.

WAR(SVC375): In dxi.cfg, aclmgr.refresh.interval must not exceed 24 hours. Using default value=600s.

Explanation: To speed up the AclManager's performance, its results are cached for a period being defined as value of aclmgr.refresh.interval in the Identity Managers configuration file dxi.cfg.

Action: Enter a valid value (in seconds) for aclmgr.refresh.interval (has to be less than 24 hours).

WAR(SVC376): Illegal value '*string1*' given for aclmgr.refresh.interval in dxi.cfg. Using default aclmgr.refresh.interval = 600s.

Explanation: The value assigned to aclmgr.refresh.interval in the Identity Managers configuration file dxi.cfg has to be an integer.

Action: Enter a valid value (in seconds) for aclmgr.refresh.interval (has to be less than 24 hours), e.g. aclmgr.refresh.interval=3600.

WAR(SVC377): Cannot evaluate matchrule '*string1*' of role *string2*: Role parameter value *string3* of parameter *string4* is invalid.

Explanation: The role parameter has a an invalid value. Thus, the user is marked inconsistent. The privileges affected by the value are removed in role resolution.

Action: Check if the missing value is correct. Otherwise, add value. Remove invalid value and/or assignment.

WAR(SVC378): Illegal value '*string1*' given for trace.level in dxi.cfg. Using default level=1.

Explanation: The value assigned to trace.level in the Identity Managers configuration file dxi.cfg has to be an integer ranging from 0 to 9.

Action: Enter a valid value for trace.level in dxi.cfg. Legal values range from 0 (no logging) to 9 (full logging).

WAR(SVC379): Illegal value '*string1*' given for trace.maxlines in dxi.cfg. Using default maxlines=5000.

Explanation: The value assigned to trace.maxlines in the Identity Managers configuration file dxi.cfg has to be an integer defining the maximum number of lines being written to one trace file. A value of 0 forces all messages to be written to one file.

Action: Enter a valid integer value for trace.maxlines in dxi.cfg.

WAR(SVC380): Illegal value '*string1*' given for cache.prefetch.size in dxi.cfg. Using default cache.prefetch.size='*string2*'.

Explanation: The value assigned to cache.prefetch.size in the Identity Managers configuration file dxi.cfg has to be an integer defining the size of the pre-fetch cache in bulk role resolution.

Action: Enter a valid integer value for cache.prefetch.size in dxi.cfg (e.g. 300).

WAR(SVC381): Illegal value '*string1*' given for batch.accumulator.size in dxi.cfg. Using default batch.accumulator.size='*string2*'.

Explanation: The value assigned to batch.accumulator.size in the Identity Managers configuration file dxi.cfg has to be an integer defining the size of the object cache used when searching users being affected from a privilege change.

Action: Enter a valid integer value for batch.accumulator.size in dxi.cfg (e.g. 500).

WAR(SVC382): Illegal value '*string1*' given for ldap.switch2offline in dxi.cfg. Using default ldap.switch2offline='*string2*'.

Explanation: The value assigned to ldap.switch2offline in the Identity Managers configuration file dxi.cfg has to be an integer defining the maximum number of affected objects (users, roles, permissions) in an online role resolution.

Action: Enter a valid integer value for ldap.switch2offline in dxi.cfg (e.g. 2000).

WAR(SVC383): Illegal value '*string1*' given for write.cache.enabled in dxi.cfg. Using default write.cache.enabled=true.

Explanation: The value assigned to write.cache.enabled in the Identity Managers configuration file dxi.cfg has to be true or false. The value of true speeds up bulk role resolution by using a write cache.

Action: Enter a valid boolean value for lwrite.cache.enabled (e.g. true).

WAR(SVC384): Illegal value '*string1*' given for ldap.filter.maxsize in dxi.cfg. Using default ldap.filter.maxsize='*string2*'.

Explanation: The value assigned to ldap.filter.maxsize in the Identity Managers configuration file dxi.cfg has to be integer defining the maximum size of automatically generated LDAP filters.

Action: Enter a valid integer value for ldap.filter.maxsize in dxi.cfg (e.g. 10000).

ERR(SVC385): *string1*.

Explanation: A generic error message that is issued from a Java script (save script or naming rule).

Action: Check the message details.

WAR(SVC386): *string1*.

Explanation: A generic warning that is issued from a Java script (save script or naming rule).

Action: Check the message details.

ERR(SVC387): The conflicting privilege DN=*string1* cannot be read from the datastore: *string2*. Check for broken link.

Explanation: The referenced privilege cannot be read from the datastore.

Action: Edit the conflicting privileges link. Remove link to the denoted object.

ERR(SVC388): Setting the checkSod flag at privilege DN=*string1* failed: Error during save: *string2*.

Explanation: The checkSod flag could not be set at the referenced privilege.

Action: Check the detailed error message and correct the error. Set flag manually in the data view.

ERR(SVC389): Removing the checkSod flag at privilege DN=*string1* failed: Error during save: *string2*.

Explanation: The checkSod flag could not be removed at the referenced privilege.

Action: Check the detailed error message and correct the error. Remove flag manually in the data view.

ERR(SVC390): Searching for active SoD policies referencing privilege DN=*string1* failed:

string2. checkSod flag not removed.

Explanation: The checkSod flag could not be removed at the referenced privilege.

Action: Check the detailed error message and correct the error. Remove flag manually in the data view.

ERR(SVC391): The following SoD policies are violated by subject DN=*string1* after adding new privileges:

Explanation: After adding new privileges, violations are detected in the SoD check.

Action: Save the new assignments to start SoD approval workflows. Remove the new assignment(s) to avoid SoD violations.

ERR(SVC392): SoD policy DN=*string1* is violated by the following set of privileges:

Explanation: The denoted SoD policy is violated by the set of privileges being assigned to the user.

Action: Save the new assignments to start SoD approval workflows. Remove the new assignment(s) to avoid SoD violations.

ERR(SVC393): SoD check aborted since policy is not active: DN=*string1*.

Explanation: The denoted SoD policy is not active. Thus, no checks are performed.

Action: Check isActive flag at the policy to activate checks.

ERR(SVC394): SoD check aborted since policy is not valid: Less than 2 conflicting privileges assigned: DN=*string1*.

Explanation: The denoted SoD policy is not valid, less than 2 conflicting privileges are assigned.

Action: Deactivate policy or assign at least two conflicting privileges.

ERR(SVC395): SoD violations found for the following users while checking policy DN=*string1*:

Explanation: SoD violations are listed for the given policy.

Action: Remove conflicting privileges or approve violations.

ERR(SVC396): '*string1*'.

Explanation: The user violates the SoD policy.

Action: Remove conflicting privileges or approve violations.

ERR(SVC397): Searching for users in SoD check failed: *string1*.

Explanation: The user violates the SoD policy.

Action: Remove conflicting privileges or approve violations.

ERR(SVC398): *string1*.

Explanation: The conflicting privileges of a SoD violation are shown.

Action: Remove conflicting privileges or approve violations.

ERR(SVC399): Remote exception during start of approval workflow for modifications of subject DN=*string1* failed: *string2*.

Explanation: During start of approval workflow, a remote exception has been thrown, indicating an error during communication to the approval engine.

Action: Check if the Java based identity server is running. Check approval workflow state.

ERR(SVC400): Start of approval workflow for modifications of subject DN=*string1* failed: *string2*.

Explanation: Start of an approval workflow failed.

Action: Quit user, and perform user modifications again.

ERR(SVC401): Resolving assignment uid=*string1* to role DN=*string2* to inherited privileges failed: *string3*.

Explanation: Resolving the role assignment failed. Thus, the inherited privileges are not determined and are not included into the SoD analysis.

Action: Reset user, check the detailed message and remove the error. Then, edit user again.

ERR(SVC402): Resolving assignment role DN=*string1* to inherited privileges failed: *string2*.

Explanation: Resolving the role assignment failed. Thus, the inherited privileges are not determined and are not included into the SoD analysis.

Action: Reset user, check the detailed message and remove the error. Then, edit user again.

ERR(SVC403): Resolving assignment permission DN=*string1* to inherited privileges failed: *string2*.

Explanation: Resolving the permission assignment failed. Thus, the inherited groups are not determined and are not included into the SoD analysis.

Action: Reset user, check the detailed message and remove the error. Then, edit user again.

ERR(SVC404): Cannot create SoD exception for user DN=*string1*, policy=*string2*: *string3*.

Explanation: Creation of a SoD exception failed for the denoted policy and user.

Action: Check the detailed message and remove the error condition.

WAR(SVC405): Segregation of duty checks are enabled, but online resolution is disabled. Enabling online resolution.

Explanation: Online resolution is a precondition for segregation of duty checks. Thus,

online resolution is enabled, despite it is disabled by the setting of the offlineResolution domain option.

Action: Disable 'Offline Resolution' or disable 'Segregation of duty (SoD) checks' at the domain root.

ERR(SVC406): Loading SoD exceptions for user DN=%s from the data store failed: %s.

Explanation: An LDAP error occurred while loading SoD Exceptions for the user from the datastore.

Action: Check detailed message. Correct error conditions and run checks again.

ERR(SVC407): Removal of SoD Exception DN=*string1* for user DN=*string2* failed: *string3*.

Explanation: The SoD exception is not valid any longer since conflicting privileges are removed from the user. An attempt to delete the SoD exception failed.

Action: Remove object manually.

ERR(SVC408): Removal of link DN=*string1* from SoD Exception DN=*string2* for user DN=*string3* failed: *string4*.

Explanation: A conflicting privilege is removed from the user and thus has to be removed from the SoD exception.

Action: Remove link manually from the exception object.

WAR(SVC409): Links have been removed from the policy. If you proceed, the affected SoD exceptions are removed.

Explanation: A privilege has been removed from the set of conflicting privileges. In case the policy is saved in this state, the affected SoD exceptions are removed.

Action: Check if removal of the link is correct. Removal and subsequent addition of the same privilege will require a new approval of all affected SoD violations.

WAR(SVC410): Removing the SoD exceptions affected by removal of link DN=*string1*.

Explanation: The privilege has been removed from the set of conflicting privileges. In case the policy is saved in this state, the affected SoD exceptions are removed.

Action: Check if removal of the link is correct. Removal and subsequent addition of the same privilege will require a new approval of all affected SoD violations.

WAR(SVC411): The policy has been deactivated. If you proceed, the affected SoD exceptions are removed.

Explanation: A policy has been deactivated. In case the policy is saved in this state, the affected SoD exceptions are removed.

Action: Check if removal of the link is correct. Removal and subsequent addition of the same privilege will require a new approval of all affected SoD violations.

ERR(SVC412): Searching for affected SoD exceptions failed for policy DN=*string1*: *string2*. Privilege links not removed.

Explanation: The search for SoD exceptions being affected by a change of the SoD policy has failed.

Action: Check detailed message and remove the error condition. Edit/delete the SoD exceptions located under the policy folder manually.

ERR(SVC413): Removal of links has failed for SoD exception DN=*string1*: *string2*.

Explanation: The approved privilege links cannot be removed from the SoD exception due to a save error.

Action: Check detailed message and remove the error condition. Edit/delete the SoD exception manually.

ERR(SVC414): Removal of SoD exception DN=*string1* failed: *string2*.

Explanation: The SoD exceptions cannot be deleted.

Action: Check detailed message and remove the SoD exception manually.

ERR(SVC415): Maximum recursion depth of *string1* exceeded during hierarchy analysis of role structure. Current privilege: *string2*.

Explanation: The role structure is too deep. Presumably a role cycle is created.

Action: Edit role structure with Identity Manager. Reduce depths or remove role cycle.

ERR(SVC416): Cannot evaluate matchrule for role DN=*string1*: No rule defined.

Explanation: The service layer evaluates a role parameter match that has no match rule.

Action: Should never occur. Check resolution result and call DirX Identity support in case of errors.

WAR(SVC417): Cannot evaluate matchrule for role DN=*string1*: No role parameter values found.

Explanation: A role matchrule cannot be evaluated since no role parameter values exist. This may occur in case a role with role parameters is added as a junior role to an existing role.

Action: Edit role assignments of the affected user. Add required role parameters.

ERR(SVC418): Role parameter *string1*: Inconsistent codes *string2* versus *string3* supplied for title *string4*.

Explanation: Different codes are provided for the same title of a role parameter.

Action: Correct the proposed values in the role parameters value list and/or the assigned proposal list.

ERR(SVC419): Role parameter *string1*: Inconsistent titles *string2* versus *string3* supplied for code *string4*.

Explanation: Different titles are provided for the same code of a role parameter.

Action: Correct the proposed values in the role parameters value list and/or the assigned proposal list.

ERR(SVC420): Error while reading role parameters: *string1*.

Explanation: An LDAP error is reported during read of the role parameters.

Action: Check the message details. Check LDAP server and network connections.

ERR(SVC421): No uid provided with role parameter DN=*string1*.

Explanation: A role parameter is read with empty uid. This is unlikely to occur, since the UID is automatically added during creation of the role parameter.

Action: Enter a uid for the parameter in the data view. Take the uid's value from existing assignments to a role with those parameters from the assignment object.

ERR(SVC422): Error in role parameter search: The object returned by the search is not a role parameter, DN=*string1*.

Explanation: This error indicates a problem with the object descriptions.

Action: Check if all object descriptions are correctly read during program start of e.g. the Identity Manager. Correct errors and restart the applications.

ERR(SVC423): Setting InApproval flag on assignment DN=*string1* failed: *string2*.

Explanation: A re-approval workflow has successfully started. Marking the assignment inApproval failed due to a save error of the assignment object. Thus, with the next run of the re-approval rule a new workflow is started.

Action: Check detailed message and correct server/network problems. Set dxrInApproval flag manually in the data view.

ERR(SVC424): Update of re-approval date for privilege DN=*string1* failed: *string2*.

Explanation: A re-approval date is configured at the privilege. The automatic update process failed to update the date.

Action: Check detailed message and correct server/network problems. Set re-approval date manually.

ERR(SVC425): Update of the domain's re-approval date failed: *string1*.

Explanation: A re-approval date is configured at the domain. The automatic update process failed to update the date.

Action: Check detailed message and correct server/network problems. Set re-approval date manually.

ERR(SVC426): Saving of deactivated policy DN=*string1* failed: *string2*. Cannot remove link to conflicting privilege DN=*string3*.

Explanation: A link to a conflicting privilege is removed from the policy and the isActive flag is reset. The changes make no effect since an error occurred during the subsequent

save of the object.

Action: Check detailed message and correct server/network problems. Remove link and deactivate policy manually.

ERR(SVC427): Saving of policy DN=*string1* failed: *string2*. Cannot remove link to conflicting privilege DN=*string3*.

Explanation: A link to a conflicting privilege is removed from the policy. The change makes no effect since an error occurred during the subsequent save of the object.

Action: Check detailed message and correct server/network problems. Remove link manually.

ERR(SVC428): Searching for active SoD policies referencing privilege DN=*string1* failed: *string2*. Cannot remove conflicting privilege links from the affected policies.

Explanation: Searching for SoD policies referencing the privilege by a conflictingPrivilegesLink failed.

Action: Check detailed message and correct server/network problems. Remove links manually or by using the LinkChecker agent.

ERR(SVC429): Calculation of the re-approval date from domain parameters failed: *string1*.

Explanation: Reading the domain object failed. Thus, the default settings are used to calculate the new re-approval date.

Action: Check detailed message and correct server/network problems.

ERR(SVC430): Loading of orders from running approval workflows failed for user DN=*string1*: *string2*.

Explanation: The system is trying to read the orders from the user's approval workflows being in state RUNNING. No orders are displayed due to the errors.

Action: Check detailed message and correct server/network problems.

ERR(SVC431): Cannot save SoD exception for user DN=*string1*, policy=*string2*: *string3*.

Explanation: Saving a SoD exception failed for the denoted policy and user.

Action: Check the detailed message and remove the error condition.

ERR(SVC432): SoD checks are not activated. No checks are performed.

Explanation: The SoD policy cannot be executed since the feature is not activated.

Action: Set dxrSodActive flag at the identity domain and run SoD check workflow.

ERR(SVC433): Remote exception during start of create workflow for subject DN=*string1*: *string2*.

Explanation: During start of create workflow, a remote exception has been thrown, indicating an error during communication to the IDS-J server.

Action: Check if the Java based identity server is running. Check create workflow state.

ERR(SVC434): Start of create workflow for subject DN=*string1* failed: *string2*.

Explanation: Start of a create workflow failed.

Action: Check log files of IDS-J server. Remove error condition and repeat creating the object.

ERR(SVC435): No matching workflow definition found to create *string1* DN=*string2*.

Explanation: No applicable workflow was found to create object of given type. Since dxrSaveUserIfNoWF is not checked at the domain, creation is aborted.

Action: Add a new create workflow. Check the when applicable definitions of you create workflows. Supply all required attributes (permission parameters) when creating the new object.

ERR(SVC436): Cannot cancel request workflow uid=*string1*: *string2*.

Explanation: The request workflow with the given UID cannot be aborted.

Action: Check detailed message. Check workflow state in the workflow monitor. Cancel workflow in Web Center.

ERR(SVC437): Error while creating order: *string1*.

Explanation: Creating an order failed. Should never occur.

Action: Check detailed message. Check installed jar files, especially dxmOrder.jar. Call your support.

ERR(SVC438): Error while removing assignment for user DN=*string1* to privilege DN=*string2*: *string3*.

Explanation: Checking if a rule assignment to be removed is in approval failed.

Action: Check detailed message. Check network and server connections.

ERR(SVC439): Cannot add assignment to privilege DN=*string1* for subject DN=*string2*: *string3*.

Explanation: An assignment could not be added. Loading of existing assignments failed.

Action: Check detailed message.

ERR(SVC440): Cannot remove *string1*=*string2* from group DN=*string3*: *string4*.

Explanation: Removal of a group member failed.

Action: Check detailed message. Remove member manually in the data view.

ERR(SVC441): Cannot change value of *string1* from *string2* to *string3* for group DN=*string4*: *string5*.

Explanation: Changing of a member reference failed.

Action: Check detailed message. Change member manually in the data view.

ERR(SVC442): Cannot refresh SoD exception: *string1*.

Explanation: Refreshing of a SoD exception failed. Unlikely to occur.

Action: Check detailed message.

ERR(SVC443): Cannot evaluate access policy for SoD exception DN=*string1*: *string2*.

Explanation: Evaluation of viewing policy of a SoD exception failed.

Action: Check detailed message.

ERR(SVC444): Parse error in dxrFilter attribute of proposal list DN=*string1*.

Explanation: Parsing the filter of a proposal list of type DN caused an error. Unlikely to occur.

Action: Edit filter in proposal list definition in Identity Manager. In case of problems, clear filter in the Data View, enter new filter in the Provisioning View.

ERR(SVC445): Proposal list search for DN=*string1* failed: *string2*.

Explanation: An error occurred during search of the objects building the tags of a DN type proposal list.

Action: Check the detailed message. Correct filter/searchbase. Check server and network connections.

ERR(SVC446): Saving of functional account DN=*string1* assigned to group DN=*string2* failed: *string3*.

Explanation: An error occurred during save of a functional account.

Action: Check the detailed message. Correct object descriptions/schema. Check server and network connections.

ERR(SVC447): Reading of functional account DN=*string1* assigned to group DN=*string2* failed. Object is of wrong type: *string3*.

Explanation: An error occurred while reading a functional account. The object is of wrong type.

Action: Check the detailed message. Correct object descriptions/schema. Correct the group's functional account assignment.

ERR(SVC448): Reading of functional account DN=*string1* assigned to group DN=*string2* failed: *string3*.

Explanation: An error occurred during read of a functional account.

Action: Check the detailed message. Correct the group's functional account assignment.

ERR(SVC449): Cycle detected in master mechanism: Removing link *string1=string2*.

Explanation: A cyclic dependency was detected in the master property mechanism. The affected link is removed.

Action: Set the link to the correct object. Remove message from TODO messages.

ERR(SVC450): Broken master link detected: Removing link *string1=string2*.

Explanation: A broken link was detected for a property that gets its value from a master. The broken link is removed.

Action: Set link to the correct object. Remove error message from the TODO messages.

ERR(SVC451): Cannot update group members for account DN=*string1* assigned to group DN=*string2*: *string3*.

Explanation: A dxrFunctionalAccount link is modified. Updating the group members according to that change failed.

Action: Correct functional account link at the group in Identity Manager. Check detailed message.

ERR(SVC452): Create of access right failed for operation=*string1*, type=*string2*: Access right container DN=*string3* does not exist.

Explanation: Creation of an access right for delegation failed. The parent folder to hold the right does not exist.

Action: Check for typing errors in dxrObjectType of the access policy definition. Create missing containers manually.

ERR(SVC453): Building access rights from access policies failed: *string1*.

Explanation: Creation of access rights for delegation failed.

Action: Check access policy definitions. Check detailed message.

ERR(SVC454): You are not allowed to read passwords from object DN=*string1*.

Explanation: The logged-in user has no access rights for reading the password attached to the denoted object.

Action: Call your access policy administrator to extend the access policy.

ERR(SVC455): You are not allowed to set a password for object DN=*string1*.

Explanation: The logged-in user has no access rights for setting a password for the denoted object.

Action: Call your access policy administrator to extend the access policy.

ERR(SVC456): Storage object for DN=*string1* could not be retrieved. Error message=*string2*.

Explanation: The object for the specified DN could not be retrieved.

Action: Please check the error message and the DN.

ERR(SVC457): The tagprovider DN=*string1* for property *string2* references object of wrong type. Must be proposal list.

Explanation: If the tagprovider class is of type Proposal or DependentProposal the DN must reference an object of type SvcProposal.

Action: Please check the property description.

ERR(SVC458): The storage object with DN=*string1* for link attribute *string2* could not be retrieved.

Explanation: The definition for the proposal list specifies a link attribute. One or more linked object could not be retrieved.

Action: Please check the definition of the proposal list.

ERR(SVC459): The specific property '*string1*' is selected. Note that this has no effect, since only the individual properties *string1*(<name>) are processed.
Please, select the individual properties.

Explanation: The selected specific property has no effect. Only individual specific properties are processed.

Action: Use the individual properties you want to approve, e.g. dxrRPValues(c), dxrOptions(accountowner),...

ERR(SVC460): The property '*string1*' is selected. Note that it has the master '*string2*'. Please, define an approval for the master instead.

Explanation: The selected property has a master definition. Thus, it does not male sense to approve its change, since the change is overwritten by the master when the order is applied.

Action: Deselect the property. Select the master property for approval.

ERR(SVC461): No object description exists with name '*string1*'.

Explanation: When processing an attribute policy, an object description name is configured that is not found.

Action: Create the missing object description.

ERR(SVC462): The selected property '*string1*' is handled in assignment approval.

Explanation: A property is selected for attribute approval that is used as a system property in assignment approval.

Action: Deselect the property. Configure approval for the related assignment.

ERR(SVC463): The selected property '*string1*' is a system property being set by the service layer automatically.

Explanation: A property is selected for attribute approval that is used as a system property.

Action: Deselect the property.

ERR(SVC464): The LDAP search for object with dxrUID=*string1* returned an error *string2*.

Explanation: An LDAP error occurred.

Action: Check the message details. Correct the cause for the error.

ERR(SVC465): The LDAP search for object with dxrUID=*string1* returns no objects.

Explanation: No object is found for the given uid. Presumably the object has been removed from the data store or the uid has been deleted.

Action: Run uid migration to create a uid for all objects.

ERR(SVC466): The LDAP search for object with dxrUID=*string1* returns multiple objects: DN=*string1*, DN=*string2*.

Explanation: Multiple objects are found for the given uid.

Action: Remove duplicate UIDs. Run uid migration to create a uid for all objects.

ERR(SVC467): The binary property '*string1*' is selected. Binary attributes are not supported.

Explanation: The selected property is a binary property (e.g. a jpeg photo or a certificate). Binary properties are not processed in the approval algorithms.

Action: Deselect the property.

ERR(SVC468): Sending of *string1*Event failed for Entry with DN='*string2*'.

Explanation: The entry with the given DN was created or changed in the DirX Identity domain. The change event could not be sent to the message server. The associated event driven workflow is not informed.

Action: Check the details of the message. Is the message server available? Is there enough space to accept new messages?

ERR(SVC469): *string1* when reading privilege with DN='*string2*': *string3*.

Explanation: A business object link from the user was changed. Therefore the service layer wants to read the BO inherited privileges.

Action: Check the LDAP entry with the given DN: Does it still exist? Is the object class correct?

ERR(SVC470): Only one of the properties 'dxrRPMatchRule' and 'dxrRoleParamLink' is selected. You have to select both attributes or none of it.

Explanation: If only one property is selected, the role parameter algorithms may be broken.

Action: Select both properties or deselect both properties.

ERR(SVC471): The property '*string1*' is relevant to TS configuration. It must not be selected for approval.

Explanation: A TS config relevant attribute is chosen for approval. The change, however,

will initiate a TS migration.

Action: Deselect the property.

ERR(SVC472): The account DN=*string1* is of wrong type *string2*. Check the account's object description.

Explanation: A search for accounts returns an object that is mapped to the wrong java class.

Action: Check and correct the mapping for the object in the related object descriptions.

ERR(SVC473): *string1* when decrypting password of length *string2* of user with DN='*string3*': *string4*.

Explanation: The service layer tried to decrypt / descramble the configured password before starting a request workflow.

Action: Check the password.

ERR(SVC474): Proposal list *string1*: Inconsistent codes *string2* versus *string3* supplied for title *string4*.

Explanation: Different codes are provided for the same title of a proposal list.

Action: Correct the proposed values in the proposal list.

ERR(SVC475): Proposal list *string1*: Inconsistent titles *string2* versus *string3* supplied for code *string4*.

Explanation: Different titles are provided for the same code of a proposal list.

Action: Correct the proposed values in the proposal list.

ERR(SVC476): Can't create order for subject DN=*string1*: *string2*.

Explanation: Creating a new order in memory failed.

Action: Check naming rules and JavaScripts.

ERR(SVC477): Cannot remove assignment from privilege DN=*string1* for subject DN=*string2*: *string3*.

Explanation: An assignment could not be removed. Loading of existing assignments failed.

Action: Check detailed message.

ERR(SVC478): Cannot change *string1* of assignment from user DN=*string2* to DN=*string3*: Operation not granted.

Explanation: The attempt of the logged-in user to change attributes of an assignment failed. The user has no grant policy to modify the assignment.

Action: Call your DirX Identity administrator to give you grant rights to the privilege.

ERR(SVC479): Loading of state machine '*string1*' failed for DN=*string2*: *string3*. Cannot

update the object's state.

Explanation: A custom state maching is configured but cannot be loaded by the software.

Action: Correct configuration and installation.

ERR(SVC480): Cannot retrieve state of associated user DN=*string1* of persona DN=*string2*.

Explanation: The associated user's state cannot be retrieved.

Action: Check message details. Delete persona manually in case the associated entry is in TBDEL state.

ERR(SVC481): Cannot read associated user DN=*string1* for functional user DN=*string2*: *string3*.

Explanation: An error occurred while reading the associated user for the functional user.

Action: Check detailed message, check for broken sponsor link.

ERR(SVC482): Resolving persona DN=*string1* for user DN=*string2* failed: *string3*.

Explanation: An error occurred while performing a state update/role resolution of a persona.

Action: Check message details. Correct errors in persona.

ERR(SVC483): Resolving functional user DN=*string1* for user DN=*string2* after removal of link '*string3*' failed: *string4*.

Explanation: An error occurred while performing a state update/role resolution of a functional user.

Action: Check message details. Correct errors in the functional user. Assign a new sponsor to the user.

ERR(SVC484): Cannot read owner DN=*string1* of persona DN=*string2* from the data store: '*string3*'.

Explanation: Cannot exchange persona and user, since the persona's owner cannot be read.

Action: Check and correct the persona's owner link.

ERR(SVC485): Cannot read JavaScript object DN=*string1* from the data store: '*string2*'.

Explanation: Cannot exchange persona and user, since the JavaScript to be executed cannot be read.

Action: Provide a JavaScript for exchanging user and persona. Check the JavaScript's DN.

ERR(SVC486): Cannot read persona DN=*string1* from the data store: '*string2*'.

Explanation: Cannot exchange persona and user, since the persona to be exchanged

cannot be read.

Action: Check message details. Check existence of the persona.

ERR(SVC487): Execution of JavaScript DN=*string1* failed: '*string2*'.

Explanation: Cannot exchange persona and user because of a JavaScript error.

Action: Check message details. Correct your JavaScript syntax.

ERR(SVC488): Verification that object DN=*string1* is a user failed: '*string2*'.

Explanation: An error occurred during save of a persona that has been converted to a user. An additional verification of the object type 'user' has also failed.

Action: Check message details. Correct user manually.

ERR(SVC489): Verification that object DN=*string1* is a persona failed: '*string2*'.

Explanation: An error occurred during save of a user that has been converted to a persona. An additional verification of the object type 'persona' has also failed.

Action: Check message details. Correct persona manually.

ERR(SVC490): Exchanging object descriptors of user '*string1*' and persona '*string2*' failed: '*string3*'.

Explanation: Cannot exchange persona and user, because the exchange of its object descriptors failed.

Action: Check message details. Correct objects manually.

ERR(SVC491): Exchanging user '*string1*' and persona '*string2*' failed: '*string3*'.

Explanation: Cannot exchange persona and user.

Action: Check message details. Exchange objects manually.

ERR(SVC492): Saving of persona DN='*string1*' failed: '*string2*'. Cannot remove dxrPersona from object class.

Explanation: Cannot exchange persona and user because removing the dxrPersona object class failed.

Action: Check message details. Exchange objects manually.

ERR(SVC493): Illegal syntax in userAttributesToCopyWithNameMapping: '*string1*', expecting <source>.<target>. Expression ignored.

Explanation: In the AccountToPersona JavaScript, an illegal expression is is found. The expression is ignored when copying user attributes to the persona.

Action: Change the expression that is contains the source name, a dot as separator, and then the target name.

ERR(SVC494): Illegal syntax in accountAttributesToCopyWithNameMapping: '*string1*', expecting <source>.<target>. Expression ignored.

Explanation: In the AccountToPersona JavaScript, an illegal expression is is found. The expression is ignored when copying account attributes to the persona.

Action: Change the expression that is contains the source name, a dot as separator, and then the target name.

ERR(SVC495): Assigning group '*string1*' to *string2* '*string3*' failed: '*string4*'.

Explanation: Cannot create direct group assignment for persona/user/functional user.

Action: Check message details. Assign group manually.

ERR(SVC496): Assigning groups from account '*string1*' to *string2* '*string3*' failed: '*string4*'.

Explanation: Cannot create direct group assignments for a persona/user/functional user.

Action: Check message details. Assign group manually.

ERR(SVC497): Creating the resource orders failed for *string1* '*string2*': '*string3*'. No workflow started.

Explanation: Cannot create the resource orders for the direct assigned groups.

Action: Check message details. Correct the errors and run rule again.

ERR(SVC498): Creating the subject order failed for creation of *string1* '*string2*'. No workflow started.

Explanation: Cannot create the add order for the persona, user or functional user.

Action: Check message details. Correct the errors and run rule again.

ERR(SVC499): Wrong type *string1* of privilege specified for *string2*.

Explanation: An invalid privilege type was used for described object.

Action: Inform DirX Identity support.

ERR(SVC50): Cannot connect to LDAP server *string1*:_string2_.

Explanation: Connection to the LDAP server failed for the given host:port address.

Action: Check connection parameters: Host, port, and bind credentials.

INF(SVC500): User to group resolution started for user DN=*string1* at *string2*.

Explanation: The system informs on starting of a role resolution for the user.

Action: None.

INF(SVC506): User *string1* has exactly one valid account in target system *string2*. Marking account primary.

Explanation: The user has exactly one account in a TS, but has no primary account. Thus, this accounts taken as primary account.

Action: None.

INF(SVC520): New account *string1* successfully reserved in target system *string2*.

Explanation: A new account has been reserved for a user.

Action: None.

INF(SVC521): Adding group *string1* / account *string2* of target system *string3* to new resolved assignments.

Explanation: An new account/group-pair is assigned to the user during role resolution.

Action: None.

INF(SVC522): Setting account DN=*string1* for user DN=*string2* to state '*string3*'.

Explanation: Role resolution informs on the state being calculated for an account.

Action: None.

INF(SVC523): Adding group *string1* / account *string2* of target system *string3* to new direct assignments.

Explanation: A new account is created for a direct assigned group.

Action: None.

INF(SVC524): Distribution list DN=*string1* is up-to-date, nothing changed.

Explanation: A distribution list has been checked, it is up-to-date.

Action: None.

INF(SVC525): Updating distribution list DN=*string1*. Applying changes.

Explanation: A distribution list has been checked, it is being updated.

Action: None.

INF(SVC526): Adding user *string1* as a unique member to group *string2*.

Explanation: The user is added to the group's unique member attribute, i.e. is added to the distribution list.

Action: None.

INF(SVC527): Removing user *string1* from the unique members of group *string2*.

Explanation: The user is removed from the group's unique member attribute, i.e. is added to the distribution list.

Action: None.

ERR(SVC53): LDAP Unbind failed: *string1*.

Explanation: Unbind from the server failed. Presumably the connection was lost or the

server is down.

Action: None.

INF(SVC530): Missing values for attributes: *string1*.

Explanation: The check for completeness of an object has detected missing attributes. The message is added to the objects toDo messages.

Action: Edit the object and enter missing values. In case the object is a target system account and the value is filled from the assigned user, edit user and enter missing attributes there.

INF(SVC531): More than one peer account exists for account DN=*string1*.

Explanation: An account in a target system being hosted by a peer TS has more than one peer accounts.

Action: Set peer account link manually in the data view.

INF(SVC532): No peer account exists for account DN=*string1*.

Explanation: The user does have no accounts in the peer TS of the account's target system.

Action: Assign a privilege leading to a peer account for that user.

INF(SVC533): Assigning peer account DN=*string1* to account DN=*string2*.

Explanation: Exactly one account exists in the peer target system. It is assigned to the account.

Action: None.

INF(SVC536): Assignment from user DN=*string1* to privilege DN=*string2* successfully initialized for re-approval.

Explanation: An assignment has been initialized for re-approval: Its end date is set and it is marked for re-approval.

Action: None.

INF(SVC537): Initialization of assignment for re-approval from user DN=*string1* to privilege DN=*string2* simulated.

Explanation: The policy agent is running in simulate mode, thus the assignment is not modified.

Action: None.

INF(SVC538): SoD check for added privileges passed for subject DN=*string1*: No added privilege requires SoD checking.

Explanation: None of the newly assigned privileges requires SoD checking.

Action: None.

INF(SVC539): SoD check for added privileges passed for subject DN=*string1*: Only one privilege requires SoD checking.

Explanation: Only one of the added privileges and none of the existing privileges requires SoD checking.

Action: None.

ERR(SVC54): LDAP read of entry DN=*string1* failed: *string2*.

Explanation: The entry could not be read from the LDAP server. Maybe it has been deleted by another client or the server is not reachable.

Action: Check the reason described in the detailed message. Check server and network connections.

INF(SVC540): SoD delta check passed for subject DN=*string1*: No SoD violations detected from new versus existing privileges.

Explanation: No violations found in SoD check.

Action: None.

INF(SVC541): User DN=*string1* has an SoD violation to SoD policy DN=*string2* that has been approved.

Explanation: A SoD violation exists for the given user that has been approved.

Action: None.

INF(SVC542): A create workflow is already running for *string1 string2*. Workflow start aborted.

Explanation: A create workflow is running for the persona/user/functional user. No second workflow is started.

Action: None.

INF(SVC543): During delete of user *string1* depending *string2 string3* was implicitly deleted.

Explanation: A user is to be deleted. As there are personas for this user these personas are deleted also.

Action: None.

INF(SVC544): During delete of user *string1* depending *string2 string3* was set to state DELETED.

Explanation: A user is to be deleted. As there are personas for this user these personas are set to state DELETED.

Action: None.

ERR(SVC55): LDAP search failed for filter='*string1*', search root='*string2*': *string3*.

Explanation: An error occurred while searching the directory.

Action: Check the reason described in the detailed message. Check server and network connections.

ERR(SVC56): Error on LDAP function *string1*: *string2*.

Explanation: An LDAP error occurred during removal of an assignment.

Action: Check the reason described in the detailed message. Check server and network connections.

ERR(SVC58): LDAP delete failed for DN=*string1*: *string2*.

Explanation: Could not delete the object in the datastore.

Action: Check if object exists and retry deletion. Check detailed message, server and network connections.

WAR(SVC700): Role DN=*string1* has neither junior roles nor permissions.

Explanation: The role has neither junior roles nor permissions assigned. To make an effect in role resolution, at least one junior role or permission assignment is needed.

Action: Assign junior role(s)/permission(s) to the role.

WAR(SVC701): No groups are assigned to user DN=*string1* via permission DN=*string2*: No groups are attached to permission.

Explanation: The permission has no groups assigned. To make an effect in role resolution, at least one target system group assignment is needed.

Action: Assign group(s) to the permission.

WAR(SVC702): No groups are assigned to user DN=*string1* via permission DN=*string2*: No group matches.

Explanation: Evaluation of matchrules did not result in a resolved group for the user.

Action: Check design of role system and the user's attributes.

INF(SVC703): No groups are assigned to the user DN=*string1* by his assigned roles.

Explanation: Role resolution does not result in assigned groups for a user.

Action: Check design of role system and the user's attributes.

INF(SVC704): Assignment for User DN=*string1* to *string2* DN=*string3* is not active. Ignoring *string4*.

Explanation: The user has a direct privilege assignment being out-of-date.

Action: Check dates of assignment, edit user and change dates or remove assignment.

WAR(SVC706): Cannot map object with DN=*string1* to any MetaRole class. Skipping processing of object.

Explanation: An object is read from the LDAP server that does not map to any object

description. It is not considered a DirX Identity object and thus it is filtered.

Action: Check the object's attributes and objectClasses. Check the correctness of object descriptions.

WAR(SVC707): Permission DN=*string1*: User attribute '*string2*' is not a valid permission parameter.

Explanation: The user attribute defined in a matchrule is not defined in the set of permission parameters.

Action: Modify attribute or add it to the permission parameters in the domain object.

WAR(SVC708): Permission DN=*string1*: Group attribute '*string2*' is not a valid permission parameter.

Explanation: The group attribute defined in a matchrule is not defined in the set of permission parameters.

Action: Modify attribute or add it to the permission parameters in the domain object.

WAR(SVC709): Assigned role DN=*string1* is not flagged user assignable.

Explanation: A role is assigned to a user that is not flagged being user assignable.

Action: Set the role's user assignment possible flag or remove role assignment from the user.

WAR(SVC710): User DN=*string1* has assigned role DN=*string2* that is not flagged user assignable.

Explanation: A role is assigned to a user that is not flagged being user assignable.

Action: Set the role's user assignment possible flag or remove role assignment from the user.

WAR(SVC711): Sizelimit exceeded while performing search: Filter=*string1*, base=*string2*.

Explanation: A search hits the size limit. Thus, only a reduced set of objects is returned.

Action: Use more precise filter conditions to get a smaller search result. Increase limit.

WAR(SVC712): Timelimit exceeded while performing search: Filter=*string1*, base=*string2*.

Explanation: A search hits the time limit. Thus, only a reduced set of objects is returned.

Action: Use more precise filter conditions to get a smaller search result. Check if indexing improves performance. Increase limit.

WAR(SVC713): Modify of group DN=*string1* failed: *string2*. Refreshing and retrying.

Explanation: Applying the modifications on group members failed. Possibly another client did perform changes in parallel.

Action: None.

WAR(SVC714): Cannot search for assigned groups of account DN=*string1*: Assign attribute

string2 is empty.

Explanation: The assigned groups of an account cannot be searched, since the attribute being used as reference is empty.

Action: Add a unique value for the account's assign attribute manually and re-run role resolution.

WAR(SVC715): Group DN=*string1* references the current user DN=*string2* by 'uniqueMember' in a TS that needs accounts!.

Explanation: The resolved group references a user by its uniqueMember attribute. However, the TS needs account, i.e. the reference chain user->account->group has to be used.

Action: Remove user manually from the group's uniqueMember attribute.

WAR(SVC717): Group DN=*string1* is marked as extension group in a TS being not configured for extension groups.

Explanation: Decide if you want to use the extension group feature in the TS.

Action: As a result, configure extension groups or remove extension group flags.

WAR(SVC718): Cannot read object with DN=*string1* being conform to the extension group naming rule: *string2*.

Explanation: An error occurred while trying to read an extension group.

Action: Check detailed message and correct errors. Check server/network connections.

WAR(SVC719): Property *string1* for multi-valued master is NOT multi-valued in object description *string2*.

Explanation: The property is defined single valued. Thus, only the first value is taken.

Action: Correct schema and object descriptions.

WAR(SVC720): User DN=*string1* has assigned permission DN=*string2* that is not flagged user assignable.

Explanation: A permission is assigned to a user that is not flagged being user assignable.

Action: Set the permission's user assignment possible flag or remove role assignment from the user.

WAR(SVC722): Did not find account DN=*string1* in any member of group DN=*string2*.

Explanation: A search of assigned accounts returns an account not being referenced in the group members.

Action: Inform DirX Identity support.

WAR(SVC730): Assignment from account DN=*string1* to group *string2* in state *string3* is removed. States are not supported in new TS configuration.

Explanation: During target system migration, states are removed. Thus, the assignment is removed, too.

Action: None.

WAR(SVC731): Assignment from group DN=*string1* to account *string2* in state *string3* is removed. States are not supported in new TS configuration.

Explanation: During target system migration, states are removed. Thus, the assignment is removed, too.

Action: None.

WAR(SVC732): Assignment from group DN=*string1* to group *string2* in state *string3* is removed. States are not supported in new TS configuration.

Explanation: During target system migration, states are removed. Thus, the assignment is removed, too.

Action: None.

WAR(SVC733): Configuration inconsistent for TS DN=*string1*: Extra large groups AND reference group from account. Assuming 'reference group from account'.

Explanation: The XXL groups mechanism only exists in case the accounts are referenced by the groups.

Action: Uncheck 'has XXL groups' or 'reference group from account'.

WAR(SVC734): Illegal value '*string1*' given for report.sizelimit. Using default report.sizelimit='*string2*'.

Explanation: The value assigned to report.sizelimit has to be an integer defining the maximum number of affected objects (users, roles, permissions) in a report.

Action: Enter an integer value (e.g. 2000).

WAR(SVC735): User *string1* has more than one account in target system *string2* none of which is marked primary.

Explanation: DirX Identity cannot determine the primary account. Thus, no account is managed by DirX Identity in this target system.

Action: Select a primary account by checking the account's 'Primary account' flag.

ERR(SVC741): Reading IDS-J SOAP configuration from connectivity LDAP server failed.

Explanation: The request workflow endpoint could not be created from the IDS-J server configuration. Thus, no request workflow is started.

Action: Check server and network connection to Java based identity server. Check preceding messages and correct configuration errors.

ERR(SVC742): Processing of SoD Exceptions failed: Object DN=*string1* is of wrong type.

Explanation: A search for SoD Exceptions returns an object of wrong type.

Action: Check if all object descriptions are correctly loaded. Check if Professional license is in use.

ERR(SVC743): No matching workflow definition found to create *string1* DN=*string2*.

Explanation: No applicable workflow was found to create object of given type. Since dxrSaveUserIfNoWF is checked at the domain, the user is directly written to LDAP.

Action: None.

ERR(SVC744): Error tracking changes: Cannot load class *string1*: *string2*.

Explanation: The class defined by its full qualified class name cannot be loaded. Its changes cannot be tracked.

Action: Check for class loading problems (location and version of dxrServices.jar).

INF(SVC745): A delete workflow is running!

Explanation: A delete workflow is running for the object.

Action: None.

ERR(SVC800): Resolution of role hierarchy failed for user DN=*string1*.

Explanation: Errors occurred during role resolution.

Action: Check for broken links, correct the errors reported previously, and re-run role resolution.

ERR(SVC801): Segregation of duties violated for user DN=*string1*.

Explanation: Not yet implemented.

Action: None.

ERR(SVC802): Errors during resolution of assigned permissions of user DN=*string1*.

Explanation: Errors occurred while resolving the inherited and/or direct permissions assigned to the user.

Action: Check for broken links, correct the errors reported previously, and re-run role resolution.

ERR(SVC803): Errors during resolution from permissions to assigned groups of user DN=*string1*.

Explanation: Errors occurred while resolving the inherited and/or direct groups assigned to the user.

Action: Check for broken links, correct the errors reported previously, and re-run role resolution.

ERR(SVC804): Resolving permissions of user DN=*string1* failed: Cannot read permission DN=*string2* of role DN=*string3* from the data store.

Explanation: Reading of a permission failed during role resolution.

Action: Correct broken links in the role structure and re-run role resolution.

ERR(SVC805): Resolving role hierarchy of user DN=*string1* failed: Cannot read junior role DN=*string2* of role DN=*string3* from the data store.

Explanation: Reading of a junior role failed during role resolution.

Action: Correct broken links in the role structure and re-run role resolution.

ERR(SVC806): Cannot evaluate permission DN=*string1*: Reading of group DN=*string2* failed: *string3*.

Explanation: Reading of a group failed during role resolution.

Action: Check the detailed message. Remove group assignment or check LDAP server connection and re-run role resolution.

ERR(SVC807): Error in permission DN=*string1*: *string2*.

Explanation: Evaluation of a match rule failed during role resolution.

Action: Correct match rule and re-run role resolution.

ERR(SVC808): Resolving permissions of user DN=*string1* failed: Permission DN=*string2* of role DN=*string3* has the wrong object class.

Explanation: The permission assignment references an object of wrong type.

Action: Check the permission assignments at the role. Check if the object descriptions are correctly loaded. Re-run role resolution.

ERR(SVC809): Resolving groups of user DN=*string1* failed: Group DN=*string2* of permission DN=*string3* has the wrong object class.

Explanation: The group assignment references an object of wrong type.

Action: Check the group assignments at the permission. Check if the object descriptions are correctly loaded. Re-run role resolution.

ERR(SVC810): Resolving role hierarchy of user DN=*string1* failed: Junior role DN=*string2* of role DN=*string3* has the wrong object class.

Explanation: The junior role assignment references an object of wrong type.

Action: Check the junior role assignments of the role. Check if the object descriptions are correctly loaded. Re-run role resolution.

ERR(SVC811): Loading junior role DN=*string1* of role DN=*string2* failed: Junior role has the wrong object class.

Explanation: The junior role assignment references an object of wrong type.

Action: Check the junior role assignments of the role. Check if the object descriptions are correctly loaded.

ERR(SVC812): Loading junior role DN=*string1* of role DN=*string2* failed: *string3*.

Explanation: Reading of a junior role failed.

Action: Correct broken links in the role structure.

ERR(SVC813): Cannot create account *string1* for user DN=*string2* in TS DN=*string3*: *string4*.

Explanation: An account could not be created for the user during role resolution.

Action: Check the detailed message. In case the value range is exceeded, define a naming rule for the cn allowing multiple values or create an account manually.

ERR(SVC814): Resolution of user DN=*string1* failed: Cannot load assigned role DN=*string2*: *string3*.

Explanation: Reading of an assigned role failed during role resolution.

Action: Check the detailed message. Remove role assignment or check LDAP server connection and re-run role resolution.

ERR(SVC815): Cannot process direct assigned group DN=*string1*: Load failed: *string2*.

Explanation: Reading of a direct assigned group failed during role resolution.

Action: Check the detailed message. Remove group assignment or check LDAP server connection and re-run role resolution.

ERR(SVC816): Cannot add attribute '*string1*' of user DN=*string2* to distribution list group DN=*string3*: Attribute is empty.

Explanation: The user's attribute to be added to a distribution list is empty. The user is not added to the list.

Action: Edit user and provide a value for the attribute.

ERR(SVC817): Value *string1* of user attribute *string2* is not unique in distribution list DN=*string3*: Users are: *string4*.

Explanation: The user's attribute to be added to a distribution list already exists for another user.

Action: Edit users, enter unique values.

ERR(SVC818): Update of distribution list DN=*string1* failed: Error during save: *string2*.

Explanation: An error occurred during save of a distribution list group.

Action: Check the detailed message. Check LDAP server connection and re-run role resolution.

ERR(SVC819): Resolution of user DN=*string1* failed: Cannot load assigned permission DN=*string2*: *string3*.

Explanation: Reading of an assigned permission failed during role resolution.

Action: Check the detailed message. Remove permission assignment or check LDAP

server connection and re-run role resolution.

ERR(SVC820): Error while loading accounts for user DN=*string1* from the data store: *string2*.

Explanation: Error(s) during load of assigned accounts for a user.

Action: Check detailed message. Check network and server connection.

ERR(SVC821): Error while loading groups for account *string1* from the data store: *string2*.

Explanation: Error(s) during load of assigned groups for an account.

Action: Check detailed message. Check network and server connection.

ERR(SVC822): Can’t read target system DN=%s of account %s from the data store: %s.

Explanation: The account’s target system, holding configuration parameters and properties, could not be read.

Action: Check the detailed message. Check and correct object descriptions for the target system. Check for exceptions in the Manager log during startup.

ERR(SVC825): Error in structure of data store: Cannot extract DN of target system from account DN=*string1*.

Explanation: The account is not member of a target system. Or the target system is not recognized.

Action: Check and correct object descriptions for the target system. Check for exceptions in the Manager log during startup.

ERR(SVC827): User *string1* has more than one primary account in target system *string2*.

Explanation: DirX Identity handles one primary account per user and target system.

Action: Choose one account being primary account. Uncheck the 'Primary account' flag for the other accounts.

ERR(SVC828): User *string1* has more than one account in target system *string2* none of which is marked primary.

Explanation: DirX Identity cannot determine the primary account. Thus, no account is managed by DirX Identity in this target system.

Action: Select a primary account by checking the account’s 'Primary account' flag.

ERR(SVC830): Cannot delete RESERVED account DN=*string1* for user DN=*string2*: *string3*.

Explanation: A temporarily created account cannot be deleted when the user is reset or next role resolution is run.

Action: Check the detailed message. Delete the RESERVED account manually.

ERR(SVC831): Modify of account DN=*string1* failed for user DN=*string2*: *string3*.

Explanation: An account cannot be modified in the datastore.

Action: Check the detailed message. In case of LDAP errors, check your object descriptions. Check if descriptions for user and account are consistent for master properties.

ERR(SVC832): Modify of group DN=*string1* failed for user DN=*string2*: *string3*.

Explanation: A group cannot be modified in the datastore.

Action: Check the detailed message.

ERR(SVC833): Unknown state *string1* for account *string2*.

Explanation: An account has an incorrect state.

Action: Check source of wrong state and correct account manually. Allowed states are ENABLED, DISABLED, DELETED, and IMPORTED.

ERR(SVC834): Loading target system of group DN=*string1* failed: target system object DN=*string2* has the wrong object class.

Explanation: Errors occurred during read of a the target system.

Action: Check the detailed message. Check if the group is located correctly in a target system. Check and correct object descriptions for the target system. Check for exceptions in the Manager log during startup.

ERR(SVC835): Loading target system of account DN=*string1* failed: target system object DN=*string2* has the wrong object class.

Explanation: Errors occurred during read of a the target system.

Action: Check the detailed message. Check if the account is located correctly in a target system. Check and correct object descriptions for the target system. Check for exceptions in the Manager log during startup.

ERR(SVC836): User *string1* has no primary account in target system *string2*.

Explanation: The user has no valid primary account in the target system.

Action: Create a new primary account manually using the Identity Manager.

ERR(SVC837): Cannot mark account DN=*string1* primary: Save failed: *string2*.

Explanation: A user just has one valid account in a target system. The system tries to mark it being the primary account, but save failed.

Action: Check detailed message, network- and server connections. Set the account's 'Primary account' flag manually.

ERR(SVC838): Cannot read configuration data from target system of *string1* DN=*string2*: *string3*.

Explanation: Errors occurred during read data from the target system.

Action: Check the detailed message. Check and correct object descriptions for the target system. Check for exceptions in the Manager log during startup.

ERR(SVC839): Error while loading groups for user DN=*string1* from the data store: *string2*.

Explanation: Error(s) during load of assigned groups for a user.

Action: Check detailed message. Check network and server connection.

ERR(SVC840): Cannot add new *string1*To*string2* assignment object DN=*string3* to the data store: *string4*.

Explanation: The user has a new attributed assignment that cannot be added to the datastore.

Action: Check detailed message. Check network and server connection.

ERR(SVC841): Cannot delete *string1*To*string2* assignment object DN=*string3* from the data store: *string4*.

Explanation: The user had an attributed assignment. The assignment or all its attributes are removed, but deletion of the assignment object in the datastore failed.

Action: Check detailed message. Check network and server connection.

ERR(SVC842): Cannot modify *string1*To*string2* assignment object DN=*string3* in the data store: *string4*.

Explanation: The user has a modified attributed assignment that cannot be saved to the datastore.

Action: Check detailed message. Check network and server connection.

ERR(SVC843): Error while loading *string1*To*string2* assignment objects for user DN=*string3* from the data store: *string4*.

Explanation: Error(s) during load of attributes assignments from the datastore.

Action: Check detailed message. Check network and server connection.

ERR(SVC844): Error in *string1*To*string2* assignment for DN=*string3*: Loading referenced object DN=*string4* failed: *string5*.

Explanation: The assignment references an object that cannot be loaded from the datastore.

Action: Check detailed message. Check for broken link. Replace assignment by removing assignment in the identity manager or in the web center and adding a new assignment.

ERR(SVC845): Error in *string1*To*string2* assignment for DN=*string3*: Referenced object DN=*string4* is of wrong type. Check link.

Explanation: The assignment references an object of wrong type.

Action: Check the DN reference. In case the reference is correct, check if object descriptions are correct and correctly loaded. Replace assignment by removing assignment in the identity manager or in the web center and adding a new assignment.

ERR(SVC846): Error in *string1*To*string2* assignment for DN=*string3*: Illegal value for DN-reference detected (null or empty), assignment is corrupted.

Explanation: The reference to the assigned privilege is missing. Should not occur in case the assignments are edited using DirX Identity clients.

Action: Replace assignment by removing assignment in the identity manager or in the web center and adding a new assignment.

ERR(SVC847): Permission DN=*string1*: Illegal matchrule *string2*: Mandatory user attribute is missing.

Explanation: The user attribute is missing in a matchrule.

Action: Load permission and correct matchrule in the identity manager or in the web center.

ERR(SVC848): Permission DN=*string1*: Illegal matchrule '*string2*': Mandatory group attribute/const value is missing.

Explanation: The group attribute or constant expression is missing in a matchrule.

Action: Load permission and correct matchrule in the identity manager or in the web center.

ERR(SVC849): Permission DN=*string1*: Use of matchrules is not possible since no permission parameters are defined in the domain object, DN=*string2*.

Explanation: The attributes being used in match rules have to be declared as permission parameters at the domain object.

Action: Remove all matchrules or edit the permission parameters in the identity manager.

ERR(SVC850): Cannot read dxrRoleParams from domain object, DN=*string1*: *string2*.

Explanation: The permission parameters cannot be read from the datastore.

Action: Check detailed message. Check network and server connection.

ERR(SVC851): Error while scanning matchrule: *string1*.

Explanation: A permission’s matchrule is corrupt.

Action: Edit permission with the Web Center or the Identity Manager and correct the corrupt matchrule.

ERR(SVC852): Error in matchrule '*string1*': <pos> has to be an integer number.

Explanation: The position number of the matchrule is missing or not an integer.

Action: Edit permission with the Web Center or the Identity Manager and correct the corrupt matchrule.

ERR(SVC853): Error in matchrule '*string1*': Unknown operator '*string2*'.

Explanation: A permission's matchrule has an unknown operator.

Action: Edit permission with the Web Center or the Identity Manager and correct the corrupt matchrule.

ERR(SVC854): Error in matchrule '*string1*': User statement is missing.

Explanation: A permission's matchrule is corrupt.

Action: Edit permission with the Web Center or the Identity Manager and correct the corrupt matchrule.

ERR(SVC855): Error in matchrule '*string1*': No Group or Const statement found.

Explanation: A permission's matchrule is corrupt.

Action: Edit permission with the Web Center or the Identity Manager and correct the corrupt matchrule.

ERR(SVC856): Error in matchrule '*string1*': Rule is incomplete.

Explanation: A permission's matchrule is corrupt.

Action: Edit permission with the Web Center or the Identity Manager and correct the corrupt matchrule.

ERR(SVC857): Initialization of masters failed: *string1*.

Explanation: The master property mechanism is mainly used to synchronize properties from the user to the account. Possible errors are different or missing object descriptions for user/account for a master property.

Action: Check detailed message. Correct object descriptions.

ERR(SVC858): Error(s) occurred during save of assignments of user DN=*string1*: *string2*.

Explanation: The assignments of a user cannot be saved to the datastore.

Action: Check detailed message. Check network and server connection. Remove errors and resolve user again.

ERR(SVC859): Error(s) occurred during save of user DN=*string1*: *string2*.

Explanation: The user cannot be saved to the datastore.

Action: Check detailed message. Check network and server connection. Remove errors and resolve user again.

ERR(SVC860): Permission DN=*string1* is invalid: It has no groups assigned.

Explanation: A permission has no groups assigned. Thus, no access right are granted via this permission.

Action: Assign groups to the permission. Remove permission in case it is not used.

ERR(SVC862): Role DN=*string1* is not valid: Neither junior roles nor permissions are assigned.

Explanation: A role has no junior roles or permissions assigned. Thus, no access right are granted via this role.

Action: Assign junior role(s)/permission(s) to the role. Remove role in case it is not used.

ERR(SVC863): Group DN=*string1* is invalid: It is not referenced by any permission.

Explanation: A group has no permissions assigned. Thus, it is not integrated in the role system.

Action: Create a permission for the group. Ignore message in case you only work with direct group assignments.

ERR(SVC864): Cannot mark object DN=*string1* 'toBeAnalyzed': Save failed: *string2*.

Explanation: The dxrTBA flag cannot be set at the object. Thus, it is not processed in a subsequent run of the service agent.

Action: Check detailed message. Check network and server connection. Set dxrTBA=TRUE manually in the data view.

ERR(SVC865): Permission is invalid: It has no groups assigned.

Explanation: A permission has no groups assigned. Thus, no access right are granted via this permission.

Action: Assign groups to the permission. Remove permission in case it is not used.

ERR(SVC867): Role is not valid: Neither junior roles nor permissions are assigned.

Explanation: A role has no junior roles or permissions assigned. Thus, no access right are granted via this role.

Action: Assign junior role(s)/permission(s) to the role. Remove role in case it is not used.

ERR(SVC868): Group is invalid: It is not referenced by any permission.

Explanation: A group has no permissions assigned. Thus, it is not integrated in the role system.

Action: Create a permission for the group. Ignore message in case you only work with direct group assignments.

ERR(SVC870): Cannot remove user DN=*string1* from the data store: *string2*.

Explanation: An error occurred during physical deletion of a user.

Action: Check the detailed message, correct the error, and delete user again.

ERR(SVC871): Error loading user DN=*string1* referenced by account DN=*string2*: *string3*.

Explanation: The user referenced by the account's dxrUserLink attribute cannot be read. Thus, the account's end date may not be cleared.

Action: Check the detailed message. Check network and server connections. Correct broken user link. Edit and save user after the error has been corrected.

ERR(SVC872): Error loading user DN=*string1* referenced by account DN=*string2*: Object is of wrong type.

Explanation: The user referenced by the account's dxrUserLink attribute is of wrong type and cannot be processed. Thus, the account's end date may not be cleared.

Action: Check the detailed message. Check network and server connections. Correct wrong user link. In case the user link is correct, check if all object descriptions are correctly loaded during startup. Edit and save user after the error has been corrected.

ERR(SVC874): Cannot associate a peer account to account DN=*string1*: Load of assigned user DN=*string2* failed: *string3*.

Explanation: Assignment of a peer account to an account in a target system needing peer accounts fails: The user referenced by the dxrUserLink attribute cannot be read.

Action: Check and repair the account's user link. Run a role resolution or set the account's dxrToPeer attribute manually.

ERR(SVC875): Cannot associate a peer account to account DN=*string1*: Cannot load assigned user DN=*string2*: Object is of wrong type.

Explanation: Assignment of a peer account to an account in a target system needing peer accounts fails: The object referenced by the dxrUserLink attribute is not a dxrUser.

Action: Check and repair the account's user link. In case the link is correct, check if the object descriptions are correctly loaded during startup. Run a role resolution or set the account's dxrToPeer attribute manually.

ERR(SVC876): State cycle detected for user DN=*string1*: Last state=*string2*, new state=*string3*.

Explanation: Should never occur.

Action: Call DirX Identity support.

ERR(SVC877): User DN=*string1* has an illegal state: '*string2*'.

Explanation: The user's state has an illegal value. Thus, no new state is calculated.

Action: Correct dxrState in the data view to one of the legal values TEMPLATE, NEW, ENABLED, DISABLED, or TBDEL. Check and correct the source of illegal state entry (a user workflow?).

ERR(SVC878): The following dates of user DN=*string1* are in wrong order: *string2*.

Explanation: The following order of dates is mandatory: StartDate <= DisableStartDate <= DisableEndDate <= EndDate < DeleteDate.

Action: Correct the sequence of dates as described in the explanation.

ERR(SVC880): Error in naming rule for attribute *string1* of class *string2*: *string3*.

Explanation: Execution of a naming rule to create a new value for an attribute failed.

Action: Check detailed message. Edit value manually.

ERR(SVC881): Error while executing Java method *string1* for attribute *string2* of class *string3*: *string4*.

Explanation: Execution of a Java method to create a new value for an attribute failed.

Action: Check detailed message. Edit value manually.

ERR(SVC882): Error(s) resolving group->permission->role->user: Cannot load object DN=*string1*, object is of wrong type.

Explanation: The search uses the correct object classes. However, the Java class is not expected, indicating an error in the mapping section of the object description.

Action: Check object descriptions of user/role/permission/group. Check if the object descriptions are loaded without errors during startup of the service agent. Re-run service agent.

ERR(SVC883): Cannot create extension group for group DN=*string1* in the data store: *string2*.

Explanation: In a target system supporting extension groups a new assignment has to be stored, but creation of a new extension group failed.

Action: Check the detailed message. Check network and server connections.

ERR(SVC884): Extension group DN=*string1* is referenced by more than one master.

Explanation: An extension group is referenced by more than one master group.

Action: Check your naming of groups and extension groups to avoid those errors. Remove extension group from the master's dxrExtensionGroup DN link in the data view.

ERR(SVC885): Extension group DN=*string1* is not referenced by a master group.

Explanation: An extension group is referenced by no master group.

Action: Check if the master group has been deleted. Decide whether to delete the extension group or to create a new master for it. Add dxrExtensionGroup DN links in the data view to create the reference from the master.

ERR(SVC886): Error while adding extension group DN=*string1* to master DN=*string2*: *string3*.

Explanation: Adding an extension group to a master group failed.

Action: Check detailed message. Add dxrExtensionGroup DN links in the data view to create the reference from the master.

ERR(SVC890): Error(s) resolving group->permission->role->user: The maximum recursion depth of 20 is exceeded.

Explanation: To avoid endless loops, a maximum recursion depth of 20 is defined in the role resolution.

Action: Check your role structure, remove endless loops, reduce depth of inheritance.

ERR(SVC893): Error resolving group->permission->role->user: Cannot set TBA-Flag for DN=*string1*: *string2*.

Explanation: A user is affected from a change of a privilege. Thus, a role resolution has to be performed on that user, but marking it dxrTBA=true failed.

Action: Check network and server connections. Re-run the service agent. Edit and save user manually.

ERR(SVC894): LDAP Error(s) resolving group->permission->role->user occurred while searching users: *string1*.

Explanation: The search of users being affected by a resolution relevant change of a privilege failed. Thus, the users are not processed in the succeeding role resolution.

Action: Check network and server connections. Check the detailed message. Correct the error and re-run the service agent.

ERR(SVC895): LDAP Error(s) resolving group->permission->role->user while processing objects of type *string1*: *string2*.

Explanation: LDAP errors occurred while searching users being affected by a resolution relevant change of a privilege. The affected users cannot be determined and are not processed in the succeeding role resolution.

Action: Check network and server connections. Check the detailed message. Correct the error and re-run the service agent.

ERR(SVC896): Error while removing all rights of user DN=*string1*: *string2*.

Explanation: An error occurred when saving a user in a delete state.

Action: Check the detailed message, correct the error, and delete user again.

ERR(SVC897): Save of account DN=*string1* failed: *string2*. OnRevocations no applied for group DN=*string3*.

Explanation: A group assignment is removed and onRevocation actions are applied to the account. Due to save errors the effect of the onRevocation actions is not saved.

Action: Check the detailed message. Correct the errors, assign the privilege again, save user, edit user again, remove assignment and save user again. In case the assignment needs approval, edit the account in the data view manually.

ERR(SVC898): Reading of user DN=*string1* referenced by account DN=*string2* failed *string3* when applying onRevocations for group DN=*string4*.

Explanation: The onRevocations and subsequent possible onAssignment actions are applied without considering rules for the user base object.

Action: Check the detailed message. Correct the errors, assign the privilege again, save user, edit user again, remove assignment and save user again. In case the assignment needs approval, edit the account in the data view manually.

ERR(SVC899): Error while loading obligation DN=*string1* for group DN=*string2* from the

data store: *string3*.

Explanation: An obligation referenced by the group's dxrObligationLink cannot be read. OnAssignment/onRevocation actions are not applied.

Action: Check server and network connections. Check detailed message, repair broken link. Repeat editing the user.

ERR(SVC900): XML syntax error in attribute *string1* of group DN=*string2*: *string3*.

Explanation: An error was detected in the XML syntax of the attribute. Thus, an onAssignment/onRevocation action is not correctly processed.

Action: Correct the attribute syntax. Edit the affected user: Remove the privilege and save the user. Then, edit user again and re-assign the privilege.

ERR(SVC901): <properties> tag missing in attribute *string1* of group DN=*string2*.

Explanation: The properties tag is missing in the XML attribute. Thus, an onAssignment/onRevocation action is not correctly processed.

Action: Add the properties tag. Edit the affected user: Remove the privilege and save the user. Then, edit user again and re-assign the privilege.

ERR(SVC902): *string1* attribute missing in *string2* <property> tag in attribute *string3* of group DN=*string4*.

Explanation: An attribute is missing in a <property> tag of an XML attribute. Thus, an onAssignment/onRevocation action is not correctly processed.

Action: Add the attribute to the tag. Edit the affected user: Remove the privilege and save the user. Then, edit user again and re-assign the privilege.

ERR(SVC903): Loading of baseObject *string1* failed in *string2* <property> tag in attribute *string3* of group DN=*string4*: *string5*.

Explanation: A base object being used in a reference rule of an onAssignment(onRevocation action cannot be loaded. Thus, an onAssignment/onRevocation action is not correctly processed.

Action: Check the detailed message and correct the error. Edit the affected user: Remove the privilege and save the user. Then, edit user again and re-assign the privilege.

ERR(SVC904): BaseObject *string1* in *string2* <property> tag in attribute *string3* of group DN=*string4* does not exist.

Explanation: A base object being used in a reference rule of an onAssignment(onRevocation action does not exist. Thus, an onAssignment/onRevocation action is not correctly processed.

Action: This may occur in case an account could not be created. Check and correct the previously reported errors. Edit the affected user: Remove the privilege and save the user. Then, edit user again and re-assign the privilege.

ERR(SVC905): BaseObject *string1* in *string2* <property> tag in attribute *string3* of group DN=*string4* is not valid in this context.

Explanation: A base object being used in a reference rule of an onAssignment(onRevocation action is not accessible in this context. Thus, an onAssignment/onRevocation action is not correctly processed.

Action: Correct the reference rule using one of the baseObjects SvcUser, SvcGroup, SvcTSAccount, or SvcTS. Remove the privilege and save the user. Then, edit user again and re-assign the privilege.

ERR(SVC906): Cannot evaluate naming rule for DN=*string1*: Property *string2* is not defined in object description of class *string3*.

Explanation: A naming rule cannot be evaluated since the property is not defined in the object description. Thus, a default value is not calculated.

Action: Add missing property description to the object description. Edit value manually.

ERR(SVC907): Errors occurred while evaluating *string1* action of group DN=*string2* for user DN=*string3*.

Explanation: Errors occurred during evaluation of an onAssignment or onRevocation action for the given group and user.

Action: Correct the preceding errors. Remove the privilege and save the user. Then, edit user again and re-assign the privilege.

ERR(SVC910): Reading of group DN=*string1* referenced by member '*string2*' of account DN=*string3* failed: *string4*.

Explanation: Reading of a group being assigned by a DN-link in an account member failed. Presumably the link is broken.

Action: Check the details of the error message. Edit the member and remove or correct the broken link.

ERR(SVC911): Reading of group DN=*string1* referenced by member '*string2*' of account DN=*string3* failed: Wrong object type: *string4*.

Explanation: Reading of the group failed. The link points to an object not being a group, or it cannot be mapped in the object description.

Action: Correct the link, in case it does not point to a group. Check that the object descriptions are loaded without errors during startup.

ERR(SVC912): Cannot read extension group DN=*string1* of group DN=*string2*: *string3*.

Explanation: Reading of an extension group whose dn is stored in the master group's dxrExtensionGroups attribute failed.

Action: Check server and network connections. In case of a broken link, correct the dns being entered in the master group's dxrExtensionGroups attribute manually.

ERR(SVC913): Error(s) during save of group DN=*string2*: *string3*.

Explanation: Save of extension group failed.

Action: Check server and network connections. Check the message details. Correct the error and re-run role resolution.

ERR(SVC914): Error while loading accounts for group *string1* from the data store: *string2*.

Explanation: An LDAP error occurred during read of the accounts being assigned to a group.

Action: Check network and server connections. Check the detailed message.

ERR(SVC915): Cannot retrieve associated assignment from assignment object, DN=*string1*.

Explanation: A user's assignment cannot be determined from an assignment object holding the assignment's attributes. Unlikely to occur.

Action: Call DirX Identity support.

ERR(SVC916): No dependson attribute defined for property *string1* in object description *string2*. Can't check for uniqueness.

Explanation: A uniqueIn requires the definition of a dependsOn property containing the counter to be incremented for next value to be created. (As an example look at $racfIDCount, in the object descriptions of the RACF target system).

Action: Add a virtual (name starting with a $) counter property of type java.lang.Integer to the object description. Reference this counter in the dependsOn attribute. Use this counter in the naming rule/Java script to create a unique value.

ERR(SVC917): No value has been created for attribute=*string1* in object DN=*string2* to be checked for uniqueness: Check naming rule and attributes of object.

Explanation: An attribute being required to be unique has not been created. Maybe a referenced attribute of a naming rule is empty, or a Java script for the property did not create a value.

Action: Check the naming rule/Java scripts and/or the attributes of the object. Edit object manually and enter a unique value for the property.

ERR(SVC918): Error reading user DN=*string1* referenced as uniqueMember of group DN=*string2*: *string3*.

Explanation: The user's dn is contained in the group's unique member attribute but the user cannot be read. Thus, updating the user's contribution to the distribution list associated with the group failed.

Action: Edit the group's uniqueMember attribute in the data view. Remove the user's DN or replace it by a corrected value.

ERR(SVC919): Error reading user DN=*string1* referenced as uniqueMember of group DN=*string2*: Object is of wrong type.

Explanation: The user's dn is contained in the group's unique member attribute but the reference object is not of type dxrUser. Thus, updating the user's contribution to the

distribution list associated with the group failed.

Action: Edit the group's uniqueMember attribute in the data view. Remove the user's DN or replace it by a corrected value. In case the DN is correct, check if all object descriptions are correctly loaded during startup.

ERR(SVC920): Approval workflow '*string1*' for creation of *string2* '*string3*' could not be started.

Explanation: Cannot start the approval workflow for persona, user or functional user creation.

Action: Verify permissions and that the workflow exists and is active.

ERR(SVC921): Start of approval workflow '*string1*' for creation of *string2* '*string3*' failed: '*string4*'.

Explanation: Cannot start the approval workflow for persona, user or functional user creation.

Action: Check message details. Correct the errors and run rule again.

ERR(SVC922): Cannot read user DN='*string1*' referenced by dxrUserLink from account DN='*string2*' from the data store: '*string3*'.

Explanation: The main identity of the account cannot be read. Thus, creating a persona from the account is aborted.

Action: Check message details. Correct the broken link and run rule again.

ERR(SVC923): Account DN=*string1* has no user assigned.

Explanation: The main identity of the account cannot be read. Thus, creating a persona from the account is aborted.

Action: Link the user manually and run the rule again.

ERR(SVC924): Creating *string1* '*string2*' under parent folder '*string3*' in memory failed: '*string4*'.

Explanation: Cannot create the persona/functional user/user object.

Action: Check message details. Verify create policies.

ERR(SVC925): Searching primary accounts for user '*string1*' in target system DN=*string2* failed: '*string3*'.

Explanation: Cannot determine the primary account of the main identity.

Action: Check message details. Correct the errors and run rule again.

ERR(SVC926): Searching running workflow instance for *string1* '*string2*' failed: '*string3*'.

Explanation: Searching for already running workflows for the persona/user/functional user failed. The create workflow is not started.

Action: Check message details. Correct the errors and run rule again.

ERR(SVC927): Setting dxrUserLink '*string1*' at account '*string2*' failed: '*string3*'.

Explanation: Cannot modify the account to reference the newly created persona.

Action: Check message details. Correct the errors and run rule again.

ERR(SVC928): Account '*string1*' is of wrong type '*string2*': expecting 'siemens.dxr.service.node.SvcTSAccount'.

Explanation: The subject of the consistency rule is not an account.

Action: Correct the search filter of your consistency rule. Correct the TSAccount.xml object description.

ERR(SVC929): Cannot determine the DN of the target system for account DN=*string1*.

Explanation: The DN of the account's target system cannot be determined. Thus, the 'ts' specific JavaScript cannot be found.

Action: Check your TSAccount.xml/TS.xml object description. Check the structure of your target system.

ERR(SVC930): Cannot create a persona for account '*string1*': Could not find the JavaScript object '*string2*'. No script to execute.

Explanation: The JavaScript could be found neither under the TS specific JavaScript container, nor under the customer specific, nor under the system JavaScript container.

Action: Check for typos in the JavaScript name of your consistency rule. The name has to be the cn of the JavaScript object. Create a JavaScript object under at least one of the locations mentioned above.

ERR(SVC931): Cannot retrieve the properties mastered for objects of type *string1*: No property descriptors found for that type.

Explanation: The property descriptors for the denoted object description are not found. Thus, the modifications to be applied to the object when the user is changed cannot be determined.

Action: Check and correct your object descriptions.

ERR(SVC932): Cannot retrieve the properties mastered for objects of type *string1*: No object descriptor found for that type.

Explanation: The object descriptor for the denoted object description name is not found. Thus, the modifications to be applied to the object when the user is changed cannot be determined.

Action: Check and correct your object descriptions.

ERR(SVC933): Resolving functional user DN=*string1* for user DN=*string2* failed: *string3*.

Explanation: An error occurred while performing a state update/role resolution of a

functional user.

Action: Check message details. Correct errors in functional user.

ERR(SVC934): Filtering assigned objects failed for user DN=*string1*: Filter=*string2*, searchBase=*string3*: *string4*.

Explanation: An error occurred while requesting assigned privileges restricted by a filter.

Action: Check message details. Correct the search filter.

ERR(SVC935): Filtering assigned objects failed for user DN=*string1*: Malformed filter=*string2*, searchBase=*string3*: *string4*.

Explanation: An error occurred while requesting assigned privileges restricted by a filter.

Action: Check message details. Correct the search filter.

ERR(SVC936): Searching for assignment objects of user *string1* failed for filter=*string1*: *string2*.

Explanation: An error occurred while requesting assigned privileges restricted by a filter.

Action: Check message details.

ERR(SVC937): Error(s) occurred while creating match rules for permission DN=*string1* from raw representation. Role resolution aborted.

Explanation: An error occurred while converting the raw representation of the match rules.

Action: Check if there are erroneous object descriptions in the system. Correct object descriptions. Edit and correct match rules in DirX Identity Manager.

ERR(SVC938): Cannot set state *string1* for privilege DN=*string2*: *string3*.

Explanation: Cannot save object with dxrState changed to the value reported in the message.

Action: Check message details. Correct error and run again.

ERR(SVC939): Cannot delete privilege DN=*string1*: *string2*.

Explanation: An error occurred while deleting a privilege.

Action: Check message details. Correct error and delete again.

ERR(SVC940): Cannot move members of group *string1* to state DELETED: *string2*.

Explanation: An error occurred while moving all members of a group to be deleted to dxrGroupMemberDelete.

Action: Check message details. Correct error and delete again.

ERR(SVC950): Professional license required to use User Facet feature.

Explanation: The User Facet feature cannot be enabled since the feature is not available with the business license.

Action: Upgrade to professional license.

ERR(SVC951): Cannot read associated user DN=*string1* for User Facet DN=*string2*: *string3*.

Explanation: An error occurred while reading the associated user for the User Facet.

Action: Check detailed message, check for broken owner link.

ERR(SVC952): An Ldap error has occurred while searching the User Facets of User DN=*string1*. Inheritance of privileges from User Facets aborted.

Explanation: The inheritance of privileges from User Facets to the denoted user failed.

Action: Check Ldap server and network connections. Check the preceding messages.

ERR(SVC953): An unexpected error code=*string1* has occurred while searching the User Facets of User DN=*string2*. Inheritance of privileges from User Facets aborted.

Explanation: The inheritance of privileges from User Facets to the denoted user failed.

Action: Check the preceding messages.

INF(SVC954): Assignment to a role DN=*string1* with parameters detected for User Facet DN=*string2*. Assignment is not inherited to the user DN=*string3*.

Explanation: The inheritance of roles with role parameters is not supported.

Action: Select roles without role parameters for your User Facet.

ERR(SVC955): Assignment to privilege DN=*string1* is broken for User Facet DN=*string2*. Assignment is not inherited to the user DN=*string3*.

Explanation: The inheritance of privileges from User Facets to the denoted user failed because the privilege does no longer exist.

Action: Repair your Datastore. Remove assignment at the User Facet and add a new assignment to a valid privilege.

ERR(SVC956): An unexpected exception of type *string1* has occurred while searching the User Facets of User DN=*string2*: *string3*. Inheritance of privileges from User Facets aborted.

Explanation: The inheritance of privileges from User Facets to the denoted user failed because an exception occurred.

Action: Check the message details.

ERR(SVC957): Cannot read owner DN=*string1* for User Facet DN=*string2*: Inheritance of privileges failed: *string3*.

Explanation: The inheritance of privileges from User Facets to the denoted user failed because the user cannot be read.

Action: Check the message details. Repair owner link of User Facet. Check Ldap server and network connections.

ERR(SVC958): Error code=*string1* reported during update of owner DN=*string2* of User Facet DN=*string3*.

Explanation: The inheritance of privileges from User Facets to the denoted user failed because the user cannot be read.

Action: Check the message details. Repair owner link of User Facet. Check Ldap server and network connections.

ERR(SVC959): Assignment of User Facet inherited privilege DN=*string1* to User DN=*string2* failed: *string3*.

Explanation: The inheritance of a privilege from User Facets to the denoted user failed.

Action: Check the message details.

ERR(SVC960): Update of User Facet DN=*string1* for user DN=*string2* failed: *string3*.

Explanation: A user change cannot be realized at the User Facet.

Action: Check the message details. Edit and save the User Facet.

ERR(SVC961): Wrong type of privilege *string1* was specified for assignment removal for *string2*.

Explanation: An invalid object type was used for assignment removal.

Action: Inform DirX Identity support.

WAR(SVC962): No 'dxrUid' available for locking. Object will not be locked.

Explanation: For synchronization purposes an LDAP lock has to be set for the given object using its value of attribute 'dxrUid'. But no value is present.

Action: Check why the given object is missing the attribute 'dxrUid'the LDAP directory.

INF(SVC963): User '*string1*' (uid: '*string2*') is currently locked and can't be processed. Trying to lock after *string3* msec again. Retry count: *string4*; Max. retry count: *string5*.

Explanation: Another application has locked the entry. Operation is repeated after a short time again.

Action: The operation is repeated again.

ERR(SVC964): User entry with uid '*string1*' is locked and can't currently be processed.

Explanation: Another application has locked the user. The user entry can't be processed by this application in parallel.

Action: None. Processing must be repeated at a later time.

WAR(SVC965): No 'dxrUid' available for releasing the LDAP lock. LDAP lock can't be released.

Explanation: For synchronization purposes an LDAP lock has to be released for the given object using its value of attribute 'dxrUid'. But no value is present.

Action: Check why the given object is missing the attribute 'dxrUid'the LDAP directory.

WAR(SVC966): 'CheckAndSave' doesn't store the object '*string1*' to LDAP due to existing LDAP problems.

Explanation: There are LDAP problems (e.g. broken DN links to other objects) while storing the given object.

Action: Check the LDAP problem.

WAR(SVC967): Object '*string1*' can't be locked; error: *string2*.

Explanation: For the given reason, the user can't be locked.

Action: Check the error problem.

ERR(SVC968): *string1*.

Explanation: A circular delegation was found.

Action: Analyze Delegations. To solve problem you may select another substitute or change the delegation period.

INF(SVC969): User '*string1*' (uid: '*string2*') is currently locked and can't be processed. During retry *string3* of creating the lock the timestamp has been changed by another application. Operation aborted to avoid inconsistencies.

Explanation: Another application has locked the entry. The timestamp has been changed, so we assume an other application has changed the entry. To avoid inconsistent data operation is aborted.

Action: None. Processing must be repeated at a later time.

INF(SVC970): User '*string1*' (uid: '*string2*') is currently locked and can't be processed. Timestamp has been changed by another application. Doing a refresh from LDAP to avoid inconsistent data.

Explanation: Another application has locked the entry and the timestamp has changed. Another application may have changed the entry. To avoid inconsistencies the entry is re-read from LDAP.

Action: None.

ERR(SVC971): Entry to be locked, can't be refreshed. DN: '*string1*' error: *string2*.

Explanation: Another application has locked the entry and the refresh does not work.

Action: None.

ERR(SVC972): Entry to be locked, can't be partial refreshed. DN: '*string1*' error: *string2*.

Explanation: Another application has locked the entry and the partial refresh does not work.

Action: None.

INF(SVC973): *string1* when decrypting password of length '*string2*' of user with DN='*string3*': *string4*.

Explanation: The service layer tried to decrypt / descramble the configured password before starting a request workflow. Password seems to be provided in cleartext.

Action: Check the password.

16.7.21. Logging Messages (SVO)

This topic describes the logging messages returned by the DirX Identity order processor and service components.

INF(SVO300): RDN=*string1* and naming attribute *string2*=*string3* differ for add order id=*string4*. Building new DN=*string5*.

Explanation: The RDN differs from the naming attribute. Thus, it is updated to it's current value.

Action: None.

INF(SVO301): Password already exists for DN=*string1*.

Explanation: A password change has been rejected since the new value already exists.

Action: None.

INF(SVO302): Reading of subject DN=*string1* failed when trying to apply *string2* order id=*string3*: *string4*.

Explanation: An order to modify or delete a subject cannot be applied because the subject cannot be read from the data store.

Action: Check message details. Perform changes manually with DirX Identity Web Center or Manager.

ERR(SVO361): Remote exception during start of approval workflow for assignment of privilege DN=*string1* to subject DN=*string2*: *string3*.

Explanation: During start of approval workflow, a remote exception has been thrown, indicating an error during communication to the approval engine.

Action: Check if the Java based identity server is running. Check approval workflow state.

WAR(SVO400): Processing add modification for attribute *string1* of ModifyOrderRequest with ID=*string2* at entry DN=*string3*: Value '*string4*' already exists, is overwritten with '*string5*'.

Explanation: The subject's single-valued attribute to be processed with an ADD modification is not empty. Its value is overwritten by the value supplied with the modification.

Action: None.

WAR(SVO401): Processing a delete order for request id=*string1*, subject DN=*string2* for an assignment to privilege *string3* that does not exist in the data store.

Explanation: The privilege referenced in the order's dxrAssignTo attribute does not exist. Since the order is a delete order, it is processed anyway, to remove the broken assignment.

Action: None.

WAR(SVO402): Cannot remove value *string1* from property *string2* when applying ModifyOrderRequest ID=*string3* on subject DN=*string4*: Value does not exist.

Explanation: The specified property value cannot be removed since it does not exist.

Action: None.

ERR(SVO403): Saving of subject DN=*string1* results in SoD violations applying *string2* order id=*string3* for resource *string4*: *string5*.

Explanation: SoD violations were detected when saving the subject. New approval workflows are started to approve the violations.

Action: None.

ERR(SVO404): A modification of *string1* is contained in the order for DN=*string2*. Ignoring modification.

Explanation: A modification of the attribute dxrPassword or dxmPassword is contained in the order. The modification is ignored, since password attributes are modified by the password manager according to the changes of the userPassword attribute.

Action: Determine source of modification and remove it.

ERR(SVO405): The attribute *string1* is contained in the order for DN=*string2*. Ignoring attribute.

Explanation: The attribute dxrPassword or dxmPassword is contained in the order. The attribute is ignored, since password attributes are set by the password manager according to the changes of the userPassword attribute.

Action: Determine source of attribute and remove it.

ERR(SVO406): Assignment of *string1* to *string2* skipped because it is in a DELETED state.

Explanation: A resource is not assigned to the subject because it is in a delete state meanwhile.

Action: None.

ERR(SVO500): Initialization of order from XML representation failed: *string1*.

Explanation: Errors occurred while reading the XML representation of an order. Should never occur in case the order is created by DirX Identity standard software.

Action: Check previous error messages. Check the detailed message. Check your workflow, especially the order creating step. Correct the XML text manually. Call DirX Identity support.

ERR(SVO501): OrderRequest contains no data.

Explanation: The order request is not initialized from order data. Should not occur in case the order is created by DirX Identity standard software.

Action: Check previous error messages. Check your workflow, especially the order creating activity. Call DirX Identity support.

ERR(SVO502): Request id=*string1* is of unknown type *string2*. Request not processed.

Explanation: The request to be processed is of unknown type. Should not occur in case the order is created by DirX Identity standard software.

Action: Check previous error messages. Check your workflow, especially the order creating activity. Call DirX Identity support.

ERR(SVO503): Cannot create object from add order id=*string1*, identifier is missing.

Explanation: No identifier is provided with the order. Thus, the dn of the object to be created is not known. Should not occur in case the order is created by DirX Identity standard software.

Action: Check your workflow, especially the order creating activity. Call DirX Identity support.

ERR(SVO504): Cannot create the subject's DN from order id=*string1*.

Explanation: The dn of the object to be processed cannot be read from the order's identifier. Maybe it is missing or the identifier is of wrong type. Should not occur in case the order is created by DirX Identity standard software.

Action: Check your workflow, especially the order creating activity. Call DirX Identity support.

ERR(SVO505): Cannot create object DN=*string1*: No meta data provided with order id=*string2*.

Explanation: No meta data are provided with the order. Creator, activation date, and directory type cannot be determined. Should not occur in case the order is created by DirX Identity standard software.

Action: Check your workflow, especially the order creating activity. Call DirX Identity support.

ERR(SVO506): Cannot create object DN=*string1*: No directory type contained in the order's meta data, id=*string2*.

Explanation: The object cannot be created since its type cannot be derived due to the missing directory type. Should not occur in case the order is created by DirX Identity standard software.

Action: Check your workflow, especially the configuration of the order creating activity. Call DirX Identity support.

ERR(SVO507): Cannot create object DN=*string1* from add order id=*string2*: Directory type '*string3*' is unknown.

Explanation: The object cannot be created since its type is unknown to DirX Identity.

Action: Check your workflow, especially the configuration of the order creating activity. The directory type has to be the name of an existing DirX Identity object description.

ERR(SVO508): Cannot create object RDN=*string1* of type *string2* under parent DN=*string3* from order id=*string4*: *string5*.

Explanation: Creating a new object in memory from an AddOrderRequest failed.

Action: Check the detailed message. Check the create access policies for objects of the given type.

ERR(SVO509): *string1* (RequestID '*string2*') for object *string3*: Trying to assign multi value attribute to single value attribute '*string4*' - skip operation.

Explanation: Modify order processing failed because of incompatible property value types.

Action: Check consistency of attribute descriptions.

ERR(SVO510): *string1* (RequestID '*string2*'): No value specified for replace property '*string3*' for storage object *string4* - skip operation.

Explanation: Modify order processing failed because no new property value is specified. Unlikely to occur in case DirX Identity standard software is used to create the order.

Action: Check source of order.

ERR(SVO511): Failure generating OrderRequest: Unknown request type *string1*.

Explanation: Supported request types: ADD, MODIFY.

Action: None.

ERR(SVO512): Job.run: Creating SvcSession failed: *string1*.

Explanation: Create SvcSession failed with given Exception. Unlikely to occur.

Action: Check your installation. Call DirX Identity support.

ERR(SVO513): Job.run: Error while loading configuration: *string1*.

Explanation: Loading session configuration failed with given Exception.

Action: Correct erroneous object descriptions in the DirX Identity manager.

ERR(SVO514): Job.run: Binding to Session failed.

Explanation: Binding to session with the given bind profile failed.

Action: Correct host/port/user/password.

ERR(SVO515): Cannot create object DN=*string1* from add order id=*string2*: Naming attribute is missing in object description '*string3*'.

Explanation: The object cannot be created since its naming attribute is missing in the object description.

Action: Add naming attribute to the object description.

ERR(SVO516): Cannot process order. Reading of order failed: *string1*.

Explanation: The order read from the workflow context cannot be converted from its XML representation.

Action: Check the detailed message and correct the error.

ERR(SVO517): *string1*: Entry DN=*string2* has no object descriptor - cannot set name from the entry's naming attribute.

Explanation: No naming attribute is defined for the entry. Thus, the name cannot be set in the request.

Action: Correct the entry's object description. Add a naming attribute to it.

ERR(SVO518): Error setting attribute *string1* from AddOrderRequest with ID=*string2* at entry DN=*string3*: Multiple values are supplied for a single-valued property.

Explanation: The add order contains multiple values for an attribute being single valued in the identity store. Only the first value is processed.

Action: Correct the object description in the identity store and/or in the workflow configuration. Both configurations must use either single or multiple values for the attribute.

ERR(SVO519): Error applying modifications *string1* from ModifyOrderRequest with ID=*string2* at entry DN=*string3*: Multiple values are supplied for a single-valued property.

Explanation: The modify order contains multiple values for a modification being single valued in the identity store. Only the first value is processed.

Action: Correct the object description in the identity store and/or in the workflow configuration. Both configurations must use either single or multiple values for the attribute.

ERR(SVO520): Illegal format of start date *string1* in order id=*string2*. Cannot update assignment from subject DN=*string3* to resource DN=*string4*.

Explanation: The order to process an assignment contains a start date in a format that is not confirm to Generalized Time. The order is not processed. With DirX Identity standard clients to create the orders this error should not occur.

Action: Make sure that dxrStartDate is created in Generalized Time format in the order. Perform changes manually with DirX Identity Web Center or Manager.

ERR(SVO521): Illegal format of end date *string1* in order id=*string2*. Cannot update assignment from subject DN=*string3* to resource DN=*string4*.

Explanation: The order to process an assignment contains an end date in a format that is not confirm to Generalized Time. The order is not processed. With DirX Identity standard clients to create the orders this error should not occur.

Action: Make sure that dxrEndDate is created in Generalized Time format in the order. Perform changes manually with DirX Identity Web Center or Manager.

ERR(SVO522): Illegal format of start date *string1* in order id=*string2*. Cannot create assignment from subject DN=*string3* to resource DN=*string4*.

Explanation: The order to process an assignment contains a start date in a format that is not confirm to Generalized Time. The order is not processed. With DirX Identity standard clients to create the orders this error should not occur.

Action: Make sure that dxrStartDate is created in Generalized Time format in the order. Perform changes manually with DirX Identity Web Center or Manager.

ERR(SVO523): Illegal format of end date *string1* in order id=*string2*. Cannot create assignment from subject DN=*string3* to resource DN=*string4*.

Explanation: The order to process an assignment contains an end date in a format that is not confirm to Generalized Time. The order is not processed. With DirX Identity standard clients to create the orders this error should not occur.

Action: Make sure that dxrEndDate is created in Generalized Time format in the order. Perform changes manually with DirX Identity Web Center or Manager.

ERR(SVO524): Reading of subject DN=*string1* failed when trying to apply *string2* order id=*string3* for resource *string4*: *string5*.

Explanation: An order to add/modify/delete an assignment cannot be applied because the subject cannot be read from the data store.

Action: Check message details. Perform changes manually with DirX Identity Web Center or Manager.

ERR(SVO525): Saving of subject DN=*string1* failed when trying to apply *string2* order id=*string3* for resource *string4*; rc: *string5*; *string6*.

Explanation: An order to add/modify/delete an assignment cannot be applied because the changes cannot be saved to the data store.

Action: Check message details. Perform changes manually with DirX Identity Web Center or Manager.

ERR(SVO526): Reading of subject DN=*string1* failed when trying to apply *string2* order id=*string3*: *string4*.

Explanation: An order to modify or delete a subject cannot be applied because the subject cannot be read from the data store.

Action: Check message details. Perform changes manually with DirX Identity Web

Center or Manager.

ERR(SVO527): Saving of subject DN=*string1* failed after applying *string2* order id=*string3*: *string4*.

Explanation: An order to add or modify a subject cannot be applied because the changes cannot be saved to the data store.

Action: Check message details. Perform changes manually with DirX Identity Web Center or Manager.

ERR(SVO528): Deleting subject DN=*string1* failed after applying *string2* order id=*string3*: *string4*.

Explanation: An order to delete a subject cannot be applied because the subject cannot be deleted in the data store.

Action: Check message details. Perform changes manually with DirX Identity Web Center or Manager.

ERR(SVO529): Cannot read the reference to the assignment order's subject, request id='*string1*'.

Explanation: An order of type dxrAssignment is processed. The order is incomplete, the reference to the order's subject is missing. With orders created by DirX Identity standard software, this error should not occur.

Action: Check source of order. The subject dn has to be stored in the dxrAssignFrom attribute. In case of modify/delete orders, this is part of the identifier attributes. Correct order manually. Call DirX Identity support.

ERR(SVO530): Cannot read the reference to the assignment order's resource, request id='*string1*'.

Explanation: An order of type dxrAssignment is processed. The order is incomplete, the reference to the order's resource, i.e. the privilege to be assigned, is missing. With orders created by DirX Identity standard software, this error should not occur.

Action: Check source of order. The subject dn has to be stored in the dxrAssignTo attribute. In case of modify/delete orders, this is part of the identifier attributes. Correct order manually. Call DirX Identity support.

ERR(SVO531): The privilege *string1* referenced by order request id='*string2*' does not exist. Processing of order for privilege assignment to subject DN=*string3* aborted.

Explanation: The privilege referenced in the order's dxrAssignTo attribute does not exist. Thus, the order is not processed.

Action: Check and edit the subject's set of assigned privileges. Check the reason for removal of the privilege. Check for broken dxrAssignTo-Link in the order.

ERR(SVO532): Errors occurred during read of privilege *string1* referenced by order request id='*string2*': *string3*. Processing of order for privilege assignment to subject DN=*string4* aborted.

Explanation: The privilege referenced in the order's dxrAssignTo attribute cannot be read. Thus, the order is not processed.

Action: Check the detailed message and remove the source of the read error. Check network and server connection.

ERR(SVO533): The object *string1* referenced by order request id='*string2*' is not a privilege: *string3*. Processing of order for privilege assignment to subject DN=*string4* aborted.

Explanation: The object referenced in the order's dxrAssignTo attribute is not a role, permission or group. Thus, the order for a privilege assignment is not processed.

Action: Check the detailed message. Check if the object descriptions are correctly loaded.

ERR(SVO534): SetProperty failed for subject=*string1* for name=*string2*, value=*string3*: *string4*.

Explanation: Setting a property value failed for the given subject.

Action: Check the detailed message. Presumably the object is of wrong type and cannot be converted, e.g. a broken DN Link.

ERR(SVO536): Cannot refresh workflow instance DN=*string1*: *string2*.

Explanation: Refresh of a request workflow instance fails. Maybe the data are out-dated.

Action: Check the detailed message. Check server/network connection.

ERR(SVO537): Cannot create subject order for workflow instance *string1* from string representation *string2*: *string3*.

Explanation: Creating an order from the XML string stored in the workflow instance's dxmSubjectState attribute failed. Should never occur in case the orders are created by DirX Identity clients.

Action: Check your installation. Maybe the order is evaluated by an older software version. Check the XML structure contained in dxmSubjectState. Call DirX Identity support.

ERR(SVO538): Cannot read workflow instances in state RUNNING for user DN=*string1*: *string2*.

Explanation: An error occurred during read of the user's running approval workflows. Its orders cannot be displayed for information.

Action: Check server/network connections.

ERR(SVO539): Cannot initialize Password Manager: No cryptographic support provided.

Explanation: The userPassword has to be set but initialization of the password manager failed due to missing cryptographic support.

Action: Check installation, certificates and PSE. Call DirX Identity support.

ERR(SVO540): Cannot initialize Password Manager: *string1*.

Explanation: The userPassword has to be set but initialization of the password manager failed.

Action: Check installation, certificates and PSE. Check message details.

ERR(SVO541): Unscrambling of password failed for DN=*string1*: *string2*.

Explanation: Unscrambling the user password failed. Unlikely to occur.

Action: Call DirX Identity support.

ERR(SVO542): Encryption of password failed for DN=*string1*: *string2*.

Explanation: Encrypting a password failed. Thus, it is not set.

Action: Check message details. Check installation, certificates and PSE.

ERR(SVO543): Changing the password failed for DN=*string1*: *string2*.

Explanation: A password change failed.

Action: Check detailed message. Set password in Web Center manually.

ERR(SVO544): The privilege assignment for role *string1* references privilege *string2* that has an illegal type *string3*. Request *string4* not processed.

Explanation: The privilege assignment to the denoted role references an unknown type. Should be role or permission.

Action: Check your installation. Check object descriptions.

ERR(SVO545): The privilege assignment for permission *string1* references privilege *string2* that has an illegal type *string3*. Request *string4* not processed.

Explanation: The privilege assignment to the denoted permission references an unknown type. Should be a group.

Action: Check your installation. Check object descriptions.

ERR(SVO546): Error processing assignment for role *string1* to privilege *string2*: *string3*. Request *string4* not processed.

Explanation: Processing the role to privilege assignment ended with an error.

Action: Check the detailed error message.

ERR(SVO547): Error processing assignment for permission *string1* to privilege *string2*: *string3*. Request *string4* not processed.

Explanation: Processing the permission to privilege assignment ended with an error.

Action: Check the detailed error message.

ERR(SVO548): Cannot read active orders for subject DN=*string1*: *string2*.

Explanation: An error occurred during read of the subject's orders. The orders cannot be displayed for information.

Action: Check server/network connections.

ERR(SVO549): Cannot create LDAP connection from session after 10 tries.

Explanation: A user hook is configured in apply changes. Creating the LDAP connection for the user hook failed.

Action: Check server/network connections.

ERR(SVO550): Cannot create UserHook *string1*: *string2*.

Explanation: A user hook is configured in apply changes. Instantiation of the user hook class failed.

Action: Check the location of your user hook jarfile, should be confdb/jobs/order/lib. Check if the user hook class has a public default constructor.

ERR(SVO551): Error occurred in pre processing user hook for subject=*string1*, resource=*string2*: *string3*.

Explanation: A user hook is configured in apply changes. The pre processing call has thrown an exception.

Action: Check the coding of your user hook. Check the message details.

ERR(SVO552): Error occurred in post processing user hook for subject=*string1*, resource=*string2*: *string3*.

Explanation: A user hook is configured in apply changes. The post processing call has thrown an exception.

Action: Check the coding of your user hook. Check the messaqe details.

ERR(SVO553): Exception *string1* occurred while converting resource order to string: *string2*.

Explanation: Could not create string representation of an order after applying SoD check. The resource orders are not updated in the workflow context.

Action: Check the messaqe details.

ERR(SVO554): Order request for subject DN '*string1*' has not been processed as no LDAP lock could be established.

Explanation: A user object needs to be locked before updating it. Creating the LDAP lock entry was not successful.

Action: Check the server logfile for further error information..

ERR(SVO555): Cannot create resource order for workflow instance *string1* from string representation *string2*: *string3*.

Explanation: Creating an order from the XML string stored in the workflow instance's dxmResourceState attribute failed. Should never occur in case the orders are created by DirX Identity clients.

Action: Check your installation. Maybe the order is evaluated by an older software

version. Check the XML structure contained in dxmSubjectState. Call DirX Identity support.

ERR(SVO556): Error processing assignment for user *string1* to privilege *string2*: *string3*. Request *string4* not processed.

Explanation: Processing the user to privilege assignment ended with an error.

Action: Check the detailed error message.

16.7.22. Logging Messages (WDM)

This topic describes the logging messages returned by the Order data manager, keeps subject and resource orders consistent to its DN links.

ERR(WDM250): Creating subject order from subject string *string1* failed: *string2*. Severe data problem, processing of workflow not possible.

Explanation: Failed to create a subject order from the subject string being stored at the request workflow instance. The order may be corrupt or the unmarshaller does not work correctly due to inconsistent jarfiles.

Action: Check the subject order in the workflow's dxmSubjectString attribute. if it is corrupt, check the source of the corrupt order. Check the installation, especially in case the error occurs after an update installation or hotfix.

ERR(WDM251): Creating resource order from resource string *string1* failed: *string2*. Severe data problem, processing of workflow not possible.

Explanation: Failed to create a resource order from the resource string being stored at the request workflow instance. The order may be corrupt or the unmarshaller does not work correctly due to inconsistent jarfiles.

Action: Check the resource order in the workflow's dxmResourceString attribute. if it is corrupt, check the source of the corrupt order. Check the installation, especially in case the error occurs after an update installation or hotfix.

INF(WDM252): The subject DN has changed in the data store. Updating subject DN from *string1* to *string2*.

Explanation: The workflow's subject DN has been changed in the data store. This change is applied to the subject and resource orders stored in the workflow.

Action: None.

INF(WDM253): The subject's parent DN has changed from *string1* to *string2*. Updating subject DN to *string3*.

Explanation: The parent folder under which the new object has to be created has changed in the data store. The change is applied to the subject and resource orders stored in the workflow.

Action: None.

INF(WDM254): Naming attribute changed in subject order from *string1* to *string2*. Changing subjectDN to *string3*.

Explanation: The naming attribute was modified in the subject order. This change is applied to the subject and resource orders stored in the workflow.

Action: None.

INF(WDM255): Naming attribute changed in the data store from *string1* to *string2*. Changing subjectDN to *string3*.

Explanation: The naming attribute was modified in the data store. This change is applied to the subject and resource orders stored in the workflow.

Action: None.

INF(WDM256): The resource DN has changed in the data store. Updating resource DN from *string1* to *string2*.

Explanation: A workflow's resource DN has been changed in the data store. This change is applied to the resource order stored in the workflow.

Action: None.

ERR(WDM257): Cannot update broken resource DN=*string1* from the data store since the resource order does not contain a uid for the resource.

Explanation: A workflow's resource DN has been changed in the data store. This change cannot be applied to the resource order stored in the workflow. This will result in an error when the order is applied.

Action: Run uid migration to create a uid for all objects. Repeat assignment of the privilege to the subject.

ERR(WDM258): Cannot update broken resource DN=*string1* from the data store since resource uid=*string2* extracted from the resource order has the wrong format.

Explanation: A workflow's resource DN has been changed in the data store. This change cannot be applied to the resource order stored in the workflow. This will result in an error when the order is applied.

Action: Run uid migration to create a uid for all objects. Repeat assignment of the privilege to the subject.

ERR(WDM259): Cannot update broken resource DN=*string1* from the data store since the dn of the resource cannot be retrieved from uid=*string2*.

Explanation: A workflow's resource DN has been changed in the data store. This change cannot be applied to the resource order stored in the workflow. This will result in an error when the order is applied.

Action: Run uid migration to create a uid for all objects. Check for duplicate objects for the given uid. Repeat assignment of the privilege to the subject.

ERR(WDM260): The LDAP search for object with dxrUID=*string1* returned an error *string2*.

Explanation: An LDAP error occurred.

Action: Check the message details. Correct the cause for the error.

ERR(WDM261): The LDAP search for object with dxrUID=*string1* returns no objects.

Explanation: No object is found for the given uid. Presumably the object has been removed from the data store or the uid has been deleted.

Action: Run uid migration to create a uid for all objects.

ERR(WDM262): The LDAP search for object with dxrUID=*string1* returns multiple objects: DN=*string1*, DN=*string2*.

Explanation: Multiple objects are found for the given uid.

Action: Remove duplicate UIDs. Run uid migration to create a uid for all objects.

ERR(WDM263): The existence check for object DN=*string1* failed: *string2*.

Explanation: Possible RDN changes are not updated in the subject and resource orders.

Action: Check detailed message. Correct the error condition.

ERR(WDM264): Cannot update broken subject DN=*string1* from the data store since the dn of the subject cannot be retrieved from uid=*string2*.

Explanation: A workflow's subject DN has been changed in the data store. This change cannot be applied to the subject order stored in the workflow. This will result in an error when the order is applied.

Action: Run uid migration to create a uid for all objects. Check for duplicate objects for the given uid.

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.