infotel

Arcsys Administration

Arcsys Administration Guide

26.2.STS
July 9, 2026

Version: 26.2.STS

Publication date of the document: 2026-07-09

This document is intended for anyone who requires more information on the Arcsys product.

This document is written for all owners of a valid license.

The reader agrees to respect the confidential nature of this document.

This document is part of the Arcsys software package, a product designed and developed by Infotel. All rights are reserved for Infotel.

Contact details:

France	***Germany***	***USA***
INFOTEL SA *Immeuble Le Valmy* *6/8/18 Avenue Léon Gaumont* *75020 Paris* *France*	*Insoft Software GmbH* *Sternstr. 9-11* *D-40479 Düsseldorf* *Deutschland*	*INFOTEL Corp.* *PO Box 7730* *St Petersburg, Florida 33743-7730* *United States*
Tel: +33 (0)1 48 97 38 38	*Tel: +49 (0) 211 44 03 16-6*	*800 543 1982 – Toll-free telephone (US only)* *Tel: +1 727 343 5958*
https://techsupport.infotel.com https://infotel-software.com *<software@infotel.com>*	https://techsupport.infotel.com https://infotel-software.com *<software@insoft-infotel.com>*	https://techsupport.infotel.com https://infotel-software.com *<software@infotel.com>*

Copy prohibited without explicit authorization. © 2026 Infotel SA All rights reserved.

Table of Contents

Preface 22
1. Introduction 22
2. Reference Documents 22
2.1. Concepts 22
2.2. Installing and Updating 22
2.3. Operations 22
2.4. GUI 22
2.5. Development 22
2.6. Option guides 23
2.7. Optional modules 23
3. Symbols and Meanings 23
4. Definitions and Abbreviations 23
1. Common Administration Operations 1
1. Introduction 2
2. Administration Tasks 3
2.1. Arcsys Startup / Stop 3
2.1.1. Order in Starting Up and Stopping Arcsys 3
2.1.2. Discovering the Status of a Component 4
2.1.3. Using license files 5
2.1.4. Using systemd on Linux 6
2.2. Recommended Purge Operations 11
2.2.1. Procedure 11
2.2.2. Purging by Module 11
2.2.3. Purge Operations Common to All Arcsys Modules 12
2.2.4. Specific Purge of Arcsys Transfer Server/Service 12
2.2.5. Specific purge of Arcsys Application Agent, ArcFF format control module and Signature validator module 13
2.2.6. Specific Purge of the Arcsys Engine 14
2.2.7. Specific Purge of the Arcsys Application Agent 14
2.2.8. Specific Purge of the Arcsys Web Agent 15
2.2.9. Specific Purge of the Relational Database 16
2.2.10. Specific Purge for Attestations 17
2.3. Modify external media manager configuration 17
2.4. Setting rights on directories shared between Arcsys modules 18
3. Archiving / Retrieval / Archive restitution / Migration 20
3.1. Changing the Request Timeouts 20
3.2. Editing Transfer Server and Transfer Service Work Directories 21
3.2.1. Introduction 21
3.2.2. Directory for Archived and Retrieved Files 21
3.2.3. Online Retention Zone Directory 22
3.3. Migrating Records from One File System to Another 22
3.3.1. Choosing the Appropriate Migration Approach 22
3.3.2. Migrating Records using zone-to-zone migrations 25

3.3.3. Migrating Records using file system expiration 28
3.4. Merging PDF Copies for Synchronous Retrieval 33
3.4.1. Introduction 33
3.4.2. Implementation 33
4. Postponed Request 34
4.1. Postponed Operations 34
4.2. Real-time Processing 35
5. Security and Fixity 36
5.1. Object size control 37
5.2. Encryption method with Arcsys Transfer Server 37
5.2.1. Modifying the Encryption Method 37
5.3. Fixity Check Method 40
5.3.1. Description 40
5.3.2. Changing the Fixity Check Method 40
5.4. Using SSL 40
5.4.1. Introduction 40
5.4.2. Configuring the LDAP Directory 41
5.4.3. Configuring Arcsys to Manage SSL on LDAP or RMI 43
5.5. Protecting the Arcsys Configuration File 45
5.5.1. Overview 45
5.5.2. Parameter Description 46
6. Alerts 47
6.1. Configuration File Path 47
6.2. notify.conf File 47
7. Authentication 49
7.1. Changing a LDAP Attribute 49
7.1.1. Effects of Changes 49
7.1.2. Description of the different LDAP Parameters 49
7.2. Authentification using Arcsys Engine 56
7.2.1. Introduction 56
7.2.2. Different Users 56
7.3. JAAS Settings File 57
7.3.1. Overview 57
7.3.2. Description 57
7.4. Declaring a New User 57
8. Connecting to the Arcsys Web Agent Using HTTPS 58
8.1. Configuring Through the Administration Console 58
9. Digital signature 59
9.1. Principle 59
9.2. Implementing a Signing Agent 59
9.2.1. Introduction 59
9.2.2. Operation 60
9.2.3. Stages 60
9.3. Signature validator module plugin (ArcVerif Option) 61
9.3.1. Specifications 63

9.3.2. Implementation 63
9.3.3. Configuration requirements 63
10. Implementing the External Timestamp 65
10.1. Operation 65
10.2. Stages 65
10.2.1. Configuring the Agent 65
10.3. Recommended Configuration 66
11. Network 67
11.1. Ports 67
11.1.1. Introduction 67
11.1.2. Arcsys Engine Network Port 67
11.1.3. Arcsys API Network Port 67
11.1.4. Arcsys Transfer Server Network Port 68
11.1.5. Arcsys Transfer Service Network Port 68
11.2. Codes for Modules, IP and host names 68
11.2.1. Description 68
11.2.2. Configuring the Arcsys Modules 68
11.2.3. Changing Parameter Values 69
11.3. Howto: change a component host 70
12. Log Files 71
12.1. All Modules Except Arcsys Transfer Server and Arcsys Transfer Service 71
12.1.1. Overview 71
12.1.2. Configuration File 71
12.1.3. etc/log4j2.xml file 71
12.2. On the Arcsys Transfer Server and the Arcsys Transfer Service 72
12.2.1. Parameters 72
12.2.2. Backing Up and Rotating Log Files 72
13. Functional Trace Files 74
13.1. Modules Affected 74
14. Specific traces for media managers 75
14.1. Traces for ArcMover Tape (ArcMover Tape Option) 75
14.2. Traces for Cloud media manager (ArcMover S3 Option) 75
14.2.1. Level of trace for Cloud media 75
14.2.2. Format of traces 76
14.3. Trace files rotation 78
15. Sending Mails 80
15.1. Configuring Mail Sending 80
15.1.1. Introduction 80
15.1.2. Configuring the SMTP Server 80
15.1.3. Configuring the Sender 80
15.1.4. Mail Recipient 80
15.1.5. Authentication 80
15.2. Customizing the Mail Template Option 81
15.2.1. Introduction 81

15.2.2. File Parameters 81
15.2.3. Specific Parameters 81
15.3. Managing Errors in Mail Sending 81
15.3.1. Introduction 81
15.3.2. User Mail Address Not Found in the LDAP 82
15.3.3. Technical Error When Sending the Mail 82
16. ArcMover Tape (ArcMover Tape Option) 83
16.1. Setting up specific traces for ArcMover Tape 83
16.2. Retrieving Expired Data 85
16.2.1. Step 1 – Retrieving the Identifier of the Tape(s) and Associated Markers 85
16.2.2. Step 2 – Retrieving the Barcode(s) 86
16.2.3. Step 3 – Launching the tapecheck_dump.sh Script For Each Tape 86
16.2.4. Step 4 – Extracting the Data Obtained 87
16.3. Changing the Identifier of a Tape 89
16.3.1. Introduction 89
16.3.2. Change in the Relational Database (Oracle Database) 90
16.3.3. Changing the Tape Header (Linux) 90
16.4. Managing mailslots 92
16.4.1. Monitoring the content of the mailslots 92
16.4.2. Adding a tape in Arcsys 92
16.4.3. Removing a tape from the tape library 93
16.4.4. Managing scratch pools 94
16.4.5. Return code and error handling 94
16.5. Recycling a tape 95
16.5.1. Behavior of recycling request 95
16.5.2. Automatic recycling 96
16.5.3. Manual recycling 97
16.6. Deleting a drive 97
16.7. Deleting a tape library 97
16.8. Replacing a drive 98
16.9. Replacing a tape library 99
17. Web 101
17.1. Changing a Label 101
17.1.1. Operating Mode for an Example: Lot Screen 101
17.2. Editing the Arcsys Web Agent Online Retrieval Directory 102
17.2.1. Online Retrieval Directory 102
17.3. Configuring Lists 102
17.3.1. Description 102
17.4. Redirecting a LDAP User Group 102
17.4.1. Description 102
17.4.2. Example 103
17.5. Creating a Category 103
17.5.1. Description 103

18. Arcsys and Business Continuity Plan: Implementation 104
18.1. Step 1: Configuring the Sites 104
18.1.1. Installing the Different Arcsys Modules 104
18.1.2. Engine 104
18.1.3. Application Agents 104
18.1.4. API 105
18.1.5. Defining the Sites 105
18.1.6. Defining the Site Association 105
18.2. Step 2: Defining Duplicated Storage Zones 105
18.2.1. Configuring Zones on Several Sites 105
18.2.2. Defining a Y Copy on Two Sites 106
18.3. Step 3: Defining the Recovery Mode 106
18.3.1. Changing the Parameter 106
18.3.2. Summary of conditions 107
18.4. Backup / Recovery in Arcsys 107
18.4.1. Backing up 107
18.4.2. Rollback 108
19. File format identification 109
19.1. Format policies 109
19.1.1. Definition 109
19.1.2. Description File for a Format Policy 110
19.1.3. Default Format Policies 112
19.1.4. Importing a format policy 113
19.1.5. Attaching the format policy to an Arcsys class 113
19.2. File format server 113
19.2.1. Introduction 113
19.2.2. Main configuration 113
19.2.3. Tools configuration 114
19.2.4. Validation of PDF/A files 114
19.3. Arcsys Application Agent 115
19.3.1. Introduction 115
19.4. Handled formats 115
19.4.1. Introduction 115
19.4.2. PDF files 116
19.4.3. TIFF files 117
19.4.4. JPEG files 117
19.4.5. PNG files 117
19.4.6. Microsoft Office/LibreOffice files 118
20. Preprocessings 119
20.1. Configuration 119
20.1.1. Overview 119
20.1.2. Description 119
20.1.3. Fixity check 122
21. Configuring JDBC Access to the Relational Database 123
21.1. Description 123

21.2. Parameter description 123
21.2.1. General parameters 123
21.2.2. Configuring the pool 123
22. Managing Files in Native Format 127
22.1. Format Handler 127
22.2. Configuration 127
22.2.1. Mapping the Handler with the Mask 127
22.2.2. Configuring Arcsys Engine 128
22.2.3. Configuring Arcsys Application Agent 128
22.2.4. Configuring the Arcsys Web Agent 128
22.3. Verification of the hash value 128
22.4. Managing AFP Files (ArcAFP Option) 129
22.4.1. NOP-structured AFP File 129
22.4.2. AFP File Structured by TLE 131
22.4.3. Configuration 133
22.5. Managing ZIP Files (ArcPak Option) 133
22.5.1. Format Handler in XML Format 134
22.5.2. Mapping File in CSV Format 137
22.5.3. Associating a format handler 138
23. Configuring ArcRFT Option 139
23.1. Installing GenericSearch 139
23.2. Configuring Arcsys to use GenericSearch 139
23.2.1. Parameter description 139
23.3. Defining an indexing policy 140
23.4. Using GenericSearch in GUI 140
24. Configuring Arcsys Engine REST callbacks 141
24.1. Configuration 141
25. Attestations 142
25.1. Attestation Policies 142
25.1.1. Definition 142
25.1.2. Description File for an Attestation Policy 142
25.1.3. Default Attestation Policies 147
25.1.4. Importing an attestation policy 147
25.1.5. Attaching an attestation policy to an Arcsys class 147
26. Workflow 148
26.1. Workflow policy description file 148
26.1.1. Workflow approval and rejection rules 150
26.1.2. Workflow actions and users belonging to multiple groups 151
26.2. Workflow policy examples 152
26.2.1. Example 1: Default behavior (all groups must approve) 152
26.2.2. Example 2: Approval quorum 152
26.2.3. Example 3: Multiple validators in the same group 153
26.2.4. Example 4: Sequential validation 154
26.3. Importing a Workflow policy 155
26.4. Default workflow policy 155

26.5. Attaching a workflow policy to an Arcsys class 155
2. Configuration files and scripts description 157
1. Overview 158
2. Parameter Files 159
2.1. Separators 159
3. Arcsys RMI, TCP/IP and SOAP API 160
3.1. Overview 160
3.2. Directory Structure 160
3.3. etc/log4j2.xml file 161
3.3.1. Definition 161
3.3.2. Parameters 161
3.4. etc/arcsysJAAS.conf File 161
3.5. etc/REFERENTIAL_JDBC.properties File 161
3.6. etc/ARCSYS_RMI_API.properties File 162
3.6.1. Definition 162
3.6.2. Arcsys RMI API Parameters 162
3.6.3. General parameters for communication with the Arcsys Transfer Server 166
3.6.4. Performance-related API parameters 167
3.6.5. LDAP Directory Parameters 169
3.7. etc/ARCSYS_TCPIP_API.properties File 169
3.7.1. Definition 169
3.7.2. Arcsys TCP/IP API parameters 169
3.7.3. Communication Parameters with the Arcsys Transfer Server ... 169
3.8. etc/ARCSYS_WS_API.properties File 169
3.8.1. Definition 169
3.8.2. Arcsys SOAP API API parameters 170
3.9. launch_arcsys_api.sh (or .bat) Script 170
3.10. shutdown_arcsys_api.sh (or .bat) Script 170
3.11. status_arcsys_api.sh (or .bat) Script 170
3.12. installService_api.cmd Script 171
3.13. removeService_api.cmd Script 171
4. Arcsys REST API 172
4.1. Overview 172
4.2. Directory Structure 172
4.3. etc/log4j2.xml, etc/log4j2_status.xml and etc/log4j2_shutdown.xml files 173
4.3.1. Definition 173
4.3.2. Parameters 173
4.4. etc/arcsys-rest.properties file 173
4.4.1. Definition 173
4.4.2. Base parameters 173
4.4.3. OAuth2 multi-client parameters 178
4.4.4. General parameters for communication with the Arcsys Transfer Server 181

4.4.5. Cluster management parameters 182
4.4.6. SSO Parameters 183
4.4.7. Performance-related API parameters 190
4.4.8. Last logon Parameters 192
4.4.9. LDAP Directory Parameters 193
4.5. etc/REFERENTIAL_JDBC.properties File 194
4.6. etc/THIRD_PARTY_SEARCH_ENGINE.properties File 194
4.6.1. Definition 194
4.7. launch_apiRestWS.sh (or .cmd) Script 194
4.8. shutdown_apiRestWS.sh (or .cmd) Script 194
4.9. status_apiRestWS.sh (or .cmd) Script 194
4.10. installService_apiRestWS.cmd Script 195
4.11. removeService_apiRestWS.cmd Script 195
4.12. Generating a JWS signature key for cluster mode (ArcHP Option) ... 195
5. Arcsys Transfer Server 196
5.1. Directory Structure 196
5.2. transferServer.conf File 200
5.2.1. Component code and host name parameters 200
5.2.2. Stage directories 202
5.2.3. Server adjustment 207
5.2.4. Media Manager 210
5.2.5. Arcsys API TCP/IP Connection properties 212
5.2.6. Communication protocol 214
5.2.7. Cluster management parameters 218
5.2.8. Encryption parameters 222
5.2.9. Code set 223
5.2.10. Alerts 224
5.2.11. Post processing 224
5.3. etc/<key_managers>.conf File 225
5.3.1. Definition 225
5.3.2. Parameters for all key managers 226
5.3.3. Parameters for PKCS11 key manager 227
5.3.4. Parameters for PKCS12 key manager 229
5.4. LogParam.cfg File for Arcsys Transfer Server 229
5.5. etc/arcmover/arcMover.conf File 233
5.5.1. Definition 233
5.5.2. "settings" Section 233
5.5.3. "changers" Section 236
5.6. etc/arcmover/filesystems.conf file 237
5.6.1. Definition 237
5.6.2. "settings" Section 237
5.7. etc/arcmover/<peripheral>.conf file 238
5.7.1. Definition 238
5.7.2. "settings" Section 238
5.7.3. "scratchpools" Section 245

5.7.4. "drive_xxx" Section 246
5.8. etc/cloud/s3/<configuration_name>/cloud.conf File 249
5.8.1. Parameters 249
5.9. etc/generic/<configuration_name>/generic.conf File 257
5.9.1. Presentation 257
5.9.2. Examples 258
5.9.3. Parameters 260
5.10. Scripts 263
5.10.1. launch_transferServer.sh (or .bat) Script 263
5.10.2. status_transferServer.sh (or .bat) Script 263
5.10.3. shutdown_transferServer.sh (or .bat) Script 264
5.10.4. clean_cache_transferServer.sh (or .bat) Script 264
5.10.5. reload_transferServer.sh (or .bat) Script 264
5.10.6. rotate_logs_transferServer.sh (or .bat) Script 266
5.10.7. monitor_transferServer.sh (or .bat) Script 267
5.10.8. keymanager_delete.sh (or .bat) Script 268
5.10.9. register_transferServer.sh (or .bat) Script 268
5.10.10. Specific scripts for ArcMover Tape Option 269
5.10.11. Windows Systems 280
6. Arcsys Engine 281
6.1. Overview 281
6.2. Directory Structure 281
6.3. etc/log4j2.xml file 283
6.3.1. Definition 283
6.3.2. Parameters 283
6.4. etc/REFERENTIAL_JDBC.properties File 283
6.4.1. Definition 283
6.5. etc/ADMINCOMMAND_params.properties File 284
6.5.1. Definition 284
6.5.2. Parameters 284
6.6. ENGINE.properties File 284
6.6.1. Component code and host name parameters 284
6.6.2. Component API 285
6.6.3. Site 286
6.6.4. Paths 287
6.6.5. Cluster management parameters 288
6.6.6. Engine internal parameters 290
6.6.7. Requests 290
6.6.8. Requests processing 297
6.6.9. Hash 301
6.6.10. Archive 301
6.6.11. Migration 303
6.6.12. Postpone 306
6.6.13. Recovery 307
6.6.14. Copy 308

6.6.15. Synchronisation 309
6.6.16. Media Migration 310
6.6.17. Recycling 311
6.6.18. Disposal 312
6.6.19. Retention dates computation 313
6.6.20. Check Zones Process Parameters 314
6.6.21. Arcsys Transfer Servers 317
6.6.22. Authentication and authorization (LDAP) 321
6.6.23. Common Truststore / Keystore 333
6.6.24. Language 334
6.6.25. Event management 335
6.6.26. Mail management 336
6.6.27. Check configuration file rights 339
6.6.28. Rename mask 339
6.6.29. Database 340
6.6.30. Parameters linked with REST callbacks 341
6.6.31. Parameters loading 345
6.6.32. Cleaning of the incoherent entities in database 345
6.6.33. Generation of attestations 347
6.7. etc/ERROR_message_xx_XX.properties File 348
6.7.1. Definition 348
6.8. etc/TEMPLATE_MAIL_Alert_End_Request_fr_FR.properties File 348
6.8.1. Definition 348
6.8.2. General Parameters For Mail Templates 348
6.8.3. Specific Parameters 348
6.8.4. Specific Parameters in Mails Sent to the Administrator 349
6.9. etc/TEMPLATE_MAIL_ AlertLot_xx_XX.properties File 349
6.9.1. Definition 349
6.9.2. General Parameters For Mail Templates 349
6.9.3. Specific Parameters 349
6.9.4. Specific Parameters in Mails Sent to the Administrator 350
6.10. etc/TEMPLATE_MAIL_ CheckZones_xx_XX.properties File 350
6.10.1. Definition 350
6.10.2. General Parameters For Mail Templates 351
6.10.3. Specific Parameters 351
6.11. etc/THIRD_PARTY_SEARCH_ENGINE.properties File 351
6.11.1. Definition 351
6.12. etc/arcsysJAAS.conf File 351
6.13. launch_engine.sh (or .bat) Script 351
6.14. shutdown_engine.sh (or .bat) Script 351
6.15. start_engine.sh (or .bat) Script 351
6.16. stop_engine.sh (or .bat) Script 352
6.17. stop_engine_gracefully.sh (or .bat) Script 352
6.18. status_engine.sh (or .bat) Script 352
6.19. encrypt.sh (or .bat) Script 352

6.20. generate-key.sh (or .bat) Script 352
6.21. reload_engine.sh (or .bat) Script 353
6.22. Managing CRON Format 353
6.22.1. Description 353
6.22.2. Format 353
6.22.3. Example of CRON range 354
6.23. formatpolicy.sh (or .bat) Script 354
6.23.1. Definition 354
6.23.2. Importing a Format Policy 354
6.23.3. Exporting a Format Policy 354
6.23.4. Modifying a Format Policy 355
6.23.5. Deleting a Format Policy 355
6.24. attestationpolicy.sh (or .bat) Script 355
6.24.1. Definition 355
6.24.2. Importing an Attestation Policy 356
6.24.3. Exporting an Attestation Policy 356
6.24.4. Modifying an Attestation Policy 356
6.24.5. Deleting an Attestation Policy 357
6.25. workflowpolicy.sh (or .bat) Script 357
6.25.1. Importing a Workflow Policy 357
6.25.2. Exporting a Workflow Policy 357
6.25.3. Modifying a Workflow Policy 358
6.25.4. Deleting a Workflow Policy 358
6.26. journal.sh (or .bat) Script 358
6.26.1. Generate a journal 359
6.26.2. Delete journal events 359
6.26.3. Count remaining journal events 360
6.27. installService_engine.cmd Script 360
6.28. removeService_engine.cmd Script 360
7. Arcsys Application Agent 361
7.1. Overview 361
7.2. Directory Structure 361
7.3. etc/log4j2.xml file 362
7.3.1. Definition 362
7.3.2. Parameters 362
7.4. etc/APPAGENT.properties File 362
7.4.1. Definition 362
7.4.2. Component Code and Host Name Parameters 362
7.4.3. Component API parameters 362
7.4.4. Directories and Paths 363
7.4.5. Archiving and retrieval 363
7.4.6. Access modes to files being archived and retrieved 364
7.4.7. Communication with the engine 365
7.4.8. Securing RMI Access 366
7.4.9. Integrity parameters 367

7.4.10. Parallelization Parameters 367
7.4.11. Manifest 368
7.4.12. Object Handler 368
7.4.13. Managing UNIX/LDAP Rights 368
7.4.14. Read Buffer Size 370
7.4.15. Signing Agent (ArcSign Option) 371
7.4.16. Timestamping (ArcSign Connector) 372
7.4.17. Native Envelope 373
7.4.18. Format control 373
7.5. etc/THIRD_PARTY_SEARCH_ENGINE.properties File 375
7.5.1. Definition 375
7.6. launch_applicationAgent.sh (or .bat) Script 375
7.7. shutdown_applicationAgent.sh (or .bat) Script 375
7.8. status_applicationAgent.sh (or .bat) Script 375
7.9. installService_applicationAgent.cmd Script 376
7.10. removeService_applicationAgent.cmd Script 376
7.11. checksignature.sh (or .bat) Script 376
8. Arcsys Transfer Service 378
8.1. Directory Structure 378
8.2. transferService.conf File 379
8.2.1. Component parameters 379
8.2.2. Stage directories 380
8.2.3. General parameters 381
8.2.4. TCP clients 382
8.2.5. Code set 383
8.2.6. Communication protocol 384
8.2.7. Alerts 388
8.3. LogParam.cfg File for Arcsys Transfer Service 389
8.4. Scripts 392
8.4.1. launch_transferService.sh (or .bat) Script 392
8.4.2. reload_transferService.sh (or .bat) Script 392
8.4.3. rotate_logs_transferService.sh (or .bat) Script 393
8.4.4. shutdown_transferService.sh (or .bat) Script 393
8.4.5. status_transferService.sh (or .bat) Script 393
8.4.6. Windows Systems 394
9. Arcsys Web Agent 395
9.1. Overview 395
9.2. Directory Structure 395
9.3. etc/log4j2.xml file 396
9.3.1. Definition 396
9.3.2. Parameters 396
9.4. etc/REFERENTIAL_JDBC.properties File 397
9.5. etc/THIRD_PARTY_SEARCH_ENGINE.properties File 397
9.5.1. Definition 397
9.6. etc/WEBAGENT_configuration.properties File 397

9.6.1. Global parameters 397
9.6.2. General parameters for communication with the Arcsys Transfer Server 398
9.6.3. Available Flags 399
9.6.4. Parameters for synchronous retrieval 401
9.6.5. Parameters for creating batch requests 402
9.6.6. Parameters for logon 403
9.6.7. Parameters for external authentication 404
9.6.8. Communication Protocol Parameters with the Arcsys Transfer Server 410
9.6.9. Parameters for display customization 410
9.6.10. Parameters for Sending Mails through the End User Interface 413
9.6.11. Various Parameters in the End User Interface 413
9.6.12. Work directories 414
9.6.13. Parameters for Uploading Files 416
9.6.14. Retrieval Parameters 416
9.6.15. Parameters for Managing Native Envelopes (ArcAFP Option) 417
9.7. etc/WEBAGENT_guipreferences_example.xml File 417
9.7.1. Definition 417
9.7.2. Parameters 417
9.7.3. Example of overwriting 417
9.7.4. Customizing display 418
9.8. etc/WEBAGENT_label_xx_XX.properties File 419
9.8.1. Definition 419
9.8.2. Syntax 419
9.8.3. Languages Supplied 419
9.8.4. Parameters 419
9.9. etc/TEMPLATE_MAIL_ContactManager.properties File 419
9.9.1. Definition 419
9.9.2. Parameters 419
9.10. etc/TEMPLATE_MAIL_CreateAccount.properties File 420
9.10.1. Definition 420
9.10.2. Parameters 420
9.11. etc/WEBAGENT_AccountCreation_xx_XX .properties File 420
9.11.1. Definition 420
9.11.2. Parameters 420
9.11.3. Account Opening Request Screen 421
9.11.4. etc/WEBAGENT_JobActionCodes_xx_XX.properties File 421
9.11.5. etc/WEBAGENT_AnomalyCodes_xx_XX.properties File 421
9.11.6. etc/arcsysJAAS.conf File 421
10. Arcsys Auto-Archive Agent 422
10.1. Overview 422
10.2. List of Return Codes Sent 422

10.3. Directory Structure 422
10.4. etc/log4j2.xml file 423
10.4.1. Definition 423
10.4.2. Parameters 423
10.5. etc/AutoArchive.properties File 423
10.5.1. Parameters for accessing the RMI API 423
10.5.2. Parameters for accessing the Application Agent 425
10.5.3. Archiving Media Parameters 425
10.5.4. Default Configuration 426
10.6. launch_autoArchive.sh (or .bat) Script 427
10.7. Module Operation 428
10.7.1. Introduction 428
10.7.2. Starting the Module / CRON 428
10.7.3. Keywords Associated with the Lot 428
10.7.4. End Date 428
10.7.5. StartDate 428
10.7.6. ArchiveDate 428
10.7.7. HostName 428
10.7.8. ServiceName 428
10.7.9. Arcsys ComponentType 428
10.8. launch_autoArchiveTrace.sh (or .bat) Script 429
10.9. Module Operation 429
10.9.1. Introduction 429
10.9.2. Starting the Module / CRON 429
11. CopyRequestManager 430
11.1. Directory Structure 430
11.2. etc/log4j2.xml file 430
11.2.1. Definition 430
11.2.2. Parameters 431
11.3. etc/REFERENTIAL_JDBC.properties File 431
11.4. copy-requests-manager.sh (or .cmd) Script 431
11.4.1. `create-copy-requests` mode 431
11.4.2. `compute-statistics` mode 432
12. ArcFF format control module 433
12.1. Directory Structure 433
12.2. General Settings 434
12.2.1. etc/arcff.properties File 434
12.3. Tool Settings 438
12.3.1. etc/arcff_tool_registry.xml File 438
12.3.2. etc/arcff_tool_assignments.xml File 439
12.3.3. etc/jhove/jhove_internal.properties File 441
12.3.4. etc/jhove/jhove_module_assignments.xml File 441
12.3.5. etc/droid/DROID_SignatureFile_X.xml File 442
12.3.6. etc/droid/container-signature-X.xml File 442

12.3.7. etc/internal-signature-validation/signature-validation.properties File 442
12.4. launch_fileformat_server.sh (or .cmd) Script 445
12.5. shutdown_fileformat_server.sh (or .cmd) Script 445
12.6. status_fileformat_server.sh (or .cmd) Script 446
12.7. installService_fileFormatServer.cmd Script 446
12.8. removeService_fileFormatServer.cmd Script 446
13. ArcsysFsComparator File systems comparator 447
13.1. Directory Structure 447
13.2. etc/log4j2.xml file 448
13.2.1. Definition 448
13.2.2. Parameters 448
13.3. etc/ARCFSC.properties File 448
13.3.1. Definition 448
13.3.2. General Parameters 448
13.3.3. Parameters for Accessing the API 449
13.3.4. Specific Parameters for RMI Mode 449
13.4. analyzeFS.sh (or .bat) Script 450
13.4.1. Description 450
13.4.2. Arguments 450
13.4.3. File Output 450
13.5. analyseRef.sh (or .bat) Script 451
13.5.1. Description 451
13.5.2. Arguments 451
13.5.3. File Output 451
13.6. compare.sh (or .bat) Script 451
13.6.1. Description 451
13.6.2. Arguments 451
13.6.3. Result 452
13.7. analyze.sh (or .bat) Script 452
13.7.1. Description 452
13.7.2. Arguments 453
13.7.3. Result 453
13.8. analyzeZone.sh (or .bat) Script 454
13.8.1. Description 454
13.8.2. Arguments 454
13.8.3. Result 455
14. Arcsys standard Clients 457
14.1. Overview 457
14.2. List of Return Codes Sent 457
14.3. Directory Structure 457
14.4. etc/log4j2.xml file 458
14.4.1. Definition 458
14.4.2. Parameters 458
14.5. etc/ArchiveClient.properties File 458

14.5.1. Definition 458
14.5.2. Parameters for accessing the RMI API 458
14.5.3. Parameters for accessing the Application Agent 460
14.5.4. Archiving Parameters 461
14.6. etc/RestoreClient.properties File 464
14.6.1. Definition 464
14.6.2. Parameters for accessing the RMI API 464
14.6.3. Parameters for accessing the Application Agent 466
14.6.4. Retrieval Parameters 466
14.7. etc/SearchClient.properties File 467
14.7.1. Definition 467
14.7.2. Parameters for accessing the RMI API 467
14.7.3. Search Parameters 469
14.8. archiveClient.sh (or .bat) Script 469
14.9. restoreClient.sh (or .bat) Script 469
14.10. searchClient.sh (or .bat) Script 469
15. ArcsysBatchs batch module 470
15.1. Batchs dealing with metadata 470
15.1.1. Export Batch 470
15.1.2. Update Batch 470
15.1.3. Update Keywords Batch 471
15.1.4. Execution Characteristics 471
15.2. Batches dealing with permissions 472
15.2.1. Update Group Batch 472
15.2.2. Update User Batch 472
15.2.3. Add Rights to Every Lot of a Collection Batch 472
15.2.4. Delete Rights from Every Lot of a Collection Batch 474
15.3. General Operation 475
15.3.1. Operation Traceability 475
15.3.2. Execution 476
15.3.3. Return code 476
15.3.4. Restrictions 476
15.4. Directory Structure 476
15.5. etc/log4j2.xml file 478
15.5.1. Definition 478
15.5.2. Parameters 478
15.6. RMI.properties File 478
15.6.1. Parameters 478
15.7. EXPORT.properties File 479
15.7.1. Filtering the Export 479
15.7.2. XML Logs 479
15.7.3. Export 479
15.8. MAPPING.properties File 480
15.9. Update.properties File 481
15.9.1. Parameters 481

15.10. XSD File 482
15.10.1. Example 482
15.10.2. Lot-Level XSD 483
15.10.3. File-Level XSD 484
15.11. Input and Output Files 484
15.11.1. Export 484
15.11.2. Update 486
15.11.3. Update Keywords 486
16. Signature validator module 488
16.1. Directory Structure 488
16.2. bin/overenv.sh (or cmd) Script 489
16.3. etc/log4j2.xml file 490
16.3.1. Definition 490
16.3.2. Parameters 490
16.4. etc/signature-validator-ws.properties file 490
16.4.1. Definition 490
16.4.2. Parameters 490
16.5. signature-validator-ws.sh (or cmd) Script 500
16.6. Import missing certificates 501
Glossary 502
Registered Trademarks 510

List of Figures

9.1. Signature Validator architecture 62
15.1. Example of an Error in Mail Sending 82
18.1. Screenshot of the zone lists 106
25.1. Example of a generated archival attestation policy 146
9.1. Flags 399
9.2. Account Opening Request Screen 421
15.1. Mask of the collection COL. 482
15.2. Lot-Level XSD Schema for the COL Collection. 483
15.3. File-level XSD schema for the collection COL 484
15.4. Output Example of an XML Export. 485
15.5. Output Example of a CSV Export (in table). 485
15.6. Output Example of an XML Log File. 486

List of Tables

2.1. Variables that can be specified in systemd service definition 7
10.1. Combinations of parameters for timestamping 65

Preface

1. Introduction

This document gives the necessary information for managing the administration of the Arcsys product. It is divided into two sections:

- The first section outlines the main topics in Arcsys (archiving and retrieval, security, network, etc.), giving instructions for current administration operations.
- The second section acts as a reference, giving the structure of the different modules and hence, their parameters.

2. Reference Documents

2.1. Concepts

Arcsys Presentation Manual: **Arcsys-presentation-26.2.0.STS-en.pdf**

Arcsys Functional Description Manual: **Arcsys-functional-description-26.2.0.STS-en.pdf**

2.2. Installing and Updating

Arcsys Prerequisites Manual: **Arcsys-requirements-26.2.0.STS-en.pdf**

Arcsys Installation Manual: **Arcsys-installation-26.2.0.STS-en.pdf**

2.3. Operations

Arcsys Administration Manual: **Arcsys-administration-26.2.0.STS-en.pdf**

Arcsys Errors Manual: **Arcsys-error-26.2.0.STS-en.pdf**

2.4. GUI

Arcsys Web Interface User Manual: **Arcsys-web-26.2.0.STS-en.pdf**

Interface Guide: **Arcsys-web-end-user-26.2.0.STS-en.pdf**

2.5. Development

Arcsys API Manual: **Arcsys-api-26.2.0.STS-en.pdf**

2.6. Option guides

ArcHP Option Guide: **Arcsys-option-archp-26.2.0.STS-en.pdf**

ArcREF Option Guide: **Arcsys-option-arcref-26.2.0.STS-en.pdf**

2.7. Optional modules

BatchReporting: **BatchReporting-UserGuide-26.2.0.STS-en.pdf**

ClassAssigner: **ClassAssigner-UserGuide-26.2.0.STS-en.pdf**

MetadataReplacement: **MetadataReplacement-UserGuide-26.2.0.STS-en.pdf**

StartRetentionDateAssigner: **StartRetentionDateAssigner-UserGuide-26.2.0.STS-en.pdf**

3. Symbols and Meanings

Note

Identifies information of particular interest

Important

Identifies important information

4. Definitions and Abbreviations

See the page 502, « Glossary ».

Part 1. Common Administration Operations

1. Introduction

This section covers the main administrative tasks of the Arcsys product.

It also covers the main topics in Arcsys (archiving and retrieval, security, network, etc.), giving instructions on how to perform the most common operations associated with the product's administration.

Further details and a list of all the configurable parameters available can be found in the second half of this manual.

2. Administration Tasks

2.1. Arcsys Startup / Stop

2.1.1. Order in Starting Up and Stopping Arcsys

To start up Arcsys, you must start up the following modules in this order:

1. Arcsys RMI, TCP/IP and SOAP API (see page 170, « launch_arcsys_api.sh (or .bat) Script »);
2. Arcsys Transfer Server (see page 263, « launch_transferServer.sh (or .bat) Script »);
3. launch Arcsys Engine (see page 351, « launch_engine.sh (or .bat) Script »);
4. Arcsys Application Agent (see page 375, « launch_applicationAgent.sh (or .bat) Script »).
5. start Arcsys Engine (see page 351, « start_engine.sh (or .bat) Script »);

Then you can optionally start:

1. All Arcsys API clients (Arcsys Auto-Archive Agent, ArcsysBatchs batch module, ArcsysFsComparator File systems comparator, etc.);
2. Other Arcsys API clients.

And in any order:

- Arcsys Web Agent (start up of the application server);
- ArcFF format control module;
- Arcsys Transfer Services (see page 392, « launch_transferService.sh (or .bat) Script »).

Important

The Arcsys Transfer Service must be started after the Arcsys Transfer Server the first time it is started or when an update has been performed on its component name, host or port number.

- Arcsys REST API (see page 194, « launch_apiRestWS.sh (or .cmd) Script »).

To stop Arcsys, you need to do the same in reverse order.

Note

Application agents do not necessarily need to be restarted when you restart the rest of Arcsys (an Arcsys Application Agent only contacts the Arcsys Engine during start up).

2.1.2. Discovering the Status of a Component

This paragraph describes the naming conventions of the statuses of the main Arcsys components.

2.1.2.1. Symbols and Meanings

The status of each Arcsys component at any stage is composed of 2 numbers xy:

- x: defines the Arcsys component type:
  - 1: Arcsys Transfer Server,
  - 2: Arcsys Transfer Service,
  - 3: Arcsys Engine,
  - 4: Arcsys Application Agent,
  - 6: Arcsys RMI, TCP/IP and SOAP API,
  - 7: Arcsys REST API,
  - 8: ArcFF format control module,
  - 9: Signature validator module.
- y: defines the component's status:
  - 1 – STARTING: the component is being started up,
  - 2 – STOPPING: the component is being stopped,
  - 3 – STARTED_NOT_WORKING: the component has been started up but cannot be used,
  - 4 – STARTED_AND_WORKING: the component has been started up and is ready for use,
  - 5 – SHUTTING_DOWN: the component is being stopped,

- 6 – SHUTDOWN: the component is stopped,
- 9 – GET_STATUS_ERROR: an error occurred while recovering the component's status.

2.1.2.2. Scripts

For each of the components mentioned, a script sends the return code corresponding to its status and displays this status.

- Arcsys Transfer Server: status_transferServer.bat (or .sh) script (see Page 263)
- Arcsys Transfer Service: status_transferService.bat (or .sh) script (see Page 393)
- Arcsys Engine: status_engine.sh (or .bat) script (see Page 352)
- Arcsys Application Agent: status_applicationAgent.bat (or .sh) script (see Page 375)

2.1.2.3. Example

The following example checks the status of the Arcsys Engine:

```
C:\Infotel\ArchivingProduct\ArcsysEngine>status_engine.bat
Getting status of engine...
Status is : [STARTED_AND_WORKING]
```

With the Arcsys Engine in "STARTED_AND_WORKING" status, the value sent by the `status_engine.bat` script is: 34 (x = 3 and y = 4).

2.1.3. Using license files

2.1.3.1. Verifying a license file

You can verify a license at any time by using `verify-license-arcsys.sh` (or .cmd) script.

The script is used as follows: `verify-license-arcsys.sh` (or .cmd) `<path to license file>`

When run, the `verify-license-arcsys` scipts displays the status of the license, or an error message. The resulting message can be one of those:

- `HIDI486I THE LICENSE FOUND FOR THE PRODUCT IS A VALID ONE (.. DAYS BEFORE EXPIRATION).`
- `HIDI483E THE LICENSE FOR THE PRODUCT HAS EXPIRED SINCE .. DAYS.`

- `HIDI445W WARNING: YOU ARE STILL USING A TEMPORARY LICENSE. YOU NEED TO ENROLL THE PRODUCT WITHIN .. DAYS.`
- `HIDI446E YOUR TRY AND BUY LICENSE IS DEFINITELY EXPIRED. PLEASE ENROLL THE PRODUCT INSTALLING A FULL LICENSE.`
- `HIDI485E THERE IS NO LICENSE FILE FOUND.`: you may encounter this error when the license file you specified is not found or cannot be read
- `HIDI443E NO LICENSE HAS BEEN FOUND FOR THE PRODUCT`: you may encounter this error when the license file you specified is relative to another machine than the one on which you run the `verify-license-arcsys` script
- `HIDI484E AN ERROR HAS OCCURED DURING THE LICENSE VERIFICATION PROCESS.`: you may encounter this error when the license file specified has been altered, when the signature of the license file is incorrect, or when the version of the license file is incorrect

2.1.3.2. Updating license file

To update your license file:

1. Locate the current license file. The current license file is the one configured in the *`-Darcsys.license.file`* parameter of the Arcsys Engine launch script.
2. Replace the current license file with the new license file supplied by the Tech Support Portal.
3. Repeat this operation for each Arcsys Engine used.
4. The new license will automatically be taken into account by each Arcsys Engine when it reloads the file each day at 5AM or during its next start.

2.1.4. Using systemd on Linux

The Arcsys RMI, TCP/IP and SOAP API, Arcsys REST API, Arcsys Transfer Server, Arcsys Engine, Arcsys Application Agent, Arcsys Transfer Service, and ArcFF format control module modules can be started using systemd on Linux.

Systemd is a system and service manager for Linux operating systems. A minimal understanding of how systemd works is required in order to use the Arcsys modules as systemd services.

Note

Configuring the Arcsys modules to be run as systemd services requires root privileges.

2.1.4.1. Configuring each Arcsys module

To start each module using systemd on each server where the module is installed:

1. Update the default `.service` configuration file, if necessary. This file is located in the `service/` directory of the module installation directory. A default configuration file is available for each module that can be started with systemd.

   In order to customize your installation, you may need to:

   - Update the `Requires=` and `After=` directives, in the `[Unit]` section of the service configuration file. The `Requires=` and `After=` directives are used to specify the dependency between modules. This dependencies are limited to the modules installed on the host and correspond to the starting order explained in the chapter « Order in Starting Up and Stopping Arcsys », on page 3. If some modules listed in the `Requires=` and `After=` directives are not installed on the host, they must be removed.

   - Specify the variables `RUN_OPTION` (used for Arcsys REST API), `JAVA_OPTION` (used for Arcsys Engine and Arcsys Application Agent), `TEMP_DIRECTORY` (used for Arcsys Application Agent) and `LD_LIBRARY_PATH` (used for Arcsys Transfer Server).

     To specify those variables, update the `Environment=` directives of `[Service]` section of the services configuration file to use customized values.

     You will find below a table presenting, for each variable to specify, the script from which to report the value and the module concerned. In a service configuration file, you only have to specify the variable(s) of the module concerned.

Variable	Script from which to report the value	ArcsysModule
`RUN_OPTION`	`launch_apiRestWS.sh`	Arcsys REST API
`JAVA_OPTION`	`launch_engine.sh`	Arcsys Engine
`JAVA_OPTION`	`launch_applicationAgent.sh`	Arcsys Application Agent
`TEMP_DIRECTORY`	`launch_applicationAgent.sh`	Arcsys Application Agent
`LD_LIBRARY_PATH`	`launch_transferServer.sh`	Arcsys Transfer Server

Table 2.1. Variables that can be specified in systemd service definition

   - Update the path to the java executable. The default configuration file adds the path to the java executable used by default on the host. If you want to use another version of java, you need to update the `ExecStart=`, `ExecStop=`, `ExectStartPost=` directives of `[Service]` section with the full path of the java executable you want to use.

- Add directives for auto-restart. You can add `Restart=on-failure` directive in `[Service]` section of the service configuration file to enable auto-restart of the service. The auto-restart can be customized to specify the delay before restarting the service, using `RestartSec=` directive in `[Service]` section. `StartLimitIntervalSec=` and `StartLimitBurst=` directives, in the `[Unit]` section can also be used to prevent a failing service from being restarted endlessly. Please refer to systemd man pages for more details (`man systemd`).

- Define any other specific configuration you may want to set. Please refer to systemd man pages for more details (`man systemd`).

Important

The changes made to the default `.service` file provided by Arcsys may be overridden during updates. Keep a copy of this file if you make any changes to it.

2. Add `systemd` to *`component_status_change_alerts`* in the `notify.conf` file of the service (see page 47, « notify.conf File »).

   For example, if *`component_status_change_alerts`* parameter was empty, change for *`component_status_change_alerts`*=`systemd`. If *`component_status_change_alerts`* parameter was set to `snmp`, change for *`component_status_change_alerts`*= `smtp,systemd`.

Note

If this step is omitted, the service will not be able to start. Systemd will raise a timeout error when starting the service:

```
Job for <module>.service failed because a timeout was exceeded.
See "systemctl status <module>.service" and "journalctl -xe" for details.
```

3. Add a link to `<module>/service/<service definition file>` in the systemd services definitions directory.

   The Arcsys Application Agent and Arcsys Engine modules are provided with several sample files. These sample files are intended to fit the different ways these modules can be run and adapt the service definition to the type of installation used.

   When Arcsys Application Agent is installed on a client machine without Arcsys Engine on the same host, use the `applicationAgent_clientInstall .service` sample file. If both Arcsys Application Agent and Arcsys Engine are installed on the host, use the `applicationAgent_serverInstall.service` and `engine_withApplicationAgent.service` sample files. When using

Arcsys Engine on a host where Arcsys Application Agent is not installed, the `engine_withoutApplicationAgent.service` sample file must be used.

For example:

```
sudo systemctl link $INSTALL_DIRECTORY/ArcsysApi/service/api.service
sudo systemctl link $INSTALL_DIRECTORY/ArcsysTransferServer/service/transferServer.
service
```

Note

If you make any changes to the service definition file after creating the link to the systemd services definition directory, you will have to update the systemd service definition:

```
sudo systemctl daemon-reload
```

4. Start the service.

   When starting a module, if it depends on another module, this dependency will be automatically started by systemd, provided that both modules are installed on the same host.

5. Optionally, you can enable the service so that it is automatically started by systemd on each machine reboot.

Note

This can be done globally at the Arcsys level by using the systemd target (see below).

For example:

```
sudo systemctl enable api.service
sudo systemctl enable transferServer.service
```

2.1.4.2. Configuring Arcsys installation on each host

After configuring each module, on each host:

1. Create an `arcsys.target` file. This file contains a `[Unit]` section and an `[Install]` section. The `[Unit]` section defines the description of the target (using the `Description=` directive) and the requirements. The requirements consists of the list of Arcsys services installed on the host. These requirements are specified using the `Wants=` directive. You can use one `Wants=` directive for each requirement, or list all the requirements in a single `Wants=` directive (separate them with a space character). The `[Install]` section defines the minimal systemd target for Arcsys to run (usually `network-online.target` when Arcsys is used on multiple machines).

For a minimum server installation, the `arcsys.target` file is:

```
[Unit]
Description="Minimum server installation"

Wants=apis.service
Wants=restWSApi.service
Wants=transferServer.service
Wants=engine.service
Wants=applicationAgent.service
Wants=fileFormatServer.service

[Install]
WantedBy=network-online.target
```

For a client installation, the `arcsys.target` file is:

```
[Unit]
Description="Client installation"

Wants=transferService.service applicationAgent.service fileFormatServer.service

[Install]
WantedBy=network-online.target
```

2. Add in each Arcsys service definition file a `PartOf=arcsys.target` directive. This directive is part of the `[Unit]` section of the service definition file.

3. Add a link to `arcsys.target` in systemd services definitions directory.

   For example:

   ```
   sudo systemctl link $PWD/arcsys.target
   ```

4. Optionally, you can enable the target, if you want it to be automatically started by systemd at each reboot of the machine.

   For example:

   ```
   sudo systemctl enable arcsys.target
   ```

5. Arcsys can be started on each host, beginning with machines hosting server installations, and then client ones.

   For example:

   ```
   sudo systemctl start arcsys.target
   ```

Particular case of many instances of a module on a single host

This chapter deals with the case where many instances of a Arcsys module are installed on a host. This is for example the case when a module is duplicated for performance reasons.

For each module:

1. Rename the service configuration file. It can be suffixed by a number or a path. For example: `apis.1.service`, `apis.2.service`, `apis.accounting.service`, `apis.humanResources.service`...

2. Change the target configuration file (`arcsys.target`) to add or update the `Wants=` directive. The service name specified in the `Wants=` directive must correspond to the name of the service configuration file. There must be one directive per module installed on the machine.

3. Create the link for the service with `systemctl link` command, as explained in the previous chapters.

2.2. Recommended Purge Operations

2.2.1. Procedure

A number of Arcsys directories must be cleaned and monitored regularly via maintenance operations because they are not automatically purged by Arcsys. These purge operations can be carried by simple scripts launched via planning tools.

Furthermore, you can evoke the purge of the relational database. In fact, the entities created in the Arcsys Database are deliberately never deleted; purge operations can be scheduled.

Purging a directory corresponds to the deletion of files and subdirectories contained in it; however, the directory itself must be kept.

2.2.1.1. Triggering Purge

Some operations may require special conditions to be launched. The purge slot recommended is always specified in this guide.

2.2.2. Purging by Module

A distinction is made between purge operations shared by all Arcsys modules (logs/traces, backup files from updates) and specific purge operations.

2.2.3. Purge Operations Common to All Arcsys Modules

2.2.3.1. Archiving Logs and Traces

Arcsys modules produce log files and traces that are never deleted. It is highly recommended that you use the Arcsys Auto-Archive Agent module to automatically archive and delete after archiving.

2.2.3.2. Backup Files From Updates by ArcsysUpdater

When a module is updated with ArcsysUpdater, a `backup<Date>` directory is automatically created to store the old version of the module (JARs, property files, etc.). Previous backup directories can be deleted or at least archived if necessary.

Files can be deleted at any time.

The volume matches the volume of files backed up for each module and can be up to several tens of MB per module and per backup.

2.2.3.3. Volume of Files

Files are unpurged envelopes. You can therefore have large volumes if archiving processes have been interrupted numerous times.

2.2.4. Specific Purge of Arcsys Transfer Server/Service

2.2.4.1. Directories

The work directories configured via *`input_stage_dir`* and *`output_stage_dir`* in the configuration file of the Arcsys Transfer Server or Arcsys Transfer Service (generally `stage/cli/in` and `stage/cli/out`) can contain non-purged envelopes resulting from erroneous archiving and retrieval operations. You should therefore purge these directories regularly.

Specifically for Arcsys Transfer Server, the directories specified by *`server_stage_dir`* and *`synchro_stage_dir`* can also contain non-purged envelopes resulting from erroneous operations. You should purge these directories regularly.

2.2.4.2. Purge Slot

The files can be deleted if they are not currently the target of any operation. If the client context does not allow this, you should perform purge operations only when Arcsys is shut down.

While Arcsys is running, the *server_stage_dir* directory of the Arcsys Transfer Server can be purged with `clean_cache_transferServer.sh` script (see Page 264).

2.2.4.3. Volume of Files

Files are non-purged envelopes. You can therefore have large volumes if archiving processes have been interrupted numerous times.

2.2.5. Specific purge of Arcsys Application Agent, ArcFF format control module and Signature validator module

2.2.5.1. Directories

Arcsys Application Agent, ArcFF format control module and Signature validator module use temporary directories for part of their processing. They are automatically purged by Arcsys, but they can contain non-purged data resulting from erroneous operations. You should therefore monitor these directories to check if they need to be purged.

The directories concerned by this monitoring operation are subdirectories of *TEMP_DIRECTORY* directory. *TEMP_DIRECTORY* directory is configured in the script that launches the module (for example in `launch_applicationAgent.sh` script, for Arcsys Application Agent).

Each module have its own subdirectory. They are:

- `ArcsysTemporaryFile` for Arcsys Application Agent
- `ArcFF` for ArcFF format control module
- `SignatureValidator` for Signature validator module

2.2.5.2. Purge Slot

The files can be deleted if they are not currently the target of any operation. If the client context does not permit this, you should perform purge operations only when Arcsys is shut down.

2.2.5.3. Volume of Files

For Arcsys Application Agent, files are non-purged envelopes. For Signature validator module and ArcFF format control module, the data that may remain in temporary directories are the file objects. You can have large volumes if archiving processes have been interrupted numerous times.

2.2.6. Specific Purge of the Arcsys Engine

2.2.6.1. Directories

If mail generation is activated and mail sending has resulted in errors, the files remain in the mail directory (configured by *MAIL_DIRECTORY* in the configuration file of the Arcsys Engine).

Note

This operation is similar to ArcIP.

2.2.6.2. Purge Slot

Files can be deleted if they are not the object of mail sending at the time. If the client context does not allow this, you should perform purge operations only when Arcsys is shut down.

2.2.6.3. Volume of Files

The volume of files concerned by the purge is not large. Indeed, as mails files are small, mail sending does not generate large files.

2.2.7. Specific Purge of the Arcsys Application Agent

2.2.7.1. Directories

If manifest retrieval is activated (parameters *RESTITUTE_MANIFEST* and *MANIFEST_DEFAULT_DIRECTORY* of the configuration file of the Arcsys Application Agent), the manifest files will be generated on each retrieval or on each archive restitution in the chosen directory.

2.2.7.2. Purge Slot

The files can be deleted as soon as users no longer need them.

2.2.7.3. Volume of Files

The volume of a manifest is small, but if a number of lots are retrieved, the total volume can become quite large.

2.2.8. Specific Purge of the Arcsys Web Agent

2.2.8.1. Directories

- If drag & drop is used:
  - when the archiving collection chosen is configured without the deletion of source data, the `upload` directory of each user contains the uploaded files that are never cleaned
  - when the archiving collection chosen is configured with the deletion of source data, the `upload` directory of each user contains empty directories for each ended archiving request

  The `upload` directory is found in each `<User name>` directory created automatically in the folder specified by the *`STOCK_USERS_REP`* parameter of the Arcsys Web Agent configuration file.
- Generally, the <User name> directory contains saved search files, preferences, etc. If a user needs to be deleted from Arcsys, this directory needs also to be deleted.
- For each synchronous retrieval, the files retrieved in the directory informed in the *`ONLINE_FILE_REP`* parameter of the Arcsys Web Agent configuration file are not purged.

2.2.8.2. Purge Slot

Files resulting from drag & drop can be deleted as soon as the matching archive processes are completely finished. If the client context does not permit this, you should perform purge operations only when Arcsys is shut down.

User directories that no longer exist and that are not used for connection can be deleted at any time.

Files resulting from a synchronous retrieval can be deleted as soon as the corresponding online display (sending of the file to the user via the interface) is finished. In addition, purge can be performed outside of Arcsys use time periods, for example, every night.

2.2.8.3. Volume of Files

For the upload directory, the volume of files can be considerable if the drag & drop feature is used frequently.

For user directories that no longer exist, the volume of user files (search files, preferences) is generally low but it can be higher if search results involving a large number of entities were saved.

For the online directory, the volume of files may be high if the synchronous retrieval or view function is used frequently.

2.2.9. Specific Purge of the Relational Database

2.2.9.1. Data Concerned

- Depending on the archiving workflow chosen, there may be lots whose archiving results in an error, giving rise to the recreation of other lots with the same data;
- You may also want to delete test or example data;
- Deleting "production lots" at the end of retention is also possible, perhaps necessary within the framework of legal restrictions.

You must remember that you cannot delete an archived lot or an entity linked to a lot still archived (for example, a collection containing an archived lot).

You must, if necessary, trigger one or more disposal(s) (remember that you can do this either by modifying the storage policy, or by individually modifying the retention start date).

You can delete entities either manually via the interface, or automatically via a client using low-level interfaces (RMI or Web Services). By default, no clients are retrieved. See the Arcsys API Manual manual for more information on deletion procedures.

2.2.9.2. Purge Slot

These operations can be performed at any time. Nonetheless, if a collection containing a large number of lots is deleted, you should do it outside of the Arcsys active period because this could slow down access to certain tables (locked).

2.2.9.3. Volume of Data

Volume saved in the database depends on the number of entities deleted.

2.2.10. Specific Purge for Attestations

2.2.10.1. Data Concerned

When using an attestation policy, Arcsys produces attestation files that are never deleted. These files are located in the directory specified by the *ATTESTATION_DIRECTORY* parameter (see Page 347).

Note

The deletion of attestation files must be done with attention. Once deleted, the attestation files can only be regenerated by using the appropriate operations in the Arcsys REST API.

2.2.10.2. Purge slot

This operation can be performed at any time.

2.2.10.3. Volume of Data

The volume of an attestation is small, but the total volume can become quite large, when an attestation policy is defined on collection(s) used for numerous lots.

2.3. Modify external media manager configuration

The good practice to modify the configuration of external media manager is:

1. Modify the configuration file of the external media manager.

   Note

   This action should be scheduled in a time slot where there is no activity on the Arcsys Transfer Server, and more specifically on external media managers. Reloading the configuration of an external media manager can only be done if it is not in use.

2. Reload the configuration of the external media manager. This can be done while the Arcsys Transfer Server is running, by using the`reload_transferServer.sh (or .bat)` script with the `mediamanager` section:

   ```
   ./reload_transferServer.sh mediamanager
   ```

   Please refer to reload_transferServer.bat (or .sh) script for more information.

3. After any change in the external media manager configuration (Cloud or Generic), it is recommended that you run the `monitor_transferServer.sh (or .bat)` script to ensure the media manager is detected as expected (see monitor_transferServer.bat (or .sh) script for more information).

```
./monitor_transferServer.sh mediamanager
```

2.4. Setting rights on directories shared between Arcsys modules

Some of the directories used by Arcsys are shared across several Arcsys modules. For those directories, you must ensure that the rights set on them allow the different modules to read from and write to them.

This is especially important when two Arcsys modules are started by different users. In this case, you must ensure that the system users starting both concerned modules belong to the same group. It is also necessary that the umask set on the system allows the creation of files and/or folders which can be accessed by the two modules. It is recommended to have an umask set to `002` (`rwx rwx r-x`).

For example, Arcsys sets the permissions to `rw-` on the group on the files and `rwx` on the group on the folders that are automatically created by the Arcsys Web Agent. However, this setting is only possible if the umask set on the system allows it. If the umask is `022` (`rwx r-x r-x`), for example, the files will not have "write" permissions on the group.

The following directories may be concerned:

- The `ArcsysBuffer/users` directory (specified in the *`STOCK_USERS_REP`* parameter of the Arcsys Web Agent) is shared between Arcsys Web Agent and Arcsys Application Agent.

  Note

  The application server running the Arcsys Web Agent may also modify the system umask in its launching scripts.

- All the `log` directories are shared between the modules creating the log files and the Arcsys Auto-Archive Agent (see Log files for more information on the configuration of the log files, and Arcsys Auto-Archive Agent for more information on Arcsys Auto-Archive Agent).

- The `cli/out` and `cli/in` directories are shared between Arcsys Application Agent and Arcsys Transfer Service or Arcsys Transfer Server. Those directories are configured in the Arcsys Application Agent by the *`ARCH_REPOSITORY_PATH`* and *`REST_REPOSITORY_PATH`* parameters. In the Arcsys Transfer Service or Arcsys

Transfer Server, they are configured by the *input_stage_dir* and *output_stage_dir* parameters.

- The `/tmp` directory (specified in the *WORKSPACE_DIRECTORY* parameter of the Arcsys REST API) is shared between Arcsys REST API and Arcsys Application Agent.

3. Archiving / Retrieval / Archive restitution / Migration

The archiving and retrieval operations are basic tasks performed in Arcsys.

For each of these processes, different specifications can be changed to meet the Arcsys users' requirements and the hardware and software configuration as closely as possible.

Important

Archive restitution is a specific type of retrieval followed by a destruction of the archive. The characteristics of retrieval (processing by the Arcsys Engine, functional permissions...) are common with those of archive restitution.

3.1. Changing the Request Timeouts

An archiving, retrieval or archive restitution request may fail due to excessive processing time in changing from one status to another.

When the maximum processing time for a request by application agent or transfer server is exceeded, the request is considered as a "timeout" and an error is generated (see Arcsys Errors Manual).

This timeout can be increased in the Arcsys Engine configuration file (see page 284, « ENGINE.properties File »):

- *SEC_ARCHPROCEEDED_TIMEOUT* and *SEC_RESTVALIDATED_TIMEOUT* parameters: Timeout for processes in the Arcsys Transfer Server;

Note

Once *SEC_ARCHPROCEEDED_TIMEOUT* (for an archiving request) or *SEC_RESTVALIDATED_TIMEOUT* (for a retrieval request) is reached, the Arcsys Engine asks the Arcsys Transfer Server if the request is still being processed.

As long as the Arcsys Transfer Server answers that the request is being processed and no final status is received by the Arcsys Engine, a status request is sent every *SEC_BETWEEN_TIMEOUT_QUERIES* seconds.

If the Arcsys Transfer Server does not respond, or indicates that the request is not in its list of requests being processed, the request is set in timeout error.

Note

The status request is handled locally by the Arcsys Transfer Server. This may cause a request to end with a timeout error while it is still processed by a distant Arcsys Transfer Server.

If the request has been forwarded to a distant Arcsys Transfer Server, it may still be processed on that distant Arcsys Transfer Server, but may not be present in the list of requests being processed on the Arcsys Transfer Server that received the request.

In this case, when the timeout is reached, the Arcsys Engine sends a status request to the Arcsys Transfer Server to which it had sent the archiving or retrieving request. The Arcsys Transfer Server, having finished processing the request locally, no longer has the request in its list of requests being processed, and responds to the Arcsys Engine accordingly. The Arcsys Engine sets this request in timeout.

Consequently, when the processing of a request can be notably longer on one site than on another (for example when the archiving is done on media with slower access), it is recommended to set the timeout parameters to the value of the time required for the processing on the slowest site.

- *SEC_ARCHVALIDATED_TIMEOUT* and *SEC_RESTPROCEEDED_TIMEOUT* parameters: Timeout for processes in the Arcsys Application Agent.

3.2. Editing Transfer Server and Transfer Service Work Directories

3.2.1. Introduction

During Arcsys installation, users will be asked to define the name of the different directories to be used during archiving and retrieval. The location of these directories can be changed after installation.

3.2.2. Directory for Archived and Retrieved Files

To change the location of the directory where the Arcsys Transfer Server copies the files being archived or retrieved, see page 200 (« transferServer.conf File »). The parameters to be changed are:

- *server_input_stage_dir* and *server_output_stage_dir*: Where the Arcsys Transfer Server and the Arcsys Transfer Service are on two different machines;

- `input_stage_dir` and `output_stage_dir`: Where the Arcsys Transfer Server and the Arcsys Transfer Service are on the same machine.

For more information on changing the directory used by the Arcsys Transfer Service to copy the files being archived or retrieved, see page 379 (« transferService.conf File »). The parameters to be changed are: `input_stage_dir` and `output_stage_dir`.

Note

If the `input_stage_dir` and `output_stage_dir` parameters are changed, the corresponding parameters in the application agent must be also be updated (see page 362, « etc/APPAGENT.properties File »): `ARCH_REPOSITORY_PATH` and `REST_REPOSITORY_PATH`.

3.2.3. Online Retention Zone Directory

The online retention zone is where the files marked for retention are stored on the Arcsys Transfer Server (see Arcsys Presentation Manual).

The online retention zone directory is specified at parameter file level in the Arcsys Transfer Server (see page 200, « transferServer.conf File »). The parameter to be changed is: `server_online_dir`.

3.3. Migrating Records from One File System to Another

When a file system can no longer be used for storing data, Arcsys provides two approaches to move the records stored on it.

The first approach consists of adding a new zone and starting zone-to-zone migrations by changing the storage policy of the records.

The second approach consists of requesting the expiration of the file system. In this case, Arcsys moves the records to one or more other file systems of the same zone.

Each approach has benefits and constraints, which are listed in the following sections. The appropriate approach depends on your needs and operating constraints. Each approach is described in detail in a dedicated subsection.

3.3.1. Choosing the Appropriate Migration Approach

Use the following comparison of benefits and constraints to decide which approach to use when a file system can no longer be used for storing data.

3.3.1.1. Zone-to-Zone Migrations

Using zone-to-zone migrations is the preferred approach when the change of storage medium must be represented by a zone change and remain visible in the list of operations performed on the record (for example to meet traceability requirements) or when you want to move records created by ArcRef Injector. This approach is also appropriate when you want to use the data movement operation to change storage characteristics, such as the media type or the compression level. This approach requires the source zone to be the last zone in each storage profile that uses it, because Arcsys allows new zones to be added only at the end of a storage profile. If the source zone is followed by another zone in one or more storage profiles, use file system expiration instead.

The benefits of zone-to-zone migrations are:

- They allow changes to storage characteristics during the migration process. For example, records can be migrated to an ArcMover Tape zone, to an S3 cloud storage, or to a zone with a different compression level.
- The requests are automatically recreated when they cannot complete.
- The storage medium change is represented through two different zones.

The constraints are:

- Zone-to-zone migrations require changes to the storage profiles of the collections that use the source zone. This requirement has the following consequences:
  - To move records from a file system by using zone-to-zone migrations, you must identify the collections of the records stored on this file system, and then change the storage profiles of these collections. This can be restrictive when the same zone is used by several storage profiles.
  - The zone that contains the file system that can no longer be used must be the last zone in each storage profile that uses it. Arcsys allows new zones to be added only at the end of a storage profile. Therefore, zone-to-zone migrations are not suitable when the source zone is followed by another zone in one or more storage profiles.
  - All records in the source zone are migrated, regardless of the file system on which they are stored. If the source zone contains file systems that must no longer be used and file systems that can still be used, zone-to-zone migrations do not provide a way to select which file systems to keep.
- Performance can require special attention because each record is migrated individually. Processing time can increase when the migration also changes storage characteristics, such as the media type, compression settings, or encryption settings.

3.3.1.2. File System Expiration

Important

This chapter refers to a feature preview (see Feature preview for more details).

File system expiration is the appropriate approach when performance is an important criterion, when the zone that contains file system that must no longer be used is followed by another zone in one or more storage profiles, or when the storage characteristics of the records do not need to change. This approach applies only to records that do not originate from ArcRef Injector. If the file system to expire contains records that originate from ArcRef Injector, use zone-to-zone migrations instead.

The benefits of file system expiration are:

- It can provide better performance depending on the configuration and the physical capabilities of the platform. It can move records up to four times faster than zone-to-zone migrations in some tested configurations.
- It allows to expire file systems associated with zones that are followed by other zones in storage profiles.

The constraints are:

- Records created by ArcRef Injector are not considered by file system expiration.
- It keeps the existing characteristics of the records: compression, encryption or storage medium type.
- Error management requires special attention and manual intervention:
  - File system expiration is represented by a single request, whose final status changes to ERROR as soon as one envelope cannot be moved. When data to move cannot be read, a manual intervention is required (see page 30, « Handling File System Expiration Errors »).
  - If the file system expiration request does not complete successfully, it must be created again manually.
- Because file systems cannot be modified while an action is in progress on their zone, a file system expiration request prevents any change to all the file systems associated with the zone during its entire processing.

3.3.2. Migrating Records using zone-to-zone migrations

3.3.2.1. Restrictions

The example shown is for a file system associated with a zone used last in a single storage policy.

Furthermore, the file system to be replaced is the only one associated to that zone.

3.3.2.2. Adding a New File System

The first thing to do is to create a new zone, then a new file system associated with this zone by following the procedure described in the Arcsys Web Interface User Manual (sections Storage/Managing Zones/Creating and Media Manager Administration/File Systems Management/Creating a File System).

This zone will receive all the records currently found in the zone of the file system to be deleted.

Next, a new zone needs to be added to the storage policy affected. To ensure the record life cycle on that storage policy is not affected, the same configuration should be kept for this zone as that used for the zone to be deleted in the storage policy: same retention schedule and compression level. The process for adding a zone to a storage policy is described in the Arcsys Web Interface User Manual (section Storage/Managing and Selecting Profiles/Managing Associated Pools).

Once the new zone is ready, the old zone must be disabled. To do this, change its retention schedule to 0 days in the storage policy.

3.3.2.3. Migration

Once the storage policy has been updated, the migration needs to be configured in the Arcsys environment. These settings are done in the Arcsys Engine responsible for migration.

Three parameters need to be configured:

- *MIGRATION_LAUNCH_INTERVAL*
- *MAX_MIGRATION_REQUESTS_PER_DAY*
- *REQUESTS_CREATE_LOOPSLEEPTIME*
- *NB_REQUESTS_CREATED*

According to the size of the zone in the file system to be replaced, the migration process will use more or less resources. Adjust the migration parameters so that other Arcsys Engine operations (archiving, retrieval) are not affected.

The first parameter specifies the creation time of the migration requests and the second gives the maximum number of migration requests generated by day.

Every *REQUESTS_CREATE_LOOPSLEEPTIME* milliseconds, the Arcsys Engine creates at most *NB_REQUESTS_CREATED* requests, within the limit of *MAX_MIGRATION_REQUESTS_PER_DAY* requests per day.

First of all, define the time range in which the Arcsys Engine must create the migration requests. The start and end times can be set, but the end time simply means that no request will be created after this end time.

The number of requests created on each iteration during the migration process needs to be refined, as the time required to perform a migration depends on a number of factors (network architecture, file size, hardware specifications, etc.).

You can also set a parameter to stop the requests in progress as soon as it goes beyond the authorized range with the *REQUESTS_PROCESSING_STOP* parameter.

Note

For some parameters, the Arcsys Engine should be restarted for the new values to take effect.

For more information on Arcsys Engine configuration, go to the section on this topic (see page 284, « ENGINE.properties File »).

3.3.2.4. Monitoring Migrations Statuses

The migration requests can be viewed to check their end date and their final status.

To do this, go to the request display screen in the Arcsys Web Admin interface and filter the requests by "Migration" type.

3.3.2.5. Monitoring Migrations Progress

To monitor the general progress of the zone-to-zone migrations process, check the number of records still present on the file system. You can also monitor progress by checking the used size on the file system.

When the remaining number of records and the number of records moved each day are known, you can estimate the time required to complete the zone-to-zone migrations process. You can use the same approach with the used size on the file system by comparing the remaining used size with the size moved each day.

The number of records and the used size are available in several ways:

- From the Arcsys REST API:
  - Use the following endpoint to retrieve the list of mover medias, which correspond to the envelopes present on the file system:

    ```
    GET /filesystems/{fsId}/moverMedias
    ```

    Use the following endpoint to retrieve the list of records present on the file system:

    ```
    GET /filesystems/{fsId}/objects
    ```

    Use the `total_count` field to get the number of envelopes or records without listing all results.

    When all data has been moved, both endpoints return a `total_count` value of `0`.
  - Use the following endpoint to retrieve the file system capacity information

    ```
    GET /filesystems/{fsId}
    ```

    Calculate the used size from the `capacity` and `leftSpace` values (`capacity - leftSpace`).
- From the administration interface of the Arcsys Web Agent:
  - Open the file system detail screen to view the chart that shows the used space.
  - Open the tab that lists the records present on the file system.

  See the Arcsys Web Interface User Manual (Media Manager Administration/File Systems Management section)

The available space on the file system is also available with `./monitor_transferServer.sh` script (see page 267, « monitor_transferServer.sh (or .bat) Script »).

The available space increases as records are moved to another zone.

3.3.2.6. End of Migration

Once there are no more lots on the file system, some operations need to be performed to ensure that the file system can no longer be used:

1. Delete the storage pool;
2. Lock the file system.

These operations are described in the Arcsys Web Interface User Manual (Storage/Managing and Selecting Profiles/Managing Associated Pools and Media Manager Administration/File Systems Management/Locking-Unlocking a File System sections).

This way, the file system will not be used to store records although it will still be available in Arcsys for tracking.

3.3.3. Migrating Records using file system expiration

Important

This chapter refers to a feature preview (see Feature preview for more details).

3.3.3.1. Creating a File System Expiration Request

A file system expiration request can be created from the Arcsys REST API or from the file system list in the administration interface of the Arcsys Web Agent (see the Arcsys Web Interface User Manual, Media Manager Administration/File Systems Management section).

To create the request from the Arcsys REST API, use the following endpoint:

```
POST /filesystems/{fsId}/_expire
```

The request can be created only if no record currently stored on the file system is involved in a request that is already in progress. Consultation requests, export requests, and any request that can run on a locked file system remain possible if they are started after the file system expiration request is created.

When the file system expiration request is created, its status is INITIALIZED.

3.3.3.2. File System Expiration Request Process

After a file system expiration request is created, it is handled by an Arcsys Engine that belongs to the same site as the zone associated with the file system to expire. The Arcsys Engine can handle the request only if its *MEDIA_MIGRATION_ENGINE* parameter is set to `true`.

The Arcsys Engine sends the request to an Arcsys Transfer Server for processing. The request status then changes to PROCEEDED.

When the Arcsys Transfer Server receives the request, it first locks the file system to expire, so that no further archiving is done on it.

The Arcsys Transfer Server processes file system expiration through a sequence of loops. At each loop, it selects a set of records to move and identifies a target file

system with enough available space within the same zone. It then moves the selected data from the source file system to the selected target file system. Over successive loops, records can be distributed across multiple destination file systems within the same zone.

The request ends when the maximum number of loops is reached or, if *expiration_fs_select_max_loop* is set to `0`, when all records have been considered.

If the request ends with the ERROR status, manual checks are required. For more information, see page 30, « Handling File System Expiration Errors ».

3.3.3.3. Configuring File System Expiration

The request is processed by the Arcsys Engine according to the media migration settings:

- *MEDIA_MIGRATION_ENGINE*: specifies whether the Arcsys Engine can handle file system expiration requests
- *MEDIA_MIGRATION_LAUNCH_INTERVAL*: specifies the creation time of the media migration requests
- *MAX_MEDIA_MIGRATION_REQUESTS_PER_DAY*: gives the maximum number of media migration requests generated by day

See page 310, « Media Migration » for more detail on these parameters.

The Arcsys Transfer Server processes the file system expiration according to the following settings, defined in the `arcMover.conf` file:

- *expiration_fs_nb_threads*: specifies the number of threads used
- *expiration_fs_select_max_movers*: specifies a maximum number of movers (envelopes) selected in a single loop
- *expiration_fs_select_max_size*: specifies a maximum cumulative size of movers selected in a single loop
- *expiration_fs_select_max_loop*: specifies the maximum number of loops

See page 233, « etc/arcmover/arcMover.conf File » for more detail on these parameters.

3.3.3.4. Storage Capacity Requirements

File system expiration is possible only if other file systems in the same zone are eligible to receive the data stored on the file system to expire.

A new file system can only be added to the zone when no request is in progress on this zone. Therefore, before creating a file system expiration request, use one of the following approaches:

- Make sure that the zone has enough available space to receive the data from the file system to expire and, if needed, the new archive requests that continue to arrive during processing.
- Set *expiration_fs_select_max_loop* to a value that stops processing regularly and allows file systems to be added or modified between expiration requests .

3.3.3.5. Monitoring file system expiration Status

The file system expiration request can be viewed to check its end date and its final status.

To do this, go to the request display screen in the Arcsys Web Admin interface and filter the requests by "Media Migration" type.

3.3.3.6. Handling File System Expiration Errors

When a file system expiration request ends with the ERROR status, manual checks are required to identify the envelopes that could not be moved and to recover the related records:

1. Locate the log file of the Arcsys Transfer Server. The log file is defined by the *File* parameter in the `LogParam.cfg` configuration file. The default file is usually `arcsys-transfer-server.log` (see « LogParam.cfg File for Arcsys Transfer Server », page 229).
2. Search for the `AMM011E` error to identify the envelopes that require attention:

   ```
   grep AMM011E arcsys-transfer-server.log
   ```

3. For each record related to an envelope that could not be moved, create a check request. You can create the check request from ArcWeb or from the Arcsys REST API.

   To create the request from the Arcsys REST API, use the following endpoint:

   ```
   POST /lots/{lotId}/_check
   ```

   When you create the check request from the Arcsys REST API, specify the zone associated with the file system to expire as zone identifier where the record needs to be checked.

   Depending on the final status of the check request:

- If the check request ends with the ENDED status, the file system expiration issue probably came from the destination media or from the database update.

  Check that the required modules are running correctly, especially the Arcsys TCP/IP API.

  Also check that the zone has enough available space to receive the data from the file system to expire. The zone must contain enough online file systems, and their total available space must be at least equal to the used size on the file system to expire. If the zone does not have enough available space, add file systems to the zone before you create a new file system expiration request.

- If the check request ends with the ERROR status, the record data on the media cannot be read correctly. The verification request automatically creates a recovery request from a zone where another copy of the record is available.

  Because the file system to expire remains locked, the recovery request uses another file system in the same zone as the destination. After the recovery request is complete, the record is no longer present on the file system to expire.

4. Create new file system expiration requests as long as data remains on the file system. In some cases, a new expiration request is not required: when the verification requests create recovery requests that move the last remaining records from the file system.

Note

- If an error occurs during file system expiration processing, the file system remains locked. To make it available for write operations again, unlock it manually.

- If a file system expiration request is in progress when the Arcsys Engine starts, the request is canceled.

- If several envelopes cannot be moved during the same file system expiration request, only the last error is displayed in the request monitoring screen.

- When a new file system expiration request is created for the same file system, processing resumes from the records that are still present on the file system to expire. Records that were already moved by the previous request are not processed again. The Arcsys Transfer Server can also try again to move records that could not be moved during the previous request.

Important

To interrupt a file system expiration request, stop the Arcsys Transfer Server with the shutdown script. When the Arcsys Transfer Server stops, the request ends with the ERROR status and must be created again manually.

3.3.3.7. Monitoring File System Expiration Progress

To monitor the general progress of the file system expiration process, check the number of records still present on the file system. You can also monitor progress by checking the used size on the file system.

When the remaining number of records and the number of records moved each day are known, you can estimate the time required to complete the file system expiration process. You can use the same approach with the used size on the file system by comparing the remaining used size with the size moved each day.

The number of records and the used size are available in several ways:

- From the Arcsys REST API:
  - Use the following endpoint to retrieve the list of mover medias, which correspond to the envelopes present on the file system:

    ```
    GET /filesystems/{fsId}/moverMedias
    ```

    Use the following endpoint to retrieve the list of records present on the file system:

    ```
    GET /filesystems/{fsId}/objects
    ```

    Use the `total_count` field to get the number of envelopes or records without listing all results.

    When all data has been moved, both endpoints return a `total_count` value of `0`.
  - Use the following endpoint to retrieve the file system capacity information

    ```
    GET /filesystems/{fsId}
    ```

    Calculate the used size from the `capacity` and `leftSpace` values (`capacity - leftSpace`).
- From the administration interface of the Arcsys Web Agent:
  - Open the file system detail screen to view the chart that shows the used space.
  - Open the tab that lists the records present on the file system.

See the Arcsys Web Interface User Manual (Media Manager Administration/File Systems Management section)

The available space on the file system is also available with `./monitor_transferServer.sh` script (see page 267, « monitor_transferServer.sh (or .bat) Script »).

The available space increases as records are moved to other file systems.

3.3.3.8. End of file system expiration

Once there are no more records on it, the file system can be removed from Arcsys. This operation is described in the Arcsys Web Interface User Manual (Media Manager Administration/File Systems Management/File System List).

After the file system is removed, it no longer appears in the list of file systems associated with the zone. This prevents it from being unlocked by mistake and used again for record archiving. The file system remains present in the backup tables.

3.4. Merging PDF Copies for Synchronous Retrieval

3.4.1. Introduction

During synchronous retrieval managed by the Arcsys APIs, PDF copies can be used which will be merged for the PDF document to be retrieved. This post-processing enables logos, watermarks, etc. that do not vary to be appended and thus reducing the size of the archived PDFs.

3.4.2. Implementation

To implement synchronous retrieval using copies, three steps need to be performed:

1. Place the copies (PDF) to be used in the `etc/layers` directory of the Arcsys Transfer Server;
2. Set the *`online_restore_post_processing`* parameter of the Arcsys Transfer Server as `MergePDFLayers` (see Page 200);
3. In a program using the RMI (Java) or SOAP web services APIs, call the `customOnlineRestore` method with the appropriate parameters (see the Arcsys API Manual).

4. Postponed Request

Some operations impact thousands of entities. This means that execution time is relatively long and could prevent the user from using the Arcsys Web Interface during processing, or cancel the operation due to a server timeout.

To solve this issue, these operations can run in the web interface as a postponed operation: they are transformed into Arcsys requests that will be run according to the schedule set by the Arcsys Engine.

4.1. Postponed Operations

The operations affected by this mechanism are the following ones :

- On collection level: deletion of a collection;
- On pool level: deletion of a pool;
- On zone level: deletion of a zone;
- On keyword level: addition or modification of the link between a keyword and a mask; deletion of the link between a controlled keyword and its values; modification of a keyword;
- On archive level: deletion of an archive.

The parameter *POSTPONE_ALL_LONG_OPERATIONS* in `WEBAGENT_CONFIGURATION.properties` defines globally the behaviour of all these operations in the web interface.

By default its value is `false`, which means the operations will be performed systematically synchronously.

If the parameter is set to `true`, all these operations will be done through postponed requests. The modification in the database is effective only when the status of the request is ENDED.

In this case, the "Requests" interface in the Administration interface displays the status of each postponed operation and enables them to be monitored.

The administrator should enable this parameter when one of the previously listed operations either fails (the web interface displays, after a long time depending on the database and web server timeout configuration, "The action could not be done") or if the operation succeeds but blocks the interface during a delay which is too long for the users of the interface.

For more information on the web configuration, go to the section on this topic (see Page 397).

At Arcsys Engine level, the parameters involved for processing these requests are *POSTPONE_ENGINE*, *SEC_POSTPONE_TIMEOUT* and *POSTPONE_LAUNCH_INTERVAL*.

For more information on Arcsys Engine configuration, go to the section on this topic (see page 284, « ENGINE.properties File »).

Important

Operations are never postponed when performed through the various APIs (REST, RMI, SOAP).

4.2. Real-time Processing

To prevent any concurrency conflicts, the archiving interface cannot be accessed while a postponed operation is in progress.

5. Security and Fixity

Archived data may be of a sensitive nature and important for a company, so ensuring the security and fixity of all objects archived is an essential part of Arcsys (see Arcsys Presentation Manual).

These mechanisms can be configured thanks to a number of parameters.

Optimum security is ensured in Arcsys:

- At Arcsys level through:
  - Implementing a systematic fixity check at Arcsys Application Agent; see the section Fixity Check Method,
  - Implementing SSL on RMI communications between the various Arcsys modules; see the section Using SSL,
  - Encrypting communications between the Arcsys Transfer Server and other modules; see the section Encryption Method with the Arcsys Transfer Server,
  - Protecting configuration files using appropriate permissions and using the *CURRENT_FILE_RIGHT_MUST_BE* parameter to prevent modules from being starting up without correct rights; see the paragraph Protecting Arcsys Configuration Files,
  - Setting the *USE_POSIX_USER* parameter at Arcsys Application Agent level so that it can only access the LDAP user data of the requests in Unix systems,
  - Implementing rights management at Arcsys entity level; see the Arcsys Functional Description Manual for the section on managing permissions,
  - Specifying the IPs authorized to communicate with the server and transfer services at settings file level in the *ip_whitelist* parameter; see etc/transferServer.conf file and etc/transferService.conf file;
- At external add-on level, which is part of the prerequisites (LDAP directory, application server, etc.):
  - Implementing HTTPS at application agent level; see the section Connecting to the Web Agent using HTTPS,
  - Implementing LDAPS at LDAP directory level; see paragraph Using SSL/STARTTLS,
  - Closing ports not used by Arcsys, see in the Arcsys Installation Manual for the list of ports used (Ports to open section).

5.1. Object size control

If Arcsys is provided with objects or archives that are larger than what can be handled by the system storage, those objects or archives might crash the system.

Additionally, Arcsys is vulnerable to compression logic bomb (small files which content, once unpacked, is larger than the system can handle).

Theses issues are deemed to be outside of the scope of Arcsys. They must be handled outside of the product, for your specific context.

As part of the archiving of native zip files, you must ensure upstream of the validity of the submission package with regard to the size.

5.2. Encryption method with Arcsys Transfer Server

Communication between the Arcsys Transfer Server and other modules (Arcsys Engine, Arcsys API (RMI, SOAP or REST), Arcsys Web Agent, Arcsys Transfer Services) can be encrypted. This encryption ensures confidentiality of the data during transfer, including where the link between the two machines is not secure.

Arcsys currently implements the SSL method and the Blowfish method.

5.2.1. Modifying the Encryption Method

Encryption selection is done using the *`TRANSFER_SERVER_SECURITY_PROTOCOL`* parameter which must be specified for the Arcsys Engine, Arcsys API (RMI, SOAP or REST) and Arcsys Web Agent. This parameter must match the *`security_protocol`* parameter of the Arcsys Transfer Server.

5.2.1.1. Encrypting communications with Arcsys Transfer Server

Parameter name	**TRANSFER_SERVER_SECURITY_PROTOCOL**
Parameter type	Possible values: `NONE`, `INTERNAL`, `SSL`. The value must be the same between all clients and servers exchanging data. It is not mandatorily the same on a global Arcsys installation.
Parameter change assessment	After module reboot.
Description	Indicates if communication to and from the transfer server is encrypted or not, and the type of encryption (symmetrical for `INTERNAL`: Blowfish algorithm is used; or asymmetrical for `SSL`). Disabling encryption increases performance but may be forbidden depending on the required security level.
Shipped value	`NONE`

Parameter name	**TRANSFER_SERVER_KEYSTORE_PATH**
Parameter taken into account only if:	*TRANSFER_SERVER_SECURITY_PROTOCOL*=SSL
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRANSFER_SERVER_KEYSTORE_PASSWORD**
Parameter taken into account only if:	*TRANSFER_SERVER_SECURITY_PROTOCOL*=SSL
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRANSFER_SERVER_KEYSTORE_TYPE**
Parameter taken into account only if:	*TRANSFER_SERVER_SECURITY_PROTOCOL*=SSL
Parameter type	Characters string. Possible values are: `jks` or `pkcs12`.
Parameter change assessment	After module reboot.
Description	Type of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRANSFER_SERVER_KEYSTORE_KEY_PASSWORD**
Parameter taken into account only if:	*TRANSFER_SERVER_SECURITY_PROTOCOL*=SSL
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the key in the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRANSFER_SERVER_KEYSTORE_ALIAS**
Parameter taken into account only if:	*TRANSFER_SERVER_SECURITY_PROTOCOL*=SSL
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Alias of the key in the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRANSFER_SERVER_TRUSTSTORE_PATH**
Parameter taken into account only if:	*TRANSFER_SERVER_SECURITY_PROTOCOL*=SSL
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRANSFER_SERVER_TRUSTSTORE_PASSWORD**
Parameter taken into account only if:	*TRANSFER_SERVER_SECURITY_PROTOCOL*=SSL
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password for the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRANSFER_SERVER_TRUSTSTORE_TYPE**
Parameter taken into account only if:	*TRANSFER_SERVER_SECURITY_PROTOCOL*=SSL
Parameter type	Characters string. Possible values are: `jks` or `pkcs12`.
Parameter change assessment	After module reboot.
Description	Type of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

5.3. Fixity Check Method

5.3.1. Description

Arcsys performs a fixity check on the processed files. You can choose the check method that best adapts to the required files and performance.

During each stage of the archiving and retrieval processes, Arcsys performs a fixity check on the processed files. This check is also performed on the records contained in the retention pool to ensure that there are no modifications or alterations in the processed files.

5.3.2. Changing the Fixity Check Method

The fixity check method can be selected on archiving, record by record. However, a default check method can be defined for all archiving processes.

The *DEFAULT_HASH_FUNCTION* parameter in the Arcsys Application Agent (see page 362, « etc/APPAGENT.properties File ») changes the fixity check method for all files processed by that application agent.

The chosen check method can also be changed through the web interface or the API at object level of a lot.

5.4. Using SSL

5.4.1. Introduction

Communications can be secured by SSL in several ways:

- via connection to the LDAP directory
- for RMI communications between the various Arcsys modules (Arcsys Engine, Arcsys Application Agent, Arcsys RMI API, Arcsys Auto-Archive Agent, Arcsys standard Clients) and the specific external developments using the Arcsys RMI API
- for HTTP communications performed by the REST APIs used in Arcsys REST API or ArcFF format control module
- for communications between Arcsys Transfer Server, Arcsys Transfer Service and Arcsys Engine, Arcsys RMI, TCP/IP and SOAP API, Arcsys Web Agent

To secure communications using SSL, you must:

- Generate the appropriate certificates
- In the case of LDAP, configure the LDAP directory

- Set new parameters in the configuration files of certain Arcsys modules

Important

When SSL is used to secure RMI communications, it needs to be simultaneously implemented on the server and on all clients according to the table below. You may find that the Arcsys Engines communicate between themselves and are client and server at the same time; the Arcsys Application Agent plays both the role of client and server at the same time.

Client / Server	Arcsys Engine	Arcsys Application Agent	API clients (Arcsys standard Clients, Arcsys Auto-Archive Agent, specific updates)
Arcsys Engine	✓	✓	
Arcsys RMI, TCP/IP and SOAP API			✓
Arcsys Application Agent	✓	✓	

The table is read as follows: when the SSL is positioned at Arcsys Engine level, it needs to be implemented at the same level in the other Arcsys Engines and Arcsys Application Agents because they are clients of the Arcsys Engine.

There is the same rule for securing communications with all the modules communicating with Arcsys Transfer Server:

- **Other Arcsys Transfer Servers**
- **Arcsys Transfer Services**
- **Arcsys Engines**
- **Arcsys Web Agents**
- **Arcsys API (RMI, SOAP or REST);**

5.4.2. Configuring the LDAP Directory

Implementing the SSL or STARTTLS protocol on LDAP involves configuring SSL or STARTTLS parameters on the LDAP server. This operation is native to each directory (OpenLDAP / Active Directory, etc.).

Remember the main stages in using a server certificate required where using OpenLDAP or 389 Directory Server.

Important

The examples given as follows are where the certification authority has been created internally. The procedure is quite similar where the certification authority is an external organization implementing certificates that will be supplied by that external organization.

5.4.2.1. OpenLDAP Example

1. Creating the CA certificate:

```
>/usr/share/ssl/misc/CA.sh –newca
```

2. Generating the server private key:

```
>openssl req –newkey rsa:1024 –nodes –keyout newreq.pem –keyform PEM –out newreq.pem
 outform PEM
```

Important

The common name for creating the server certificate matches the name of your server machine. Check that you can convert the name into its IP address (via a DNS server or via your /etc/hosts file).

3. Creating the server certificate and getting the CA signature:

```
>/usr/share/ssl/misc/CA.sh –sign  (...)
Signed certificate is in newcert.pem
```

4. Configuring the value in the slapd.conf configuration file:

The following table gives the correspondence between the files produced previously and the parameters of the OpenLDAP configuration file `slapd.conf`.

File	Generation stage	Parameter for slapd.conf
demoCA/cacert.pem	1	TLSCACertificateFile
newreq.pem	2	TLSCertificateKeyFile
newcert.pem	3	TLSCertificateFile

Important

Careful with the permissions of the file containing the server key (TLSCertificateKeyFile), which must be "read only" for the user starting up the LDAP.

You can optionally request a client certificate for establishing the session. This is done by the *TLSVerifyClient* parameter in `slapd.conf`. The summary table gives the different values that it can take:

TLSVerifyClient parameter value	Meaning
never	The server never requests a certificate from the client.
allow	The server requests a certificate from the client. If it doesn't have one or it is not checked, authentication can continue nonetheless without causing an error.
try	The server requests a certificate from the client. If it doesn't have one, authentication can continue, but if it has one and it cannot be checked, the session ends immediately.
demand	The server requires a valid certificate from the client.

Configuring the SLAPD Launch:

According to the operating mode desired, the following shows how to configure the execution of the SLAPD program. An example of the LDAP startup script is shown:

- Connecting to the LDAP in normal or STARTTLS mode:

```
SLAPD_URLS="ldap:///"
```

- Connecting to the LDAP in SSL mode only:

```
SLAPD_URLS="ldaps:///"
```

- Connecting to the LDAP in normal, SSL or STARTTLS mode:

```
SLAPD_URLS="ldap:/// ldaps:///"
```

5.4.3. Configuring Arcsys to Manage SSL on LDAP or RMI

You need to generate the truststore files (and optionally, keystore) and then configure each Arcsys module to use the SSL protocol (or STARTTLS for the LDAP).

5.4.3.1. Generating the Keystore File

This file must contain the certificate in JKS format. This certificate, which contains a private key, will be used by Arcsys to authenticate it with the LDAP.

To obtain a certificate signed by the certification authority, you should use the Java **keytool** command as well as the **OpenSSL** command. Creating this certificate involves 3 steps:

1. Creating the client certificate:

```
keytool -genkey -keyalg rsa -v -alias hostclient -keystore keystore.jks
```

2. Creating a signed certification request:

```
keytool -certreq -keystore keystore.jks -alias hostclient -file csr
```

3. Step 3 – Getting the CA signature:

```
openssl ca -in csr -out certificate -notext
```

The keystore file can be generated from the client certificate that has been signed by the certification authority.

```
keytool -import -keystore keystore.jks -alias hostclient -file certificate
```

Note

Instead of steps 2 and 3, you can also generate a self-signed certificate (where securing the RMI protocol, for example).

```
keytool -export -alias hostclient -keystore keystore.jks -rfc -file cetificate.
cer
```

5.4.3.2. Generating the Truststore File

This file must contain the CA certificate in JKS format. This certificate, which contains a public key, will be used by Arcsys to verify LDAP authentication.

It can be generated using the Java **keytool** command.

```
keytool -import -trustcacerts -keystore truststore.jks -alias cacert -file cacert.pem
```

Note

For self-signing certificates, the import command is the following:

```
keytool -import -alias certificate -file certificate.cer -keystore truststore.
jks
```

5.4.3.3. Configuring the Arcsys Modules

The Arcsys modules (Arcsys Engine, Arcsys Web Agent, Arcsys API (RMI, SOAP or REST)) need to be configured to use the SSL or STARTTLS protocol (*LDAP_SECURITY_PROTOCOL* for the *LDAP directory*, *RMI_SECURITY_PROTOCOL* for the RMI

connection) and the truststore and eventually the keystore (*TRUSTSTORE_FILE* and *KEYSTORE_FILE* parameters).

- *LDAP_SECURITY_PROTOCOL*: Indicates the security protocol to be used to connect to the LDAP directory. Possible values: NONE / SSL / STARTTLS (the TLS value is accepted, it is equal to the SSL value).
- *COMPONENT_API_RMI_SECURITY_PROTOCOL* (for a component that acts as an RMI server: Arcsys RMI API;Arcsys Application Agent; Arcsys Engine) or *RMI_API_SECURITY_PROTOCOL* (for a component that only acts as an RMI client: Arcsys Auto-Archive Agent; Arcsys standard Clients): Indicates the security protocol to be used for incoming and outgoing RMI connections (to and from the Arcsys Engine). Possible values: SSL / NONE.
- *TRUSTSTORE_FILE*: Location of the truststore file. This file is used for securing both LDAP and RMI. You can inform it in the etc directory of the component, for example. This parameter is optional.
- *TRUSTSTORE_PASSWORD*: Encrypted password of the truststore file (passphrase). This parameter is optional.
- *KEYSTORE_FILE*: Location of the keystore file. This file is used for securing both LDAP and RMI. You can, for example, set it in the component's directory, `etc`. This parameter is optional.
- *KEYSTORE_PASSWORD*: Encrypted password of the keystore file (passphrase). This parameter is optional.

5.4.3.4. Keytool (Java)

For more information on this command, consult the following documentation: http://java.sun.com/docs/books/tutorial/security1.2/summary/tools.html

5.4.3.5. OpenSSL

For more information on this command, consult the following documentation: http://www.openssl.org/docs/apps/openssl.html

5.5. Protecting the Arcsys Configuration File

5.5.1. Overview

Certain Arcsys configuration files are sensitive and may contain passwords (access to the JDBC database, to the LDAP administrator, etc.). A parameter enables Arcsys to

control that the access right to a configuration file is the same as a fixed value. Failing this, an error is notified and Arcsys cannot start up the module concerned.

Note

This function is only available in Unix.

5.5.2. Parameter Description

The parameter is called *CURRENT_FILE_RIGHT_MUST_BE* and is optional (if left blank or 0, it is ignored). It can be set in any configuration file of the following modules with the ".properties" suffix:

- Arcsys RMI, TCP/IP and SOAP API
- Arcsys Application Agent
- Arcsys Engine
- Arcsys Web Agent
- Arcsys Auto-Archive Agent
- Arcsys standard Clients

Its value is an octal number (like a classic Unix chmod).

Example: if *CURRENT_FILE_RIGHT_MUST_BE* is 600 in `ENGINE.properties`, only the root is authorized to read this file. If you do a `chmod 777` on this file and restart Arcsys, an error message appears in the logs and stops the Arcsys Engine from starting up.

6. Alerts

To monitor Arcsys operation, the various modules can be configured to issue alerts to a monitoring server (Centreon, Nagios, etc.) in the event of error or a change in status. Arcsys supports SNMP protocol and systemd notifications.

6.1. Configuration File Path

ALERT_PROPERTIES_FILE: Configuration of the alerts is done using a configuration file whose file path is defined by the *ALERT_PROPERTIES_FILE* parameter for the API (`ARCSYS_RMI_API.properties`), the Arcsys Engine (`ENGINE.properties`), the Arcsys Application Agent (`APPAGENT.properties`), the Arcsys Web Agent (`WEBAGENT_configuration.properties`), the Arcsys REST API (`arcsys-rest.properties`), ArcFF format control module (`arcff.properties`) and by the *notify_conf* parameter for the Arcsys Transfer Server (`transferServer.conf`) and the Arcsys Transfer Service (`transferService.conf`). The contents of this file are defined below (see Page 47).

6.2. notify.conf File

snmp_host_name: Optional parameter giving the name of the Arcsys module host issuing the SNMP notifications. It cannot contain the value "localhost".

snmp_server_ip: Mandatory parameter giving the IP address of the server to which SNMP notifications are sent.

snmp_port_receiver: Optional parameter giving the remote port to which SNMP notifications are sent. Default value: `161`

snmp_port_sender: Optional parameter giving the local port through which SNMP notifications are sent. Default value: `162`

snmp_retry: Optional parameter indicating the number of attempts in sending an SNMP notification. Default values: `3`

snmp_timeout: Optional parameter indicating the timeout for sending an SNMP notification. Default value: `5000`

snmp_community: Optional parameter indicating the SNMP community security name for sending SNMP notifications. Default value: `public`

requests_alerts: Parameter giving the protocol on which alerts regarding Arcsys requests are sent. An alert of this type is sent to signal a change of status in an Arcsys request. Possible value: `snmp`. By default, no value (alerts deactivated).

hardware_failure_alerts: Parameter giving the protocol on which alerts regarding hardware errors are sent. An alert of this type is sent to signal a hardware error. Possible value: `snmp`. By default, no value (alerts deactivated).

software_failure_alerts: Parameter giving the protocol on which alerts regarding software errors are sent. An alert of this type is sent to signal a software error. Possible value: `snmp`. By default, no value (alerts deactivated).

software_communication_failure_alerts: Parameter giving the protocol on which alerts regarding communication errors are sent. An alert of this type is sent to signal a communication error between an Arcsys module and another module (Arcsys or external). Possible value: `snmp`. By default, no value (alerts deactivated).

component_status_change_alerts: Parameter giving the list of protocols (separated by commas) on which alerts regarding changes in module status are sent. An alert of this type is sent to signal that the software component has changed status (started up, paused, stopped, etc.). Possible value: `snmp`,`systemd`. By default, no value (alerts deactivated).

request_alerts_snmp_on_status: Parameter giving the list of request statuses (separated by commas) to be considered when sending an alert for Arcsys requests in SNMP notification form. Possible values: `EACH / INITIALIZED / VALIDATED / WORKING / PROCEEDED / CLOSED / ENDED / ERROR / ABORTED`. Default values: `ERROR, ABORTED`

request_alerts_snmp_on_types: Parameter giving the list of request types (separated by commas) to be considered when sending an alert for Arcsys requests in SNMP notification form. Possible values: `ALL / A / R / O / M / C / S / P / D / Y / X / E / L`. Default value: `ALL`.

hardware_failure_alerts_snmp_minimal_severity: Parameter indicating the minimum severity level of a hardware error to be used when sending an alert in SNMP notification form. Possible values: `WARNING / CRITICAL`. Default value: `CRITICAL`.

software_failure_alerts_snmp_minimal_severity: Parameter indicating the minimum severity level of a software error to be used when sending an alert in SNMP notification form. Possible values: `WARNING / CRITICAL`. Default value: `CRITICAL`.

software_communication_failure_alerts_snmp_minimal_severity: Parameter indicating the minimum severity level of a communication error to be used when sending an alert in SNMP notification form. Possible values: `WARNING / CRITICAL`. Default value: `CRITICAL`.

arcsys_component_status_change_snmp_on_status: Parameter giving the list of statuses (separated by commas) to be used when sending an alert in SNMP notification form where there is change in status of a software component. This functionality is only implemented for the Arcsys Transfer Server. Default values: STARTED_AND_WORKING, SHUTDOWN

7. Authentication

Arcsys is a multi-user system. This means that different people can have access to the applications with different rights and roles, according to the structures.

In order to limit administration tasks, Arcsys connects using existing authentication mechanisms (LDAP directories). This centralizes user management and permissions.

7.1. Changing a LDAP Attribute

7.1.1. Effects of Changes

Only the Arcsys Engine, Arcsys Web Agent , Arcsys RMI, TCP/IP and SOAP API, Arcsys REST API contain parameters linked with LDAP authorization server (server URL, etc.). These parameters are described in the following paragraph.

The files to edit when changing one of the attributes of the LDAP directory are the following:

- page 284, « ENGINE.properties File » file of the Arcsys Engine.
- etc/WEBAGENT_configuration.properties file of the Arcsys Web Agent(see Page 397).
- etc/ARCSYS_RMI_API.properties of the Arcsys RMI, TCP/IP and SOAP API (see Page 162).
- etc/arcsys-rest.properties of the Arcsys REST API (see Page 173).

7.1.2. Description of the different LDAP Parameters

7.1.2.1. General specifications

LDAP_PROPERTIES_FILE: Optional parameter giving the location of a properties file that contains all the LDAP properties. This enables all LDAP connection properties to be externalized in a unique file.

Parameter name	**LDAP_TYPE**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	A value from: `OPEN_DJ` / `OPEN_DS` / `OPEN_LDAP` / `ACTIVE_DIRECTORY` / `TIVOLI_DIRECTORY_SERVER` / `SUN_ONE_DIRECTORY_SERVER`. Please note that to use 389 Directory Server you must choose the `OPEN_LDAP` value.
Shipped value	`OPEN_LDAP`

Parameter name	**LDAP_MODE**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Optional parameter giving the LDAP connection mode. In `STANDARD` connection mode, users and groups are in locations determined by the directory. In `GENERIC` and `FIRST_GROUP` connection modes, LDAP searches are performed to be able to search for users and groups found in different locations of the directory (see LDAPv3 specification and `ldapsearch` utility documentation). In these modes, only users present in at least one group can be recovered. In `GENERIC` , all users and groups are retrieved, and then groups are associated with users (therefore, there are fewer requests to the LDAP server, but more RAM is used). In `FIRST_GROUP` mode, the groups are searched first, then each user is retrieved from the LDAP server (there are more requests to the LDAP server, but less RAM is used). Possible values: `GENERIC`/ `STANDARD` / `FIRST_GROUP`
Shipped value	`GENERIC`

Parameter name	**LDAP_SERVERS**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	List of LDAP server codes then defined in *<xxx>.HOST* and *<xxx>.PORT* form. For example, if you define `LDAP_SERVERS=SERVER1,SERVER2` you must define the following lines: `SERVER1.HOST`, `SERVER1.PORT`, `SERVER2.HOST`, `SERVER2.PORT`. When more than one LDAP server is specified, and where Arcsys cannot contact the first server, it will try to establish a connection with the second, and thus successively (i.e. primary and secondary LDAP).
Shipped value	`SERVER1`

7.1.2.2. Arcsys attribute names (parameters shared by two LDAP modes)

LDAP_ARCSYS_ATTRIBUT_RIGHT_NAME : Specific Arcsys attribute to define permissions. Default value: `arcsysRight` ;

LDAP_GLOBAL_ATTRIBUT_MAIL_NAME : Global attribute defined for users which defines the mail. It enables the user's email parameters to be located in the LDAP directory;

7.1.2.3. STANDARD mode parameter

Parameter name	**LDAP_REFERRAL_FOLLOW**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Indicates whether referrals should be followed or not. Possible values: `follow` or `ignore`
Shipped value	`follow`

Parameter name	**LDAP_ADMIN_LOGIN**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Name of a user with sufficient rights to consult the entire LDAP and, in the case of the Arcsys Web Agent, to change a password (for example, for Active Directory, this user must be part of the Administrators group). If `anonymous` is chosen, an anonymous connection is used (under the premise that this is authorized by the specified directory). Such a connection cannot be used to write in the LDAP; therefore the `anonymous` login cannot be used if you want to enable the password change option.
Shipped value	`anonymous`

Parameter name	**LDAP_ADMIN_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the user with sufficient rights to consult the entire LDAP. If *`LDAP_ADMIN_LOGIN`* is `anonymous`, then the value of this parameter is ignored.
Shipped value	No shipped value.

Parameter name	**LDAP_ADMIN_PATH**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Access path (in LDAP) to the user with sufficient rights to view the entire LDAP. If *`LDAP_ADMIN_LOGIN`* is `anonymous`, then the value of this parameter is ignored.
Shipped value	`dc=infotel,dc=com`

Parameter name	**LDAP_USER_PATH**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Access path (in LDAP format) in the user directory.
Shipped value	`ou=users,dc=infotel,dc=com`

Parameter name	**LDAP_ARCSYS_GROUP_PATH**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Access path (in LDAP) to the Arcsys groups created in the LDAP.
Shipped value	`or=arcsysgroup,ou=groups,dc=infotel,dc=com`

Parameter name	**LDAP_USER_TYPE_LIST**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	List of attributes used to define a user.
Shipped value	`cn,uid`

Parameter name	**LDAP_GROUP_TYPE**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Attribute used to define the Arcsys groups.
Shipped value	`cn`

Parameter name	**LDAP_DISPLAY_NAME_ATTRIBUTE_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Global attribute used to read the user name that will be displayed in the web interface.
Shipped value	`sn`

7.1.2.4. Parameters in GENERIC mode

Parameter name	**LDAP_AUTHENT_DN**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The DN/login of a user with sufficient rights to consult the entire LDAP and, in the case of the Arcsys Web Agent, to change a password (for example, for Active Directory, this user must be part of the Administrators group). If `anonymous` is chosen, an anonymous connection is used (under the premise that this is authorized by the specified directory). Such a connection cannot be used to write in the LDAP; therefore an `anonymous` login cannot be used if you want to enable the password change option.
Shipped value	`anonymous`

Parameter name	**LDAP_AUTHENT_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the user with sufficient rights to consult the entire LDAP. If *`LDAP_AUTHENT_DN`* is `anonymous`, then the value of this parameter is ignored.
Shipped value	No shipped value.

Parameter name	**LDAP_USER_LOGIN_ATTRIBUTE_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Attribute name of an LDAP user entry containing the user login. This attribute must not change as it is stored in the database.
Shipped value	`cn`

Parameter name	**LDAP_GROUP_ID_ATTRIBUTE_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Attribute name of an LDAP group entry containing the group name. This attribute must not change as it is stored in the database.
Shipped value	`cn`

Parameter name	**LDAP_USER_DISPLAY_NAME_ATTRIBUTE_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Attribute name of an LDAP user entry containing the name to display for a user.
Shipped value	`sn`

Parameter name	**LDAP_GROUP_DISPLAY_NAME_ATTRIBUTE_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Attribute name of an LDAP group entry containing the name to display for a group.
Shipped value	`sn`

Parameter name	**LDAP_GROUP_MEMBERSHIP_ATTRIBUTE_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Attribute name of an LDAP group entry containing the user DN for a group.
Shipped value	`member`

Parameter name	**LDAP_USER_SEARCH_BASE_DN**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Base DN of the LDAP server for user searches.
Shipped value	`ou=users,dc=infotel,dc=com`

Parameter name	**LDAP_USER_SEARCH_SCOPE**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Scope of the user search in the LDAP server.
Possible values	• `BASE`: only the entry defined by the base DN is considered for the search • `ONE`: only the entries immediately below the entry defined by the base DN are considered for the search, the entry defined by the base DN is excluded • `SUB`: all the entries below the entry defined by the base DN and the entry defined by the base DN are considered for the search • `SUBORDINATE_SUBTREE`: all the entries immediately below the base DN are considered for the search, the entry defined by the base DN is excluded
Shipped value	`SUB`

Parameter name	**LDAP_USER_SEARCH_FILTER**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	User search filter in the LDAP server.
Shipped value	`(objectClass=inetOrgPerson)`

Parameter name	**LDAP_GROUP_SEARCH_BASE_DN**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Base DN of the LDAP server for group searches.
Shipped value	`or=arcsysgroup,ou=groups,dc=infotel,dc=com`

Parameter name	**LDAP_GROUP_SEARCH_SCOPE**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Scope of the group search in the LDAP server.
Shipped value	`SUB`

Parameter name	**LDAP_GROUP_SEARCH_FILTER**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Search filter for groups in the LDAP server.
Shipped value	`(objectClass=groupOfNames)`

Parameter name	**LDAP_FOLLOW_REFERRALS**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies if the LDAP server referral entries should be monitored during LDAP searches by users and groups.
Shipped value	`false`

Parameter name	**LDAP_ALIAS_DEREFERENCING_POLICY**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Policy for dereferencing aliases during LDAP user and group searches. Possible values: `FINDING` (de-references the aliases during search of the entry defined by the base DN but it is advised you don't do this when searching subordinates) / `SEARCHING` (de-references the aliases during search of the subordinates defined by the base DN but it is advised you don't do this when searching the entry defined by the base DN itself) / `ALWAYS` (always de-reference aliases) / `NEVER`(never de-reference aliases)
Shipped value	`NEVER`

7.1.2.5. Security Characteristics

See Configuring Arcsys to manage SSL on LDAP or RMI.

7.2. Authentification using Arcsys Engine

7.2.1. Introduction

The Arcsys Engine uses LDAP parameters to connect to the LDAP directory. It also uses 3 users specified in the configuration files for different identifications. The following provides details on these users.

7.2.2. Different Users

1 - Startup user of the Arcsys Engine

– The login (*USERNAME*) and the password (*USERPASSWORD*) are found in the page 284, « ENGINE.properties File »;

– It is only used to call the script: [launch_engine.sh (or .bat) script].

2 - Command execution user

– The login (*ENGINE_API_USERNAME*) and the password (*ENGINE_API_USERPASSWORD*) are found in the etc/ADMINCOMMAND_params.properties file;

– It is used to call the scripts: [start_engine.sh (or .bat) script], [status_engine.sh (or .bat) script], [stop_engine.bat (or .sh) script], [shutdown_engine.sh (or .bat) script].

Note

This user must have the GRANT_ALL or ADMIN right.

3 - LDAP configuration retrieval user

– The login (*LDAP_ADMIN_LOGIN*) and the password (*LDAP_ADMIN_PASSWORD*) are found in page 284, « ENGINE.properties File »;

– This is used to retrieve the LDAP properties (mail, etc.) for all Arcsys users;

– The path used for identifying this user is *LDAP_ADMIN_PATH* in page 284, « ENGINE.properties File »;

– This user can be "anonymous" or a valid user (Note: In this case, it cannot be an alias or a reference).

7.3. JAAS Settings File

7.3.1. Overview

This file contains the parameters of the Java authorization and authentication service.

7.3.2. Description

The format of the JAAS file is defined in:

http://java.sun.com/j2se/1.5.0/docs/guide/security/jaas/tutorials/LoginConfigFile.html

The inputs are:

```
<name used by application to refer to this entry> {
   <LoginModule> <flag> <LoginModule options>;
   <optional additional LoginModules, flags and options>;
};
```

7.4. Declaring a New User

A new user is declared on the LDAP directory of the organization.

To assign this new user the necessary rights to use the product, they need to be associated with a predefined Arcsys group (this operation is also done on the LDAP directory).

8. Connecting to the Arcsys Web Agent Using HTTPS

To connect to the Arcsys Web Agent using HTTPS, you need to:

- Generate a keystore file,
- Configure the application server hosting the web application.

Note

To ensure that the Arcsys updates do not overwrite the HTTPS configuration, the parameters should be set at server instance level and not in the Arcsys Web Agent. The procedure for doing this is described below.

8.1. Configuring Through the Administration Console

You can choose to connect exclusively via HTTPS. Just mark the "SSL Listen Port Enabled" option.

The Keystores & SSL tab allows you to specify the corresponding keystore.

Note

If you want to connect to the web agent via HTTPS only, deselect the "Listen Port Enabled" option.

9. Digital signature

Important

This features set can only be used by buying ArcSign Option.

9.1. Principle

The electronic signature is a mechanism guaranteeing the integrity of an electronic document and the authentication of the author, to ensure that they cannot be challenged.

The aim of electronic signatures is double:

- to guarantee the integrity of a document, i.e. to ensure that the document has not been altered between signature and consultation
- to authenticate its author, i.e. to ensure the identity of the person signing the document

Electronic signatures have the following characteristics:

- **authentic:** the identity of the signatory must be traceable with certainty
- **forgery-proof:** one person cannot impersonate another one by forging his signature.
- **non-reusable:** the signature is part of the signed document and cannot be moved to another document
- **inalterable:** once a document has been signed, it cannot be altered
- **irrevocable:** the person who signed it cannot contest it

9.2. Implementing a Signing Agent

9.2.1. Introduction

A signing agent is an agent that signs all the objects that it archives. It uses the certificate indicated in the configuration file to sign the hash values of the objects for archiving.

Hence this solution can be useful for forcing the use of the signature for all archiving using that agent.

This solution automates the signing operation to a certain extent.

Furthermore, the signing agent can create a common signature file that can be useful for reducing the number of archiving tokens and for super-signing in the future.

9.2.2. Operation

Arcsys only signs the file hashes that it archives: it does not sign the file itself but the hash of that which is dependent on the Arcsys archive configuration. These signatures are saved in the database, in the manifest and if necessary, in the common signature file.

9.2.3. Stages

9.2.3.1. Creating the Keystore

This file must contain the certificate in JKS format. This certificate, which contains a private key, will be used by Arcsys to authenticate the signature with the application agent.

To obtain a certificate signed by the certification authority, you should use the Java **keytool** command:

```
keytool -genkey -keyalg rsa -v -alias yourAliasName -keystore votreKeystore.jks
```

9.2.3.2. Configuring the Agent

The application agent must be configured as a signing agent (see page 362, « etc/APPAGENT.properties File »). Here is an example of parameters configuration:

```
# ----- <SIGNING parameters> -----
# SIGNING_AGENT -> If true, the agent signs object (true|false)
SIGNING_AGENT=true
# SIGNING_KEYSTORE_FILE -> Path of keystore which contains the SIGNING_PRIVATE_KEY_ALIAS
SIGNING_KEYSTORE_FILE=/opt/Infotel/ArchivingProduct/keystore/yourKeystore.jks
SIGNING_KEYSTORE_PASSWORD=22688A72369C087BF30CC021471EBC9A
SIGNING_PRIVATE_KEY_ALIAS=yourAliasName
SIGNING_PRIVATE_KEY_PASSWORD=22688A72369C087BF30CC021471EBC9A
```

The *SIGNING_KEYSTORE_PASSWORD* and *SIGNING_PRIVATE_KEY_PASSWORD* encrypted passwords correspond respectively to the keystore password and the private key password used.

9.2.3.3. Configuring the Agent to Create the Common Signature File

You need to configure the Arcsys Application Agent as a signing agent and change the file (see page 362, « etc/APPAGENT.properties File ») to include the following parameters:

```
#----- <common Signatures file parameters> -----
# USE_COMMON_SIGNATURE_FILE-> If true, the agent create a common signature file for each
 lot
USE_COMMON_SIGNATURE_FILE=true
# COMMON_SIGNATURE_FILE_FOLDER --> default retrieval folder for common signatures file
 (absolute or relative path)
COMMON_SIGNATURE_FILE_FOLDER=commonSignatureFileFolder/
```

9.3. Signature validator module plugin (ArcVerif Option)

The Signature validator module is a web service that validates digital signatures. It is called by ArcFF format control module through an embedded client to process validation requests. During validation, the web service relies on an external library to verify signatures. Currently, only the DSS (Digital Signature Services) library is supported.

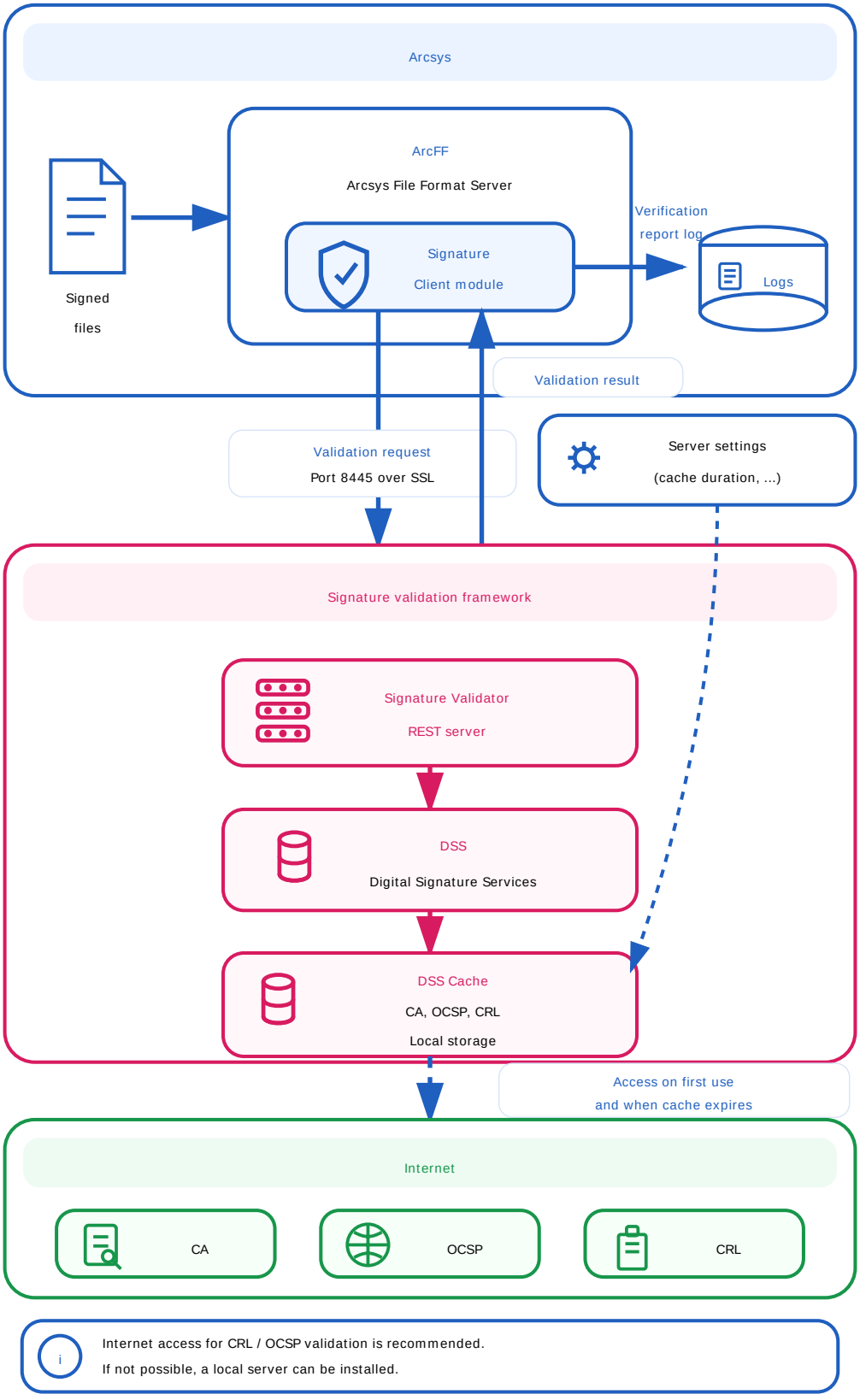


Figure 9.1. Signature Validator architecture

9.3.1. Specifications

Electronic signature formats determine the way in which the electronic signature is integrated into the electronic document. The three main formats are:

- XAdES (XML Advanced Electronic Signatures)
- PAdES (PDF Advanced Electronic Signatures)
- CAdES (CMS Advanced Electronic Signatures).

The Signature validator module is only able to validate PAdES.

9.3.2. Implementation

Electronic signature validation is based on DSS (Digital Signature Services). DSS is an open-source software library for electronic signature creation and validation. DSS supports the creation and verification of interoperable and secure electronic signatures in line with European legislation. DSS particularly aims to follow the eIDAS Regulation[1] and related standards closely.

The Signature validator module requests the DSS Web Service to validate the electronic signature. This validation follows the policy defined in the `signature-validator-ws.properties` file to validate all or part of the electronic signature:

- **hash**: the document has not been altered
- **certificate and authority**: the signer is authenticated
- **timestamp**: the signer is authenticated by a valid certificate
- **revocation**: the signer certificate had not been compromised
- **irrevocable**: the person who signed it cannot contest it

9.3.3. Configuration requirements

In order for the Signature validator module to operate properly, it must be correctly linked to the ArcFF format control module. This configuration step is required to enable communication between the two components.

In practice, it is the ArcFF format control module that initiates the validation of electronic signatures by sending requests to the Signature validator module. The actual signature analysis and validation are entirely handled by the Signature validator module.

[1] https://ec.europa.eu/digital-building-blocks/sites/x/5YTXGw

The configuration enabling communication between the two modules is described in detail on page 110, « Description File for a Format Policy » and on page 442, « etc/internal-signature-validation/signature-validation.properties File ».

10. Implementing the External Timestamp

This solution indicates the precise date and time with a timestamp. The principle is to contact a timestamping authority (called TSA: Time Stamp Authority) during archiving.

Timestamping justifies the existence of a data item at a precise moment. This way, you can prove that a file was archived on a precise date by an Arcsys Application Agent (and hence, the timestamp is saved by the TSA and the hash function used for the timestamp is considered as valid).

10.1. Operation

Timestamping is determined at collection level.

If *external timestamp* is set to no in the collection, external timestamp is disabled. Internal timestamp is then used. Internal timestamp has no associated configuration. It means that the timestamps inserted in the database will become the system timestamps. In particular, *TIMESTAMP_SOURCE_TYPE* parameter has no effect in this case.

If the external timestamp is activated for the collection being archived, an External TSA will be called. According to the agent's parameters, Arcsys will timestamp:

- the hash of each of the files archived;
- the hash of the signature of each of the files archived if it exists;
- only the hash of the common signature file (or its signature) if this has been created by the agent.

10.2. Stages

10.2.1. Configuring the Agent

You must configure the application agent (see page 362, « etc/APPAGENT.properties File »).

The following table presents the number and nature of generated timestamp tokens, according to the configuration chosen. The recommended configurations (with or without signature) are highlighted by a green background.

USE_COMMON_ SIGNATURE_FILE	false	false	false	true	true
TIMESTAMP_ SOURCE_TYPE	DOCUMENT	DOCUMENT	DOCUMENT_ SIGNATURE	DOCUMENT	DOCUMENT_ SIGNATURE
SIGNING_AGENT	true	false	true	true	true

Number and nature of timestamp tokens	N (for each document's hash value)	N (for each document's hash value)	N (for each document's signed hash value)	1 (for the hash value of the common signature file)	1 (for the signed hash value of the common signature file)

Table 10.1. Combinations of parameters for timestamping

The combinations of parameters that are not present in the above table are incoherent.

10.3. Recommended Configuration

When ArcSign is installed, it is recommended you to configure the application agent as a signing agent, to activate timestamping and to generate a common signature file in order to minimize costly performance in terms of timestamping tokens and prepare for super-signing.

11. Network

This section presents the different Arcsys settings associated with the network connection.

11.1. Ports

11.1.1. Introduction

As indicated in the Arcsys Presentation Manual, Arcsys is scalable and allows to define the network ports used.

This guarantees that the list of ports used by Arcsys can be changed, whatever the configuration used in the network.

Important

Any change in the configuration of the network ports used must be done while Arcsys is shut down.

11.1.2. Arcsys Engine Network Port

The *COMPONENT_API_PORT* parameter of the Arcsys Engine (see page 284, « ENGINE.properties File ») defines the RMI port number of the Arcsys Engine. If that port causes a problem (e.g. used by another application), it can be changed.

The other parameter to change is then:

- The parameter for each application agent (see page 362, « etc/ APPAGENT.properties File »).

11.1.3. Arcsys API Network Port

The *COMPONENT_API_PORT* parameter of the API (see page 162, « etc/ ARCSYS_RMI_API.properties File ») defines the RMI port number of the Arcsys API.

If this port number changes, you must also change:

- The *api_port_number* parameter of the Arcsys Transfer Server:page 200, « transferServer.conf File »;
- The *API_PORT_NUMBER* parameter of the standard clients:
  - page 458, « etc/ArchiveClient.properties File »;

- page 464, « etc/RestoreClient.properties File »;
- page 467, « etc/SearchClient.properties File »;

Note

Where changing the port, don't forget to change the configuration of all the specific client developments that use the Arcsys API.

11.1.4. Arcsys Transfer Server Network Port

The *component_port_number* parameter (see page 200, « transferServer.conf File ») of the Arcsys Transfer Server defines the Arcsys Transfer Server port number. You can change it if necessary (for example, when it is used by another application).

11.1.5. Arcsys Transfer Service Network Port

The *component_port_number* parameter (see page 379, « transferService.conf File ») of the transfer service defines the Arcsys Transfer Service port number. You can change it if necessary (for example, when it is used by another application).

11.2. Codes for Modules, IP and host names

11.2.1. Description

To make the system open, the Arcsys Application Agents, Arcsys Engines, Arcsys Transfer Servers and Arcsys Transfer Services have a unique code that is used to identify them in the configuration files or in the Arcsys interface. This way, the host of a module can be changed without it impacting the other modules.

This code is determined by the Arcsys administrator, who will choose a code with meaning for the context.

11.2.2. Configuring the Arcsys Modules

The following parameters apply:

- *COMPONENT_CODE*: Unique component code. This parameter is mandatory. The code must identify the module in the Arcsys installation. It is limited to 250 characters and must contain only alphanumerical characters ([a...z][A...Z][0...9]) without any spaces. For the Arcsys Engine, this code is used as the RMI service name under which the Arcsys Engine is deployed.

Note

The parameter is noted as *component_code* in the Arcsys Transfer Server and the Arcsys Transfer Service.

- *COMPONENT_HOST*: Module host. This parameter is optional. This is used to "force" the name under which the module will be registered in the Arcsys Database, overwriting the default name chosen by Arcsys. Use, for example, a `localhost`, an IP, a host name, etc.

Note

The parameter is noted as *component_host* in the Arcsys Transfer Server and the Arcsys Transfer Service. For these modules, there is also another parameter called *external_component_host*. This optional parameter defines the address with which the Arcsys Transfer Server or the Arcsys Transfer Service will be contacted from a module present on a different site than the Arcsys Transfer Server or the Arcsys Transfer Service.

- *COMPONENT_AUTO_REFERENCE_MODE*: Module referencing mode in the relational database: the possible values are by host name (`HOSTNAME`) or IP (`IP`). This parameter is mandatory.

Note

The parameter is noted as *component_auto_reference_mode* in the Arcsys Transfer Service.

This parameter is not currently available for the Arcsys Transfer Server. It is the contents of the `hosts` file that determine the host with which it is registered in the relational database.

11.2.3. Changing Parameter Values

You can change the code of a component or its host, but you must ensure that you do not change all the parameters at once so that the module is not duplicated in the relational database.

Note

An Arcsys Engine to register in the Arcsys Database uses the following algorithm:

- Does the Arcsys Engine exist with the code specified in *COMPONENT_CODE*?

- If yes, it is registered with the name by updating the Arcsys Engine host (IP or name after `COMPONENT_AUTO_REFERENCE_MODE`);
- If no, it searches to see if there is an Arcsys Engine with the same host (IP or host name) and the same port:
  - If yes, it updates the information for this Arcsys Engine in the Arcsys Engine list,
  - If no, it registers a new Arcsys Engine in the Arcsys Engine list.

11.3. Howto: change a component host

If you need to change the name of the host where at least one module is installed, or move a module from a host to another one, you need to proceed carefully.

There is an impact on Arcsys if the change involves one of the modules which is referenced by another component or by any specific development.

That is why we recommend the following procedure:

1. Search for the host name and/or the associated IP (if you have explicitly used IPs instead of host names) in all configuration files (`*.properties`, `*.conf`) of all modules, and replace all the found occurrences with the new name;
2. Apply the same operation on your specific developments if you have deployed any;
3. Restart the modules.

12. Log Files

Arcsys's default behavior for each of its components is to generate log files containing the error traces and warnings triggered by each component in a `log/` directory.

Nevertheless, you can modify this behavior, changing:

- The structure of the messages;
- The type of messages (information, error, etc.);
- The location of the generated file;
- File rotation;
- etc.

12.1. All Modules Except Arcsys Transfer Server and Arcsys Transfer Service

12.1.1. Overview

This paragraph describes the configuration of the log files generated by all the modules except the Arcsys Transfer Server and the Arcsys Transfer Service.

Arcsys uses the Log4J standard of the Apache Foundation (http://logging.apache.org/log4j/) for its components.

12.1.2. Configuration File

Log4J requires a file that defines the configuration of the log processes. By default, this file is `log4j2.xml` present in the `etc/` directory of each component. However, another file can be used. The location of this file is set by the `log4j.configurationFile` environment variable.

12.1.3. etc/log4j2.xml file

12.1.3.1. Definition

This file contains the settings for the log file. The logs are generated with the log4J Apache library.

12.1.3.2. Parameters

The parameters of this file depend on the information desired in the logs. Hence please consult the standard documentation of Log4J for any changes.

See: https://logging.apache.org/log4j/2.x/

12.2. On the Arcsys Transfer Server and the Arcsys Transfer Service

12.2.1. Parameters

This section describes the configuration of the log files generated by the Arcsys Transfer Server and the Arcsys Transfer Service.

To set log parameters, refer to « LogParam.cfg File for Arcsys Transfer Server », page 229 and « LogParam.cfg File for Arcsys Transfer Service », page 389.

12.2.2. Backing Up and Rotating Log Files

Rotating the log files can be done either using the internal rotation mechanism of the Arcsys Transfer Server or in a more generic way, by **logrotate**. This paragraph describes how to use and configure **logrotate** for Unix and Linux.

The **logrotate** options are found in one or more configuration files. Each configuration file can define global options and requires one or more log files to switch. The base configuration file is found in `/etc/logrotate.conf`. By default, any configuration file registered in `/etc/logrotate.d` will be taken into account.

Below shows an example that describes the configuration of **logrotate**:

```
[
# rotate for transferServer logs files.
#

/opt/Infotel/ArchivingProduct/ArcsysTransferServer/log/transferServer.log {
  create 600 user_name
  user_group notifempty
  daily rotate 10
  missingok
  nocompress
  prerotate # just if the program is really running

  if test -x /opt/Infotel/ArchivingProduct/ArcsysTransferServer/status_transferServer.sh
  then
    /opt/Infotel/ArchivingProduct/ArcsysTransferServer/status_transferServer.sh > /dev/null
    #if test "$?" == "13"
    if [ "$?" eq"13" ]
    then
```

```
      /opt/Infotel/ArchivingProduct/ArcsysTransferServer/reload_transferServer.sh > /dev/
null
    fi
  fi
  postrotate

  # just if the program is really running
  if test -x /opt/Infotel/ArchivingProduct/ArcsysTransferServer/status_transferServer.sh
  then
    /opt/Infotel/ArchivingProduct/ArcsysTransferServer/status_transferServer.sh > /dev/null

    #if test "$?" == "13"
    if [ "$?" eq"13" ]
    then
      /opt/Infotel/ArchivingProduct/ArcsysTransferServer/reload_transferServer.sh > /dev/
null
    fi
  fi
  endscript
}
```

The "**logrotate -fv /etc/logrotate.d/ArchivingProduct**" command forces rotation of the log file; you will find a new empty transferServer.log file used by the daemon and a backup version of the same file called transferServer.log.1.

The `reload_transferServer.sh` script is used to flush the logs (done in pre-rotate and post-rotate).

Note

– For a complete description on configuring Logrotate, check its documentation (`man logrotate`);

– When run, Logrotate performs the set actions on the configuration file only once and then stops. For this reason, it needs to be launched periodically using the cron daemon (or equivalent).

To configure the automatic launching of Logrotate using cron, check the associated documentation (`man cron`).

13. Functional Trace Files

Arcsys can generate traces by repository and/or collection, provided that the corresponding option is triggered in these entities. The activation of the trace on these entities can be done separately and independently. These traces are generated in the `trace` directory.

13.1. Modules Affected

The modules affected by traces are the following:

- Arcsys Engine
- Arcsys Web Agent
- Arcsys RMI, TCP/IP and SOAP API
- Arcsys REST API

14. Specific traces for media managers

Arcsys can generate specific traces for the activity encountered on media managers.

Arcsys supports operations on different storage media. For some of them, the Arcsys Transfer Server can generate additional traces, in addition to the standard logs. For more information about standard logging, see page 71, « Log Files ».

Additional traces are available for:

- tape storage managed with ArcMover Tape Option
- cloud storage managed with ArcMover S3 Option

This chapter describes the purpose of these traces and explains how to configure and use them for each supported storage type.

14.1. Traces for ArcMover Tape (ArcMover Tape Option)

See page 83, « Setting up specific traces for ArcMover Tape ».

14.2. Traces for Cloud media manager (ArcMover S3 Option)

When using ArcMover S3 Option, you may want to activate specific traces that will log operations handled on the cloud media manager in a specific trace file. This can be useful when encountering an issue with the cloud provider, or for administrative tasks or reporting needs.

Those traces are activated by setting the *cloud_trace_file* parameter, in the `cloud.conf` configuration file (See page 249, « etc/cloud/s3/<configuration_name>/cloud.conf File »). When this parameter is set, all operations performed by the Arcsys Transfer Server on cloud media managed by the corresponding configuration are recorded in the specified trace file. Each cloud zone is managed by a configuration file, and each configuration can apply to one or more zones. Traces are enabled for all zones associated with the configuration in which the trace file is defined. If several configurations specify the same trace file, all traces generated for the associated zones are written to this file. This mechanism allows you to define trace generation with different levels of granularity by using different configurations and parameter values.

14.2.1. Level of trace for Cloud media

The traces generated for Cloud can be of two different levels: `STANDARD` or `VERBOSE`. The level of traces is specified by the *cloud_trace_level* parameter, in the `cloud.conf`

configuration file (See page 249, « etc/cloud/s3/<configuration_name>/cloud.conf File »).

`STANDARD` traces are used to log basic activity on the cloud provider: write, read and delete operations or errors encountered.

`VERBOSE` level gives more detailed information regarding each part (for multi-partitioned operation) or the beginning of operations.

14.2.2. Format of traces

The log entries are of the form:

```
<timestamp> <pid> <filename> <bucket> <operation> <additional_information>
```

- The `<timestamp>` corresponds to the time of the operation, in the form of “YYYY-MM-DD HH:MM:SS,<microseconds>”.
- The `<pid>` corresponds to the identifier of the process.
- The `<filename>` corresponds to the object name on the bucket.

Note

The `<filename>` is usually formed like `lot_<lot_id>_seq_<seq_id>`. However, for objects coming from an injection with ArcRef Injector (ArcRef Option) it corresponds to the name of the file injected and has no link with the identifier of the lot. If the injected document is located in a sub-directory, the `<filename>` includes the sub-directory (i.e. `MyInjectionSubDirectory/MyInjectedFile`).

- The `<bucket>` corresponds to the name of the bucket in which the operation is performed.

The entries at `STANDARD` level are:

- successful read operation, formed as:

```
<timestamp> <pid> <filename> <bucket> READ <return_code> <optime> <size>
```

  - `<return_code>`: Arcsys Transfer Server error code, in hexadecimal.
  - `<optime>`: operation duration, in milliseconds.
  - `<size>`: size in bytes of data read.

- successful write operation, formed as:

```
<timestamp> <pid> <filename> <bucket> WRITE <upload_id> <return_code> <optime> <size>
```

  - `<upload_id>`: the identifier of the multipart upload of the file. In case of error, it can be used to search for associated information in the cloud provider's logs.
  - `<return_code>`: Arcsys Transfer Server error code, in hexadecimal.
  - `<optime>`: operation duration, in milliseconds.
  - `<size>`: size in bytes of data written.
- successful delete operation, formed as:

```
<timestamp> <pid> <filename> <bucket> DELETE <return_code> <optime>
```

  - `<return_code>`: Arcsys Transfer Server error code, in hexadecimal.
  - `<optime>`: operation duration, in milliseconds.
- successful delete of a specific version of a document, formed as:

```
<timestamp> <pid> <filename> <bucket> DELETE_VERSION <version_id> <return_code> <optime>
```

  - `<version_id>`: the identifier of the version deleted.
  - `<return_code>`: Arcsys Transfer Server error code, in hexadecimal.
  - `<optime>`: operation duration, in milliseconds.
- operation in error, formed as:

```
<timestamp> <pid> <filename> <bucket> ERROR <return_code> <error_message>
```

  - `<return_code>`: Arcsys Transfer Server error code, in hexadecimal.
  - `<error_message>`: the error message associated with the error.

The entries at `VERBOSE` level are those of `STANDARD` level, plus:

- initializing of an object to archive, formed as:

```
<timestamp> <pid> <filename> <bucket> INIT <size>
```

  `<size>`: size in bytes of the document.
- beginning of a write operation, formed as:

```
<timestamp> <pid> <filename> <bucket> BEGIN_WRITE <upload_id>
```

  `<upload_id>`: the identifier of the multipart upload of the file.
- successful write of a part of the document, formed as:

```
<timestamp> <pid> <filename> <bucket> SEND_PART <part_number> <return_code> <optime>
 <size>
```

- `<part_number>`: the index of the part.
- `<return_code>`: Arcsys Transfer Server error code, in hexadecimal.
- `<optime>`: operation duration, in milliseconds.
- `<size>`: size in bytes of data written in this part.

- successful delete of partial document (in case of error during the write operation), formed as:

```
<timestamp> <pid> <filename> <bucket> DELETE_UPLOAD <upload_id> <return_code>
```

- `<upload_id>`: the identifier of the multipart upload of the file.
- `<return_code>`: Arcsys Transfer Server error code, in hexadecimal.

- beginning of a read operation, formed as:

```
<timestamp> <pid> <filename> <bucket> BEGIN_READ <size>
```

`<size>`: size in bytes of data to read.

- successful read of a part of the document, formed as:

```
<timestamp> <pid> <filename> <bucket> GET_PART range <return_code> <optime> <size>
```

- `<return_code>`: Arcsys Transfer Server error code, in hexadecimal.
- `<optime>`: operation duration, in milliseconds.
- `<size>`: size in bytes of data read in this part.

- beginning of a delete operation, formed as:

```
<timestamp> <pid> <filename> <bucket> START_DELETE
```

14.3. Trace files rotation

The automatic rotation of ArcMover Tape and Cloud traces files is not supported by Arcsys Transfer Server. Although these trace files contain a limited number of entries, you may consider setting up your own rotation mechanism using system tools if their size grows significantly in your environment.

One possible way to implement system-based rotation of trace files, proposed below as an example, is to use **logrotate** with **cron**.

1. create a configuration file in `/etc/logrotate.d` directory. This file defines how **logrotate** should manage the trace files.

   An example of content for this file is:

```
/opt/Infotel/ArcsysTransferServer/log/cloud.log /opt/Infotel/ArcsysTransferServer/log/
arcmover.log {
    daily
    rotate 7
    compress
    dateext
    dateformat .%Y-%m-%d-%H-%M-%S
    missingok
    notifempty
    copytruncate
}
```

   The settings allow you to specify the behavior of **logrotate**:

   - `daily`: rotate the log once per day.
   - `rotate <nb>`: keep `nb` archived logs before deleting older ones.
   - `compress`: compress rotated logs (usually with gzip), producing files like: `<file>.<extension>.gz`
   - `dateext`: archive old versions of log files adding a daily extension like `YYYYMMDD` instead of simply adding a number.
   - `dateformat <format_string>`: Specify the extension for `dateext`. The specified pattern (`.%Y-%m-%d-%H-%M-%S`) generates a log extension similar to that used for standard log files of the Arcsys Transfer Server.
   - `missingok`: do not report an error if the log file does not exist.
   - `notifempty`: do not rotate the file if it is empty.
   - `copytruncate`: copy the current log to the rotated file, then truncate the original file in place.

     This is useful when the application/script keeps the log file open and cannot reopen it automatically after rotation (case of the trace file for ArcMover Tape, for example). Without `copytruncate`, the Arcsys Transfer Server would continue writing into the old renamed file instead of the new log file.

2. On most distributions, **logrotate** is already called every day via **cron** (script in `/etc/cron.daily` or **systemd** timer). In this case, there is nothing more to do. Otherwise, add a line in `/etc/crontab`:

```
59 23 * * * root /usr/sbin/logrotate /etc/logrotate.conf
```

15. Sending Mails

During certain events (end of archiving/retrieval request, changes in zones, zone check), the Arcsys Engine can send personalized alert mails using a defined SMTP server.

Note

The mail configuration presented here is also used by the notification mechanism described in the Arcsys Functional Description Manual.

15.1. Configuring Mail Sending

15.1.1. Introduction

All the parameters presented are part of the page 336, « Mail management » section of the page 284, « ENGINE.properties File ».

15.1.2. Configuring the SMTP Server

The MAIL_SMTP_SERVER and MAIL_SMTP_SERVER_PORT parameters define the SMTP server host and port.

15.1.3. Configuring the Sender

The MAIL_SENDER parameter indicates the name of the sender used.

15.1.4. Mail Recipient

As regards the recipient, this depends on the function:

- Where changing zones, the mail is sent to the person who created the collection containing the lot being changed zone (their mail is retrieved from the LDAP);
- Where checking zones, the recipient is the mail address specified in the CHECKZONE_SEND_MAIL_TO parameter.

15.1.5. Authentication

Where authentication is required in the SMTP server (this is specified by *MAIL_USING_PASSWORD*), the user name is indicated in *MAIL_LOGIN* and the encrypted password in *MAIL_PASSWORD*.

15.2. Customizing the Mail Template Option

15.2.1. Introduction

To send mails, Arcsys uses localized templates that can be modified and customized as required.

These files found in the /etc directory of the Arcsys Engine or the Arcsys Web Agent are called TEMPLATE_<function>_xx_XX.properties

Where xx defines the language and XX defines the country.

15.2.2. File Parameters

The different parameters of the template are:

- MAIL_TITLE: Subject of the mail sent.
- MAIL_LINE_HEADER_xx: Lines to add at the beginning of the mail. You can define as many rows as required.
- MAIL_LINE_FOOTER_xx: Lines to add at the end of the mail. You can define as many rows as required.

15.2.3. Specific Parameters

According to the functions, supplementary parameters can be defined.

These parameters can contain predefined variables (where the syntax is %Name of the variable%). These variables, which are documented in each template, are automatically set by Arcsys when generating the mail.

15.3. Managing Errors in Mail Sending

15.3.1. Introduction

There are two types of error generated by Arcsys when sending mails:

- When sending, where the mail address of a user to be sent an alert in the LDAP directory is missing;
- A technical error when sending the mail itself.

15.3.2. User Mail Address Not Found in the LDAP

When the alert searches for the mail address of the user to be sent the mail in the LDAP directory (in the user "mail" attribute) and does not find this attribute, an error mail is automatically sent to the administrator defined in EVENT_ALERT_ADMIN_MAIL_LIST (page 284, « ENGINE.properties File »).

15.3.3. Technical Error When Sending the Mail

When the sending of the mail (stored in the directory indicated by the MAIL_DIRECTORY parameter) is rejected for some reason, the error appears in the Arcsys Engine logs and the mail is moved to the MAIL_DIRECTORY directory, to a sub-directory called Error.

If the administrator wants to recover the mail sending (for example, because the recipient address was temporarily unavailable but is now available), it should be moved to the root of the mail sending directory MAIL_DIRECTORY. On the next processing of the mails, a new attempt to send the mail will be made.

In the example Example of an Error in Mail Sending, an alert mail was moved to the sub-directory Error (MailAlert_181 directory). Seeing that sending could be tried again, the administrator moves the mail to the mail directory.

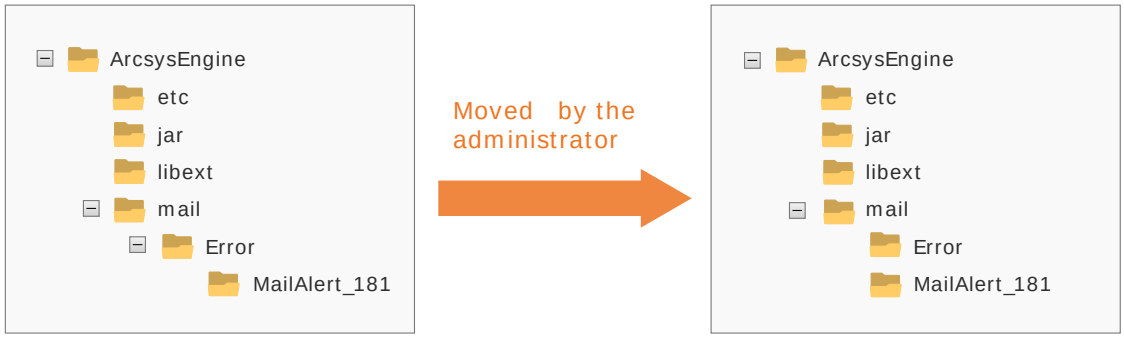


Figure 15.1. Example of an Error in Mail Sending

16. ArcMover Tape (ArcMover Tape Option)

16.1. Setting up specific traces for ArcMover Tape

When using ArcMover Tape, it is recommended to activate specific traces that will log operations handled on the tape library in a specific trace file. This can be useful when encountering an issue with the tape library, or for administrative tasks or reporting needs.

Those traces are activated by setting *arcmover_trace_file* parameter, in the `arcMover.conf` configuration file (See page 233, « etc/arcmover/arcMover.conf File »). When set, the operations carried out by the Arcsys Transfer Server on the tape library are recorded in the file specified by this parameter.

The format of the log entries depends on the type of operation. It can be:

- informative message (`INFO`), formed as:

```
<timestamp> INFO <message>
```

  - `<timestamp>`: in `YYYYMMDD-HHMNSS` format
  - `INFO`: INFOrmative log entry
  - `<message>`: informational message (e.g. "`ArcMover Server starting`", "`Tape SERIAL already loaded in drive SERIAL`"...)

- successful operation on a tape (`TAOP`), formed as:

```
<timestamp> TAOP <req_id> {W|R}[D][M] [<dismounted_volume_id>] <volume_id> <drive>
 <lot_id> <seq_id> <accesstime> <optime> [<optime_mount>] [<optime_dismount>] <size>
```

  - `<timestamp>`: in `YYYYMMDD-HHMNSS` format
  - `TAOP`: “TApe OPeration” log entry
  - `<req_id>`: Identifier of the request
  - `{W|R}`: Write or Read on the tape (one of the two must be present)
  - `[D]`: optional, indicates that a tape has been removed from the drive for this operation
  - `[M]`: optional, indicates that a tape has been mounted for this operation

- `[<dismounted_volume_id>]`: optional, depends on `[D]`. identifier of the tape that was dismounted. If `[D]` is present, `[<dismounted_volume_id>]` is present too
- `<volume_id>`: identifier of the tape that was used for the operation (if `[M]` is set, then this tape was mounted, otherwise it was already present in the drive)
- `<drive>`: serial number of the drive used
- `<lot_id>`: identifier of the lot
- `<seq_id>`: identifier of the sequence of the lot
- `<accesstime>`: duration of access to the media (including tape mounting/dismounting operations), in milliseconds
- `<optime>`: duration of the operation (Write or Read), in milliseconds
- `[<optime_mount>]`: mount operation duration, in milliseconds
- `[<optime_dismount>]`: dismount operation duration, in milliseconds
- `<size>`: size in bytes of data written or read

- operation in error on a tape (`TOFA`), formed as:

```
<timestamp> TOFA <req_id> <lot_id> {W|R|D|M} <volume_id> <drive> <return_code>
```

- `<timestamp>`: in `YYYYMMDD-HHMNSS` format
- `TOFA`: “Tape Operation FAilure” log type
- `<req_id>`: Identifier of the request
- `<lot_id>`: identifier of the lot
- `{W|R|D|M}`: The error was encoutered while writing (`W`) or reading (`R`) or dismounting (`D`) or mounting (`M`) a tape
- `<volume_id>`: Identifier of the tape that was used for the operation
- `<drive>`: serial number of the drive used
- `<return_code>`: Arcsys Transfer Server error code, in hexadecimal

- operation in error on a drive (`DOFA`), formed as:

```
<timestamp> DOFA <req_id> <lot_id> {D|M} [<volume_id>] <drive> <return_code>
```

- `<timestamp>`: in `YYYYMMDD-HHMNSS` format

- `DOFA`: “Drive Operation FAilure” log type
- `<req_id>`: Identifier of the request
- `<lot_id>`: identifier of the lot
- `{D|M}`: The error was encountered while dismounting (`D`) or mounting (`M`) tape
- `[<volume_id>]`: optional, identifier of the tape that was used for the operation when available
- `<drive>`: serial number of the drive used
- `<return_code>`: Arcsys Transfer Server error code, in hexadecimal

16.2. Retrieving Expired Data

This section covers the issue of retrieving data that has expired due to error.

To do this, you must:

- Have access to the Arcsys Database via SQL console (SQLPlus, SQL client, etc.)
- Know the identifier of the lot to retrieve.

16.2.1. Step 1 – Retrieving the Identifier of the Tape(s) and Associated Markers

First, you must get the identifier(s) (TAPIDENT field) of the tape(s) containing the archived data, as well as the filemarks of the tapes (MOVMEDMARKER field) and the envelope numbers (MOVIDENT field). To do this, launch the following SQL command:

```
SELECT MOVIDENT, TAPIDENT, MOVMEDMARKER
FROM MOVER_MEDIA_BKP
WHERE LOTIDENT=<Identifier of the archive> AND TAPIDENT>0 AND MOVSTATUS='A'
ORDER BY MOVIDENT ASC, TAPIDENT ASC;
```

16.2.1.1. Example

```
SELECT MOVIDENT, TAPIDENT, MOVMEDMARKER
FROM MOVER_MEDIA_BKP
WHERE LOTIDENT=3697771 AND TAPIDENT>0
ORDER BY MOVIDENT ASC, TAPIDENT ASC;

MOVIDENT | TAPIDENT | MOVMEDMARKER
1      | 4651   | 741
2      | 4651   | 742
3      | 4651   | 743
```

Note

If data is deleted several times during the life of the record or if multi-zone copies are created, the same envelope number (MOVIDENT) can appear several times because the envelope has been saved several times. It is therefore up to the operator to choose the tape numbers to use for retrieval (exclude destroyed tapes, for example). The barcode of the tapes is obtained by SQL request in the following paragraph.

16.2.2. Step 2 – Retrieving the Barcode(s)

For any of the tapes obtained (value other than TAPIDENT), the following SQL request must be run:

```
SELECT TAPBARCODE FROM TAPE WHERE TAPIDENT=<TAPIDENT identifier>;
```

16.2.2.1. Example

```
SELECT TAPBARCODE FROM TAPE WHERE TAPIDENT=4651;
TAPBARCODE |
000001L8   |
```

16.2.3. Step 3 – Launching the tapecheck_dump.sh Script For Each Tape

The **tapecheck_dump.sh** script allows to dump an entire tape, a specific filemark and/or a range of filemarks.

```
./tapecheck_dump.sh <Tape barcode> <Dump target folder> [<filemarks range> <drive serial>]
```

- To recover an entire tape:

```
<filemarks range>=all
```

- To recover one or more filemarks (the comma is used as a separator):

```
<filemarks range>=741,751
```

- To recover a range of filemarks (the underscore is used as a separator):

```
<filemarks range>=741-751
```

Note

Leave the default configuration (all filemarks) if the retrieval time and location available are not critical, or if a large number of filemarks were obtained in the first stage.

The comma can be used to separate several filemarks or range of filemarks. Space character are not allowed between filemarks.

1. Check that the target directory selected exists and that there is sufficient space to receive the data.
2. Launch the **tapecheck_dump.sh** for each tape from which you want to recover data.

Note

If several lots need to be recovered, first check which tapes and filemarks are to be recovered in order to not recover the same tapes several times.

16.2.3.1. Example

The following example shows how to dump 10 filemarks from the same tape.

Launching the script:

```
 ./tapecheck_dump.sh  000001L8 /opt/Infotel/ArchivingProduct/dump_tape_temp_folder 740,750,
752-760
```

Execution:

```
---Start dumping tape---
------------------------
Drive serial:     (autoselect)
Tape bar code:    000001L8
Filemarks:        740,750,752-760
Output directory: /opt/Infotel/ArchivingProduct/dump_tape_temp_folder
Operation queued with jobID=36
```

16.2.4. Step 4 – Extracting the Data Obtained

In the target directory, you will find the envelopes extracted for the tape(s) and filemark(s) selected:

```
/opt/Infotel/ArchivingProduct/dump_tape_temp_folder $ ll
  total 8
-rw------- 1 my_user my_group    1966 dec. 14 11:22 tape_000001L8_file_000000
-rw------- 1 my_user my_group    2099 dec. 14 11:22 tape_000001L8_file_000001_lot_255_seq_1
-rw------- 1 my_user my_group    3093 dec. 14 11:22 tape_000001L8_file_000002_lot_255_seq_2
...
-rwxrwxrwx 1 my_user my_group 1048576 dec. 14 11:22 tape_000001L8_file_000011_regroup.tar
-rwxrwxrwx 1 my_user my_group 1048576 dec. 14 11:22 tape_000001L8_file_000012_regroup.tar
```

The files containing the `_regroup.tar` suffix are the chunks resulting from archiving on a zone grouped by `Spooler`. They are TAR files containing XML manifests and envelopes containing the files.

The files containing the `_seq_1` suffix are the XML manifest; it is not normally of use.

The other files are the envelopes containing the files. They are simple TAR files.

1. If you extract one of the files with `_regroup` suffix using the command:

```
tar xvf tape_000001L8_file_000011_regroup.tar
```

   You obtain a list of files:

```
lot_263_seq_1
lot_263_seq_2
lot_264_seq_1
lot_264_seq_2
```

   The files containing the `_seq_1` suffix are the XML manifest; the other files are the envelopes containing the files.

2. If you extract one of the envelopes containing the files using the command:

```
tar xvf tape_000001L8_file_000001_lot_255_seq_2
```

   or, after extracting the `_regroup.tar` file, if you extract one of the envelopes containing the files using the command:

```
tar xvf lot_264_seq_2
```

   You obtain a list of files:

```
file0
file0.arcsys
file1
file1.arcsys
file2
file2.arcsys
```

   The data files are `file<number>` files. The `file<number>.arcsys` files contain the XML handler, which indicates the correspondence between the LOGICAL and PHYSICAL tags;

```
<LOGICAL>/c317/billing/DATA/dwh/ArchivingProduct/DFACTU/F3GF
AR_DWH_DFACTU_20100706071735_0.dat</LOGICAL>
```

```
<PHYSICAL>file0</PHYSICAL>
```

3. Simply rename the file `file0` in `/c317/billing/DATA/dwh/ArchivingProduct/DFACTU/F3GFAR_DWH_DFACTU_20100706071735_0.dat` and find the initial data file with its origin name.

4. You can check that the hash value of the original file has not been altered thanks to the `HASH_VALUE` and `HASH_FUNCTION` tags in the same `file<number>.arcsys` file:

```
<HASH_FUNCTION>MD5</HASH_FUNCTION>
<HASH_VALUE>b7c5d5c803ae2ef15b479695d090c4be</HASH_VALUE>
```

You can recalculate the MD5 hash value of the file with the **md5sum** command in Linux, for example.

Note

If there are too many files, filemarks or tapes, these manual operations can be too numerous. In this case, it would be better to write a client batch to automate these processes. Infotel can help you in either partially or completely writing this batch.

16.3. Changing the Identifier of a Tape

16.3.1. Introduction

It may be necessary for a number of different reasons to change the identifier of a tape recorded in the Arcsys Database:

- Import to another relational database,
- Error in a sequence, etc.

The procedure required to change the identifier is described below.

Note

In this procedure, you must replace:

- *`New tape identifier`* with the new identifier for the tape;
- *`Old tape identifier`* with the identifier of the previous tape;
- *`First sequence identifier`* with the first sequence identifier.

The procedure for Oracle / Linux is provided only. For other combinations, please contact Arcsys technical support.

Important

Prior to execution of this procedure, any Arcsys Transfer Server using the ArcMover Tape connected to the tape library concerned must be stopped.

16.3.2. Change in the Relational Database (Oracle Database)

You must first start the following SQL script. This script deletes the sequence on the tapes, updates the mover media (presence of lots on tapes), updates the tapes, recreates the sequence starting from the first ID desired.

```
DROP SEQUENCE TAPE_SEQ;
ALTER TABLE MOVER_MEDIA DROP CONSTRAINT FK_MOVER_MEDIA_2;
UPDATE MOVER_MEDIA SET TAPIDENT=<New tape identifier> WHERE TAPIDENT=<Old tape identifier>;
COMMIT;
UPDATE TAPE SET TAPIDENT=<New tape identifier> WHERE TAPIDENT=<Old tape identifier>;
COMMIT;
ALTER TABLE MOVER_MEDIA ADD CONSTRAINT FK_MOVER_MEDIA_2 FOREIGN KEY (TAPIDENT) REFERENCES
 TAPE(TAPIDENT);
CREATE SEQUENCE TAPE_SEQ INCREMENT BY 5 START WITH <First sequence identifier> MAXVALUE 1.
0E28 MINVALUE 1 NOCYCLE CACHE 20 NOORDER;
ALTER TRIGGER TAPE_TRIG COMPILE;
```

16.3.3. Changing the Tape Header (Linux)

The tape header contains the ID for the tape. It is read by ArcMover Tape each time a tape is mounted; the tape must be unmounted if there is no correspondence with the relational database. You must therefore modify it manually for it to contain the new ID.

Follow the procedure below which consists of a series of Unix commands that must be run on the server with access to the tape library.

1. The tape must be mounted in a drive. In this example, the `/dev/rmt/2cbn` drive is chosen.

2. Check the drive status:

   ```
   user@host:~> mt -f /dev/rmt/2cbn status
    HP Ultrium 5-SCSI tape drive: sense key(0x0)= No Additional Sense residual= 0 retries=
    0 file no= 0 block no= 0
   ```

   Here, you must ensure that the `sense key` is set to `0x0`. If there is another value, such as `0x6` (Attention Unit), this means that the status of the drive changed due to tape loading. In this case, simply restart the `status` command. If after this, the `sense key` contains a value other than `0x0`, stop the procedure and contact Arcsys technical support.

3. Rewind to the very beginning of the tape:

   ```
   user@host:~> mt -f /dev/rmt/2cbn rewind
   user@host:-> mt -f /dev/rmt/2cbn status
   HP Ultrium 5-SCSI tape drive: sense key(0x0)= No Additional Sense residual= 0 retries= 0
    file no= 0 block no= 0
   ```

   Here, ensure that the tape is still set to `file no=0` and `block no=0` , and check that the `sense key` is still `0x0`.

4. Read the header with the **dd** command:

```
user@host~> dd if=/dev/rmt/2cbn > header.txt
0+1 enregistrements entrants 0+1 enregistrements sortants
user@host:~> more header.txt
<ARCMOVER_MEDIA_HEADER><DATA><MEDIA_IDENT><Old tape identifier></
MEDIA_IDENT><CREATE_DATE>25-09-2014 09:31:12</CREATE_DATE> </DATA></
ARCMOVER_MEDIA_HEADER>
```

   The `header.txt` file displays a `MEDIA_IDENT` tag that corresponds to the Old tape ID.

5. Prepare the new header with the **sed** command:

```
user@host:~> sed -e 's/><Old tape identifier>/<New tape identifier></g' header.txt >
 header_modif.txt
# (Example: sed -e 's/>27</>31</g' header.txt >  header_modif.txt)
user@host:~> more header_modif.txt
<ARCMOVER_MEDIA_HEADER><DATA><MEDIA_IDENT> <New tape identifier></
MEDIA_IDENT><CREATE_DATE>25-09-2014 09:31:12</CREATE_DATE> </DATA></
ARCMOVER_MEDIA_HEADER>
```

   Use this command to modify the required bytes only (an editor can add an unwanted line break that will change the physical size of the header).

6. Rewind the tape (to be in position 0):

```
user@host:~> mt -f /dev/rmt/2cbn rewind
user@host:~> mt -f /dev/rmt/2cbn status
HP Ultrium 5-SCSI tape drive: sense key(0x0)= No Additional Sense residual= 0 retries= 0
 file no= 0 block no= 0
```

7. Write to the new header:

```
user@host:~> dd of=/dev/rmt/2cbn if=header_modif.txt
0+1 enregistrements entrants
0+1 enregistrements sortants
```

8. Rewind and check the new header:

```
user@host:~> mt -f /dev/rmt/2cbn rewind
user@host:~> mt -f /dev/rmt/2cbn status HP Ultrium 5-SCSI tape drive: sense key(0x0)= No
Additional Sense residual= 0 retries= 0 file no= 0 block no= 0

user@host:~> dd if=/dev/rmt/2cbn > header_new.txt
0+1 enregistrements entrants 0+1 enregistrements sortants

user@host:~> more header_new.txt
<ARCMOVER_MEDIA_HEADER><DATA><MEDIA_IDENT><New tape identifier></MEDIA_IDENT><CREATE_
DATE>25-09-2014 09:31:12</CREATE_DATE> </DATA></ARCMOVER_MEDIA_HEADER>
```

9. Rewind:

```
user@host:~> mt -f /dev/rmt/2cbn rewind
user@host:~> mt -f /dev/rmt/2cbn status HP Ultrium 5-SCSI tape drive: sense key(0x0)= No
Additional Sense residual= 0 retries= 0 file no= 0 block no= 0
```

10. Eject the tape.

16.4. Managing mailslots

The mailslots are SCSI hardware devices (the IEE, Import/Export Element) that allow to load or unload tapes into or from the tape library.

The tapes located in the mailslots are ignored by the Arcsys Transfer Server. They can not be selected and are considered as offline. Restore requests of data from tapes located in the mailslots or outside the tape library will fail.

To control the mailslots content while the Arcsys Transfer Server is running, use the **mailslot.sh** (or .bat) script located in the `scripts/arcmover` directory of the Arcsys Transfer Server installation directory.

Important

The Arcsys Transfer Server must manage the tape library where the tape will be moved.

The Arcsys Transfer Server must be launched before launching mailslot.sh.

16.4.1. Monitoring the content of the mailslots

The content of the tape library mailslots can be viewed using the script `monitor_transferServer.sh` (or .bat).

For example, the following output of **monitor_transferServer.sh** script shows that the tape `000199L4` is located in the mailslot `IEE[1]` of the tape library whose serial number is `559000300295`. On this tape library, there are four free mailslots.

```
STK SL500 - 559000300295:
  Drives:
        Drive 0 - 9310246860
                Tape    000045L3
        Drive 1 - 1310060174
                Tape    Empty
  Zone/Drive association:
        Zone 122 (Pool 50)
                Drive   9310246860
                Drive   1310060174
  Scratch pools:
        Pool 50 - 1 available media(s)
                Media type LTO-400G - 1 available media(s)
                        000001L3
  Mailslot:
        IEE[1] - 000199L4
  Slot(s) free: 4/5
```

16.4.2. Adding a tape in Arcsys

The procedure below presents the different steps to add a tape in Arcsys.

Note

At least one slot must be empty and available in the tape library to receive the tape imported from the mailslot.

1. Place one or several tapes in the tape library's mailslots.

   ### Note

   This step has to be done physically by an external operator.

2. If the tape that is inserted in the tape library is a blank one, change the tape library configuration file to specify scratch pools (See managing scratch pools chapter). This will tell the Arcsys Transfer Server which scratch pool this tape should be added to.

   This step can be omitted if the tape that will be inserted has already been used by the Arcsys Transfer Server and is already known by it. Such case can be encountered when full tapes have been set offline and physically unloaded from the tape library to be stored on-shelf. In this case, the tape can be moved back in the tape library, so that a retrieval request of a lot stored on it can be processed.

3. Launch `mailslot.sh` script, with the following parameters:

   a. `ONLINE` : to tell the script to move the tape from the mailslot to a regular slot. Once in the regular slot, the tape can be used by the Arcsys Transfer Server.

   b. the serial number of the tape library where the tape is located

   c. the name of the tape

   For example, to move the tape `00001L3` from the mailslot of the tape library with serial `559000300295`, launch:

```
$ ./mailslot.sh ONLINE 559000300295 000001L3
```

16.4.3. Removing a tape from the tape library

Note

At least one mailslot must be empty to receive the tape that will be removed from the tape library.

Once removed from the tape library, if the tape has already been used by Arcsys, it is still known in the Arcsys Database, but is set offline. To remove a tape from the Arcsys Database, see page 272, « tape_delete.sh Script ».

The tape will only be moved to the mailslot if no operation is in progress on it.

Launch **mailslot.sh** script, with the following parameters:

1. `OFFLINE` : to tell the script to unload a tape from the tape library. The tape will no longer be used by the Arcsys Transfer Server.
2. the serial number of the tape library where the tape is located
3. the name of the tape

For example, to send the tape `00001L3` to the mailslot of the tape library with serial `559000300295`, launch:

```
$ ./mailslot.sh OFFLINE 559000300295 000001L3
```

16.4.4. Managing scratch pools

In order to add new scratch tapes in Arcsys Transfer Server, the configuration file of the tape library must be changed to associate those tapes to the corresponding scratch pool.

For example, to add the tape `00199L4` to the scratch pool `94`:

```
[scratchpools]
000001L3=1
...
00199L4=94
```

Important

The modification of the configuration file must be done before running mailslot.sh script.

16.4.5. Return code and error handling

In case of success, the script returns with a return code of `0`, and displays `Success`.

Most common errors are displayed in the script output. For example, if the tape library or the tape is not found, if the Arcsys Transfer Server cannot be reached or if a request is proceeded on a tape when requesting it to be moved to the mail slot. For example :

```
./mailslot.sh OFFLINE 559000300395 000001L3
[…]
Moving offline tape '000001L3' of tape library '559000300395'...
Tape library was not found (iRc=0x019E9001).
```

When an error is encountered, the error code is displayed and the script returns the severity of the error. More details about the error can be found in the log file of the Arcsys Transfer Server.

The different values for the return code of the **mailslot.sh** script are listed below.

Return code value	Severity of the error
0	No error
4	Warning
8	Error
12	Severe
16	Fatal

16.5. Recycling a tape

When archived documents expire, they are not physically removed from tapes.

When a tape is filled mostly with documents that reached their end of life, it can be recycled. Data that is still valid on the recycled tape is moved to another tape, and the origin tape becomes available for new archives.

Note

The recycling process does not proceed a true reset:

- the tape is still affected to the same zone as before
- the process does not erase data but only writes an EOF (end-of-file) marker at the end of the Arcsys header on the tape. This choice is made to make the recycling request faster to process

The recycling operation is designed to use only one drive, so that the recycling process does not unduly slow down the normal operation of Arcsys.

Important

The recycling request differs from expiration request. After recycling a tape, this tape remains available for new archiving request on the zone. After a tape expiration, it is no longer available for any request.

16.5.1. Behavior of recycling request

One recycling request concerns all the archives that did not reach the end of their retention period and are stored on the tape. Each recycling request concerns a tape.

The recycling process moves all the envelopes that are in their retention period to another tape of the same zone and resets the initial tape which is emptied of its content.

The destination tape of a recycling request is not necessarily a scratch one. It can be any tape belonging to the same zone and having enough free space to receive all the envelopes still retained.

The recycling operation proceeds as follows:

1. Select the movers (envelops) to move from the tape to expire to an other tape of the same zone, up to the specified limits (in size or number of movers (see *recycling_sect_max_movers* and *recycling_select_max_size* parameters in page 238, « etc/arcmover/<peripheral>.conf file ») and calculate the total size of the data that will have to be moved.
2. Select a destination tape.
3. Load the origin tape in a drive.
4. Copy the data to the recycle stage directory of the Arcsys Transfer Server (see tape recycling parameters section in the Arcsys Transfer Server configuration file).
5. Unload the origin tape from the drive and load the destination one.
6. Copy the data from the recycle stage directory of the Arcsys Transfer Server to the destination tape.
7. Continue from step 1 until all the lots currently on the tape and that have not expired have been considered or until the maximum number of loops have been raised (see *recycling_select_max_loop* parameter in page 238, « etc/arcmover/<peripheral>.conf file »).
8. If all the lots that were on the tape have correctly been migrated to another tape (or other tapes) of the zone, empty the origin tape (rewind and rewrite tape header).

   If an error occurred, or if the maximum number of loops have been raised before all the lots have been moved, the recycling request ends in error status.

16.5.2. Automatic recycling

A recycling request can be created automatically by the Arcsys Transfer Server, when it detects that the threshold of recyclable space for this type of tape is reached.

The threshold can be customised in the Arcsys Web Agent, in the screen of modification of tape type. By default, it is set to 100%. This value can never be

reached, due to the presence of filemarks or backhitchs on the tape. It needs to be changed to a lower value to activate recycling requests. For more information on the tape type modification screen, see the Arcsys Web Interface User Manual. After modifying the value in the Arcsys Web Agent, the Arcsys Transfer Servers need to be restarted to take this modification into account.

As for all automatic requests, the recycling request handling need to be activated in the parameters of the Arcsys Engine. See page 284, « ENGINE.properties File » for more details.

16.5.3. Manual recycling

A recycling request for a particular tape can be created manually in the Arcsys Web Agent, in the screen of modification of the tape.

For more information on the tape modification screen, see the Arcsys Web Interface User Manual.

16.6. Deleting a drive

It may be necessary to delete a drive from the Arcsys Database, for example after a technical incident on a drive, or in the event of a hardware change. This can be done using **drive_delete.sh** script.

This script is located in the `scripts/arcmover` directory of the Arcsys Transfer Server installation directory. See page 269, « drive_delete.sh Script » for more details.

To remove a drive from the Arcsys Database proceed as follows:

1. Stop the Arcsys Transfer Server managing the drive.
2. Update the tape library configuration file to remove the drive you want to delete (see page 238, « etc/arcmover/<peripheral>.conf file »).
3. Start the Arcsys Transfer Server.
4. Ask for drive deletion by calling **drive_delete.sh** script.

16.7. Deleting a tape library

It may be necessary to delete a tape library from the Arcsys Database, for example after a hardware change. This can be done using **drive_delete.sh** and **tape_library_delete.sh** scripts.

These scripts are located in the `scripts/arcmover` directory of the Arcsys Transfer Server installation directory. See page 269, « drive_delete.sh Script » and page 273, « tape_library_delete.sh Script » for more details.

To remove a tape library from the Arcsys Database proceed as follows:

1. Preliminary checks:

    1. Check that no lot is archived on the tape library. This can be checked, for example, on the list of tapes screen of Arcsys Web Agent, by checking that no lot is listed in the "Lot(s) stored on the tape" tab for all of the tapes associated to the tape library.

    2. Check that there is no request planned on the tape library. This can be checked, for example, by using the **monitor_transferServer.sh** script with `activity` section (see page 267, « monitor_transferServer.sh (or .bat) Script »).

2. Stop the Arcsys Transfer Server managing the tape library.

3. Update the ArcMover configuration file to remove the tape library (see page 233, « etc/arcmover/arcMover.conf File »).

4. Start the Arcsys Transfer Server.

5. Ask for drives deletion by calling **drive_delete.sh** script.

6. Ask for tape library deletion by calling **tape_library_delete.sh** script.

16.8. Replacing a drive

It may be necessary to replace a drive from a tape library by a new one in the same tape library, for example after a technical incident on a drive, or in the event of a hardware change. This can be done using **drive_replace.sh** script.

This script is located in the `scripts/arcmover` directory of the Arcsys Transfer Server installation directory. See page 269, « drive_replace.sh Script » for more details.

The **drive_replace.sh** script updates the association of tape types and the association of the drive to the zones. Once the **drive_replace.sh** script is successfully ended:

- the new drive is associated to the same tape types as the previous drive was,
- the new drive is associated to the same zones as the previous drive was,
- the replaced drive is removed from Arcsys Database.

This section does not concern the change of a drive from a tape library by a drive of another tape library. The drive to replace and the one that will be used instead must be accessed by the same tape library.

To replace a drive by a new one of the same tape library, proceed as follows:

1. Stop the Arcsys Transfer Server.
2. Update the configuration file of the tape library:
    1. Add the configuration of the new drive.
    2. Remove the configuration of the drive to replace.
3. Start the Arcsys Transfer Server.
4. Switch drive associations with zones and tape types by calling **drive_replace.sh** script.
5. Review the association between the drive and the supported tape types.

    This step is necessary especially when the drive that is removed and the one used to replace it are not of similar generation.

    Changing the association between drives and supported tape types can be done in the administration interface of the Arcsys Web Agent. Any change in the supported tape types requires to stop and start the Arcsys Transfer Server to be taken into account.

16.9. Replacing a tape library

It may be necessary to replace a tape library by a new one, for example after a technical incident, or in the event of a hardware change. This can be done using **tape_library_replace.sh** script.

This script is located in the `scripts/arcmover` directory of the Arcsys Transfer Server installation directory. See page 274, « tape_library_replace.sh Script » for more details.

The **tape_library_replace.sh** script is used to associate all the drives and tapes used with the previous tape library to the new one. Once the **tape_library_replace.sh** script is successfully ended, the previous tape library is removed from Arcsys Database.

To replace a tape library by a new one, proceed as follows:

1. Stop the Arcsys Transfer Server.
2. Move the drives (if you plan to keep them in the new tape library) and tapes from the previous tape library to the new one. This action requires manual intervention and must be performed on the physical device.

    It is not required to keep the drives of the previous tape library in the new one. If you plan to entirely change both tape library and its drives, you can still use the **tape_library_replace.sh** script to update the tapes association. The script

will associate the old drives to the new tape library, and those drives, being unreachable, will be set offline. The Arcsys Transfer Server will not try to use them. You can then remove them by using **drive_delete.sh** script (see page 269, « drive_delete.sh Script » for more details).

3. Update the ArcMover configuration files:

   1. Add the new tape library to the configuration, in the `[changers]` section of the `arcMover.conf` configuration file (see page 233, « etc/arcmover/arcMover.conf File »).

   2. Add the configuration of the drives and of the scratch tapes of the new tape library (this can either be new devices, or previous ones that are physically moved to the new tape library) (see page 238, « etc/arcmover/<peripheral>.conf file »).

   3. Disable the tape library to replace (by either commenting or removing its definition) in the `[changers]` section of the `arcMover.conf` configuration file (see page 233, « etc/arcmover/arcMover.conf File »).

4. Start the Arcsys Transfer Server.

   Note

   Starting the Arcsys Transfer Server creates in the Arcsys Database new scratch tapes with the same barcode as the previous ones. This data is not consistent at this step of the procedure, and the following steps will fix this situation.

5. Switch associations between tape library and tapes and between tape library and drives by calling **tape_library_replace.sh** script.

6. If the new tape library uses new drives, you can remove the old drives by using **drive_delete.sh** script (see page 269, « drive_delete.sh Script » for more details).

17. Web

The web interface allows you to perform archiving and retrieval operations.

For more information on this component, see the Arcsys Web Interface User Manual.

17.1. Changing a Label

All labels added on the web are located in a configuration file by language/country (see page 419, « etc/WEBAGENT_label_xx_XX.properties File »).

Labels can be changed independently.

17.1.1. Operating Mode for an Example: Lot Screen

Suppose that you want to change the label of the "Lot name" column on the screen listing the lots (Section 17.1.1, “Operating Mode for an Example: Lot Screen” [101]) to the label "Name of lot".

17.1.1.1. Finding the Key to Change

In the settings file, you need to find the key corresponding to the label to be changed. In this example, you need to find the key corresponding to "Lot name". To do this, you must first find the corresponding page. The structure is as follows:

1. If the label is a general label, it is found in the first section of the file "General labels";

2. If the label is a label in a specific page, it is found in the section "Labels by Page Number" and its key starts with "PG" followed by the page number. To determine the page number, add the source code of the page (right click and then select view source code) and read the line found the first section that starts with "<!--Page name:";

   In this example, the page name is: 029-listeLot.jsp. Therefore you need to edit page number 29. After search the labels starting with PG029 for "Lot name", you will find:

   PG029_TAB_TITRE_CODE=Lot name

17.1.1.2. Editing a Key Value

To edit the value of a key:

PG029_TAB_TITRE_CODE=Nameof lot

and then restart the Arcsys Web Agent so that the change is taken into account.

17.2. Editing the Arcsys Web Agent Online Retrieval Directory

17.2.1. Online Retrieval Directory

The synchronous retrieval directory of the Arcsys Web Agent enables files to be stored during synchronous retrieval just before sending them to the web browser.

It is specified in the configuration file of the web agent (see page 397, « etc/WEBAGENT_configuration.properties File ») in the *ONLINE_FILE_REP* parameter.

Note

This directory is automatically empty. The space required varies according to the number and importance of synchronous retrievals done by users.

17.3. Configuring Lists

17.3.1. Description

On each of the pages containing a list, different filters as well as the maximum number of elements per page can be set.

These parameters are defined in the configuration files for the Arcsys Web Agent (see page 397, « etc/WEBAGENT_configuration.properties File »):

- MAX_SELECT_ENREG_xxx: Defines the maximum number of items retrieved from the relational database;
- MAX_AFFICH_ENREG_xxx: Defines the number of items displayed per page;
- AFFICHE_ZONE_SELECT_xxx and AFFICHE_ZONE_FILTRE_xxx: Determines whether filters can be displayed. Note: Lists of Lots, Objects, Controlled Values, Requests and Jobs are generated differently to other lists; for this reason, there is only the MAX_AFFICH_ENREG_xxx parameter.

17.4. Redirecting a LDAP User Group

17.4.1. Description

A redirection URL can be associated to each LDAP group. So when a user logs in, they are redirected to the appropriate URL.

These associations are done using the MAPPING_GROUP_URL and REDIRECT.URL.xxx parameters in the configuration file of the Arcsys Web Agent (see page 397, « etc/WEBAGENT_configuration.properties File »). If no URL is associated to the user's group(s), they will be redirected to the Arcsys Web home screen.

If the user belongs to several groups, the URL taken will be that of the group that appears first in the list defined by the administrator (MAPPING_GROUP_URL parameter).

17.4.2. Example

The configuration below is done to redirect a user of the group 'group2' to the Infotel website. The users of groups 'group1' and 'group3' will be redirected to the Arcsys web home screen.

```
MAPPING_GROUP_URL = group1,1|group2,url|group3,test
REDIRECT.URL.1 =
REDIRECT.URL.url = http://www.infotel.com
```

17.5. Creating a Category

17.5.1. Description

Record categories can be defined to display a warning message for all partial or full retrieval operations performed on a lot.

To create a category, you need to perform a record search and save that search. The name given to the search will be the category name.

Then, copy the file generated (`search_X.xml`) in the category folder. This folder is defined in the configuration file of the Arcsys Web Agent by the *`CATEGORIES_DIR`* parameter (see page 397, « etc/WEBAGENT_configuration.properties File »).

18. Arcsys and Business Continuity Plan: Implementation

The Arcsys Functional Description Manual describes the Arcsys functions regarding business continuity plan at a conceptual level. This section will provide more details on the configuration to be carried out in order to implement these functions.

18.1. Step 1: Configuring the Sites

18.1.1. Installing the Different Arcsys Modules

Install the different Arcsys modules with an Arcsys core (= Engine + Transfer Server + API + Application Agent) for the installation site.

Certain modules can be installed several times on the same site; see the Arcsys Functional Description Manual for more information on their different possibilities and use.

18.1.2. Engine

Each Arcsys Engine can be used for some or all of the transfer servers of a site. The *TRANSFER_SERVERS* parameter lists the set of servers reachable by an Arcsys Engine.

If different Arcsys Engines of a site are installed on the same machine, you need to ensure that the port specified in *COMPONENT_API_PORT_NUMBER* is different for each Arcsys Engine.

The specialization of the different Arcsys Engines can be specified by setting the specialization parameters of the Arcsys Engine (ending with _ENGINE).

18.1.3. Application Agents

Application agents register in the relational database when contacting the Arcsys Engine(s) sharing their site, which are specified in the *ENGINE_API_LIST* parameter of the Arcsys Application Agent configuration file. An order of priority is allocated to Arcsys Engines according to order of appearance in the list. The first Arcsys Engine contacted will be the first in the list.

If different application agents are installed on the same machine, you need to ensure that the port specified in *COMPONENT_API_PORT_NUMBER* is different for each agent.

18.1.4. API

If different APIs of a same site are installed on the same machine, you need to ensure that the port specified in *COMPONENT_API_PORT_NUMBER* is different for each API.

18.1.5. Defining the Sites

You need to define the different sites in the Arcsys Web Agent, although the first site code *PRIMARY_SITE* is already defined by default (this is the site to which all the components belong by default).

18.1.6. Defining the Site Association

The *SITE_CODE* parameter needs to be set in the configuration file of each of the following components:

- Arcsys Engine
- Arcsys RMI, TCP/IP and SOAP API
- Arcsys REST API;
- Arcsys Transfer Server
- Arcsys Web Agent

18.2. Step 2: Defining Duplicated Storage Zones

18.2.1. Configuring Zones on Several Sites

You need to define the association between the different zones and the previously created sites in the Arcsys Web Agent. To enable recovery and hence perform a replication of data, a Y copy will be defined with the first storage zone belonging to the primary site and the second storage zone to the next site.

The data stored in a collection using this storage policy will also be duplicated on both sites.

18.2.2. Defining a Y Copy on Two Sites

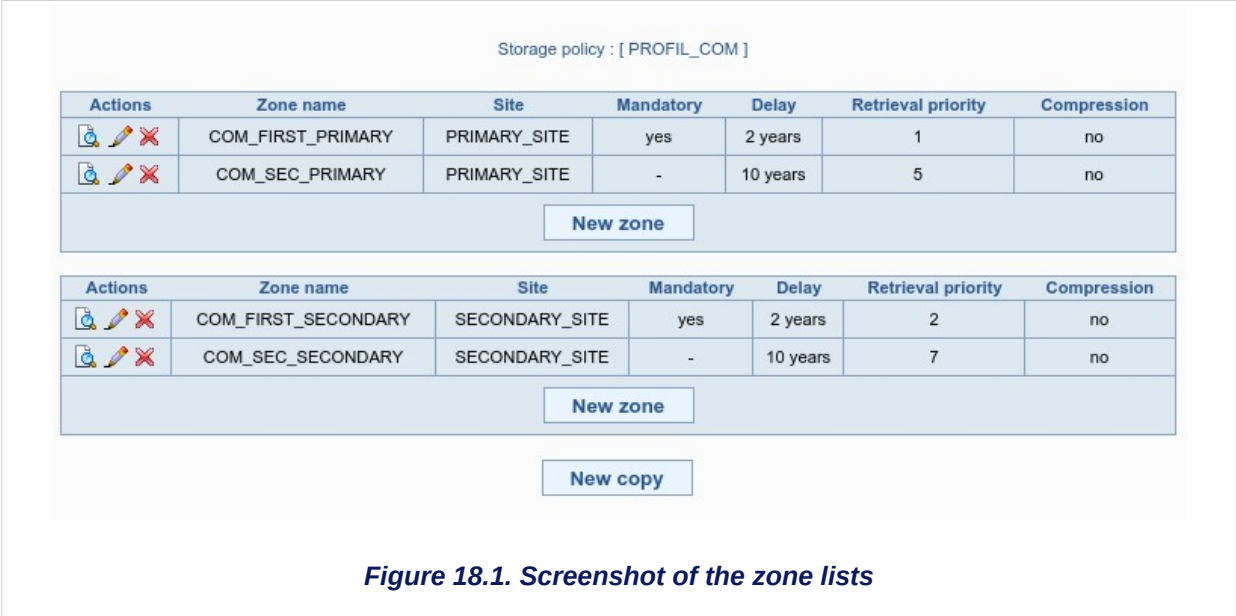


Figure 18.1. Screenshot of the zone lists

Note

When adding a new copy to an existing storage profile, use CopyRequestManager to copy existing archives over. For more information, see page 430, « CopyRequestManager ».

18.3. Step 3: Defining the Recovery Mode

18.3.1. Changing the Parameter

To enable a request to finish even if one of the sites is not operational and hence enable later recovery, set the *ALL_SITES_REQUIRED* parameter to false for all Arcsys Engines installed.

To enable a request to finish even if one of the archiving processes has not finished in one of the mandatory zones and hence enable later recovery, set the parameter *ALL_ZONES_REQUIRED* to false for all installed Arcsys Engines.

The following section contains a summary of the conditions for an archiving request to finish as ENDED following the different values of the *ALL_SITES_REQUIRED* and *ALL_ZONES_REQUIRED* parameters:

18.3.2. Summary of conditions

Request status as Ended		ALL_SITES_REQUIRED	
		true	false
ALL_ZONES_REQUIRED	true	archiving on all mandatory zones successful, for all sites	archiving on all mandatory zone of at least 1 site has succeeded
	false	at least 1 archiving on 1 zone of each site has succeeded	at least 1 archiving has succeeded on a site

Important

A recovery request is triggered *only* if the archiving request is Ended.

18.4. Backup / Recovery in Arcsys

18.4.1. Backing up

Backup in the Arcsys environment affects three different types of data. Retrieving all these data must be synchronous.

This means that the different Arcsys modules must be turned off to ensure fixity and coherence in backup.

18.4.1.1. Arcsys Modules

The Arcsys installation must be backed up. This involves the installation directories of the different modules: Arcsys Engine, Arcsys Transfer Server, etc.

In addition, the directory defined by `server_online_dir` parameter in the Arcsys Transfer Server configuration file must be backed up. Please refer to the chapter on Arcsys Transfer Server configuration for more information (see Page 200).

18.4.1.2. Relational Database

A full dump of the Arcsys Database must be done to keep all the configuration data, the existing records and also any records at the end of their retention period.

18.4.1.3. Storage File Systems

The file systems defined in Arcsys must also be backed up. To obtain a list of the file systems involved, you can check the list of those declared in the Arcsys Database,

as described by the Arcsys Web Interface User Manual (section Media Manager Administration/File Systems Management/File System List).

18.4.2. Rollback

18.4.2.1. Description

Rollback cannot be done partially: all backed up data must be restored at the same time.

Rollback involves:

- Restoring backed-up folders to their original location;
- Importing the dump in a database;
- Editing the system parameters.

The system parameters to be edited are described in the following sections.

18.4.2.2. Services File

If the file is used (not mandatory from version 5.0 of Arcsys), you must ensure that the `services` file contains a row for the Arcsys Transfer Server and a row for the API.

This `services` file is found in the directory:

- <Windows installation directory>`/system32/drivers/etc` in Windows,

`/etc` in other systems.

19. File format identification

This part describes the different steps to identify and validate archived file formats in Arcsys, using the ArcFF format control module.

Format identification is a process that extracts the detailed characteristics of a file format (format name, format version, mime type, ...).

Format validation is a process that checks if a file is well-formed and valid according to its format.

According to JHOVE documentation:

- A file is **well-formed** if it meets the purely syntactic requirements for its format.
- A file is **valid** if it is well-formed and it meets the higher-level semantic requirements for format validity.

For more details, you can refer to http://www.jhove.sourceforge.net/.

Note

The information relating to the identification and validation of the format of a document is visible in the Arcsys web GUI, on the descriptive page of the document in question (accessible after an Arcsys search).

19.1. Format policies

You need first to determine the format policies that will be usable by the different archiving processes.

19.1.1. Definition

A format policy is an XML file used to define a set of rules concerning format checks for a given type of file. These rules are used to specify the action that will be performed, the expected results of these actions, as well as the error cases triggering an archiving failure.

Important

For a format policy to be applied to a record, it must be associated with the record class.

19.1.2. Description File for a Format Policy

The description file for a format policy is an XML file that complies with the following structure:

```
<?xml version="1.0"?>
<FORMAT_POLICY>
    <CODE>FORMAT_POLICY_SAMPLE</CODE>
    <LABEL>FORMAT_POLICY_SAMPLE</LABEL>
    <CREATOR>USER</CREATOR>
    <FORMAT_RULES>
        <FORMAT_RULE>
            <TYPE>default</TYPE>
            <MAX_FILE_SIZE>100000000</MAX_FILE_SIZE> <!-- In bytes -->
            <ACTIONS>
                <ACTION>IDENTIFICATION</ACTION>
            </ACTIONS>
            <FAILURE_CONDITIONS>
                <FAILURE_CONDITION>MAXFILESIZEREACHED</FAILURE_CONDITION>
                <FAILURE_CONDITION>NOTCHARACTERIZED</FAILURE_CONDITION>
            </FAILURE_CONDITIONS>
        </FORMAT_RULE>
        <FORMAT_RULE>
            <TYPE>fileNameMatch</TYPE>
            <PATTERN>(.*).pdf</PATTERN>
            <MAX_FILE_SIZE>100000000</MAX_FILE_SIZE>
            <ACTIONS>
                <ACTION>IDENTIFICATION</ACTION>
                <ACTION>VALIDATION</ACTION>
                <ACTION>SIGNATUREVALIDATION</ACTION>
            </ACTIONS>
            <FAILURE_CONDITIONS>
                <FAILURE_CONDITION>MAXFILESIZEREACHED</FAILURE_CONDITION>
                <FAILURE_CONDITION>NOTCHARACTERIZED</FAILURE_CONDITION>
                <FAILURE_CONDITION>NOTVALID</FAILURE_CONDITION>
                <FAILURE_CONDITION>NOTWELLFORMED</FAILURE_CONDITION>
                <FAILURE_CONDITION>SIGNATURENOTVALID</FAILURE_CONDITION>
            </FAILURE_CONDITIONS>
            <ALLOWED_FORMATS>
                <ALLOWED_FORMAT>application/pdf</ALLOWED_FORMAT>
            </ALLOWED_FORMATS>
        </FORMAT_RULE>
    </FORMAT_RULES>
</FORMAT_POLICY>
```

The following must be specified for each format policy:

- A code corresponding to the Arcsys code for the format policy and which must be unique for a given Arcsys repository. A label and a user can also be specified and will be inserted in the Arcsys Database, but they are optional.
- One or more rules on format control.

The following must be specified for each rule:

- Its type: To define a rule as active for the files whose path corresponds to a certain Java regular expression, its value is `fileNameMatch`. To define a rule as active for the files that do not correspond to any `fileNameMatch` type rule, its value is `default`.

- A Java regular expression (only for `fileNameMatch`) type rules. The rule will be active for all files whose path corresponds to this regular expression.

Important

If the path of a file to be archived does not correspond to any Java regular expression of the different format rules, and if there is no format rule with `default` type, the file will not be controlled by the module.

- A maximum file size (optional) in bytes.
- Actions (optional) to be performed for this file type. An action can have a `identification`, `validation` or `signatureValidation` value (case non-sensitive). Identification action is required for validation and/or signature validation. The possible combinations concerning the actions are therefore:
  - `identification`
  - `identification`, `validation`
  - `identification`, `signatureValidation`
  - `identification`, `validation`, `signatureValidation`
- A list of error cases (optional) that will trigger an archiving error. Possible error cases are (case non-sensitive):

Value	Description
ALWAYS	Archiving will end in error for all files of this type.
NOTCHARACTERIZED	Archiving will end in error for files that could not be correctly identified by the format control module.
NOTWELLFORMED	Archiving will end in error for files whose validation process has failed or that are not well-formed.
NOTVALID	Archiving will end in error for files whose validation process has failed or that are not valid.
SIGNATURENOTVALID	Archiving will end in error for files whose signature validation process has failed or whose signature is not valid.
MAXFILESIZEREACHED	Archiving will end in error for files whose size is higher than the size specified in the *`MAX_FILE_SIZE`* parameter.
NEVER	Archiving will never end in error for a format processing problem.

Important

The use of the `NEVER` error condition will ensure that even in case of technical error that interrupts the process of format identification

and validation (ArcFF crash, internal ArcFF framework error, ...), the archiving process will continue. However, no information about format identification and validation will be retrieved and stored in the database.

If you don't want to use any of the error conditions above but still want the archiving process to end in error if a technical error occurs during the format identification and validation, simply remove the `FAILURE_CONDITIONS` block.

Note

If the *`MAX_FILE_SIZE`* parameter is set but the *`MAXFILESIZEREACHED`* failure condition is not present, archiving requests containing files whose size exceeds the value set in *`MAX_FILE_SIZE`* will not end in error. ArcFF format control module will simply not be called for these files.

- A list of MIME types (optional) expected as a result of the identification process. These MIME types will be sent as a request parameter to the format control module and will be used to define the tools to be used by the latter and to check that the MIME type obtained after identification is that expected for this type of file.

Note

In the case of identification with DROID, if you want to determine which MIME types should be expected for a particular file extension, you can refer to the PRONOM format registry https://www.nationalarchives.gov.uk/PRONOM/Format/proFormatSearch.aspx?status=new. This allows, among other things, searching for formats by extension and viewing their characteristics (like their MIME types).

19.1.3. Default Format Policies

Default format policies are supplied in the `formats/policies` directory of the Arcsys Engine. These policies can be imported in the repositories with the **formatpolicy.sh(.bat)** command. These are the following policies:

- **Standard**: Standard policy that identifies and validates the PDF, AIFF, WAV, JPEG, JPEG2000, GIF, TIF, XML, HTML formats. If a file has one of the possible extensions for these formats and is not identified or validated correctly, the archiving shows an error.

- **Extended**: Extended policy that can identify all files and validates only the Standard policy formats. If a file is not identified or validated, archiving returns an error.

- **PDF**: Policy that identifies and validates the PDF format only. If a file has a .pdf extension and is not identified or validated, archiving returns an error.
- **PDF_signature_validation**: Same principle as the `PDF` policy, but with the addition of signature validation and archiving failure in case of invalid signature.

19.1.4. Importing a format policy

In order to be accessible from Arcsys, format policies must be imported using the `formatpolicy` script delivered in the Arcsys Engine. The usage of this script is detailed page 354, « formatpolicy.sh (or .bat) Script ».

19.1.5. Attaching the format policy to an Arcsys class

The only way to use a format policy for an archiving process is to attach this format policy to a class, and to use this class for the archiving process (For the class creation, see Interface Guide).

To attach a format policy to a class, you must open the `class edition` menu. Then you can select the desired format policy via the field `Format policy`.

Note

If you have a single format policy for a given collection, you can use the field `Default class for the new lots` in the collection edition menu, that will avoid selecting the desired class for each lot.

19.2. File format server

19.2.1. Introduction

The next step is to configure the ArcFF format control module which is a set of web services that Arcsys contacts to identify and validate file formats.

Note

If you have installed Arcsys before the 5.0 version, you must explicitly install this module (see Arcsys Installation Manual).

19.2.2. Main configuration

The ArcFF format control module main configuration is carried out via the file `etc/arcff.properties`

The only mandatory settings are for OAuth2 authentication and possibly for SSL connection if you use SSL security protocol.

The different parameters of this file are described page 434, « etc/arcff.properties File ».

19.2.3. Tools configuration

The configuration of ArcFF format control module external tools (droid, jhove) is detailed page 438, « Tool Settings ».

Note

A default configuration of the external tools is delivered in the module.

19.2.4. Validation of PDF/A files

The validation of PDF/A files can be done exclusively with the **VeraPDF** tool.

Note

VeraPDF cannot identify file formats and shall be only used for format validation.

To validate a file of type PDF/A, it is necessary that this file has been previously identified by **DROID** because JHOVE is not able to identify PDF/A files.

ArcFF must then be configured so that VeraPDF is used to validate files identified as PDF/A files (which is the default case in Arcsys 5.2.0.0). This configuration is done in the `arcff_tool_assignments.xml` file (see page 439, « etc/arcff_tool_assignments.xml File », in which each extended mime type (`application/pdf; version=xxx`) must be associated to a VeraPDF validation.

For example, with the configuration below, all PDF files identified by DROID as version **1a** or **1b** will be validated by VeraPDF. All the other files will be validated by JHOVE.

```
<?xml version="1.0"?>
<ASSIGNMENTS>
    <ASSIGNMENT>
        <TYPE>DEFAULT</TYPE>
        <PROCESSES>
            <PROCESS>
                <PROCESS_TYPE>IDENTIFICATION</PROCESS_TYPE>
                <TOOL_CODE>DROID</TOOL_CODE>
            </PROCESS>
            <PROCESS>
                <PROCESS_TYPE>VALIDATION</PROCESS_TYPE>
```

```
                <TOOL_CODE>JHOVE</TOOL_CODE>
            </PROCESS>
        </PROCESSES>
    </ASSIGNMENT>
    <ASSIGNMENT>
        <TYPE>MIME_TYPE</TYPE>
        <MIME_TYPE>application/pdf; version=1a</MIME_TYPE>
        <PROCESSES>
            <PROCESS>
                <PROCESS_TYPE>VALIDATION</PROCESS_TYPE>
                <TOOL_CODE>VERAPDF</TOOL_CODE>
            </PROCESS>
        </PROCESSES>
    </ASSIGNMENT>
    <ASSIGNMENT>
        <TYPE>MIME_TYPE</TYPE>
        <MIME_TYPE>application/pdf; version=1b</MIME_TYPE>
        <PROCESSES>
            <PROCESS>
                <PROCESS_TYPE>VALIDATION</PROCESS_TYPE>
                <TOOL_CODE>VERAPDF</TOOL_CODE>
            </PROCESS>
        </PROCESSES>
    </ASSIGNMENT>
</ASSIGNMENTS>
```

19.3. Arcsys Application Agent

19.3.1. Introduction

The final step is to configure the Arcsys Application Agent to make it able to contact the ArcFF format control module. This configuration is done via the file `APPAGENT.properties` (see page 362, « etc/APPAGENT.properties File »).

19.4. Handled formats

19.4.1. Introduction

The formats supported by ArcFF format control module for identification and validation are chosen from among the standard preservation file formats that are most likely to be accessible in the future. Other formats may work, but their support cannot be guaranteed in future versions, so they should be avoided.

References:

- http://www.nationalarchives.gov.uk/PRONOM/
- http://jhove.openpreservation.org/modules/

19.4.2. PDF files

Format	Identified by DROID	Validated by JHOVE
Acrobat PDF 1.0	Yes	Yes
Acrobat PDF 1.1	Yes	Yes
Acrobat PDF 1.2	Yes	Yes
Acrobat PDF 1.3	Yes	Yes
Acrobat PDF 1.4	Yes	Yes
Acrobat PDF 1.5	Yes	Yes
Acrobat PDF 1.6	Yes	Yes
Acrobat PDF 1.7	Yes	Yes
Acrobat PDF/A 1a	Yes	Yes
Acrobat PDF/A 1b	Yes	Yes
Acrobat PDF/A 2a	Yes	No
Acrobat PDF/A 2b	Yes	No
Acrobat PDF/A 2u	Yes	No
Acrobat PDF/A 3a	Yes	No
Acrobat PDF/A 3b	Yes	No
Acrobat PDF/A 3u	Yes	No
Acrobat PDF/E	Yes	No
Acrobat PDF/X-1	Yes	Yes
Acrobat PDF/X-1a	Yes	Yes
Acrobat PDF/X-2	Yes	Yes
Acrobat PDF/X-3	Yes	Yes
Acrobat PDF/X-4	Yes	No
Acrobat PDF/X-5	Yes	No
Adobe Illustrator 9.0	Yes	No
Adobe Illustrator 10.0	Yes	No
Adobe Illustrator 11.0	Yes	No
Adobe Illustrator 12.0	Yes	No
Adobe Illustrator 13.0	Yes	No
Adobe Illustrator 14.0	Yes	No
Adobe Illustrator 15.0	Yes	No
Adobe Illustrator 16.0	Yes	No

19.4.3. TIFF files

Format	Identified by DROID	Validated by JHOVE
Tagged Image File Format 4.0	Yes	Yes
Tagged Image File Format 5.0	Yes	Yes
Tagged Image File Format 6.0	Yes	Yes
Digital Negative Format	Yes	Yes
Tagged Image File Format for Image Technology	Yes	Yes
Tagged Image File Format for Electronic Photography	Yes	Yes
Geographic Tagged Image File Format	Yes	Yes
Tagged Image File Format for Internet Fax	Yes	Yes

19.4.4. JPEG files

Format	Identified by DROID	Validated by JHOVE
JPEG File Interchange Format 1.00	Yes	Yes
JPEG File Interchange Format 1.01	Yes	Yes
JPEG File Interchange Format 1.02	Yes	Yes
Exchangeable Image File Format 2.0	Yes	Yes
Exchangeable Image File Format 2.1	Yes	Yes
Exchangeable Image File Format 2.2	Yes	Yes
Exchangeable Image File Format 2.2.1	Yes	No
JPEG Tiled Image Pyramid	Yes	Yes
JPEG-LS	Yes	Yes
JP2 (JPEG 2000 part 1)	Yes	Yes
JPX (JPEG 2000 part 2)	Yes	Yes
Raw JPEG Stream	Yes	Yes

19.4.5. PNG files

Format	Identified by DROID	Validated by JHOVE
Portable Network Graphics 1.0	Yes	Yes
Portable Network Graphics 1.1	Yes	Yes
Portable Network Graphics 1.2	Yes	Yes
Animated Portable Network Graphics	Yes	No

19.4.6. Microsoft Office/LibreOffice files

Currently, none of the tools used by the ArcFF format control module are able to validate Microsoft Office/LibreOffice files. However, DROID is able to identify Microsoft Office/LibreOffice formats present in the PRONOM registry (http://www.nationalarchives.gov.uk/PRONOM/).

20. Preprocessings

Preprocessing allows operations to be performed on Arcsys objects during archiving, prior to envelopes formation. Currently, only the execution of an external script, which can take various Arcsys entity-related information as parameters, is possible.

20.1. Configuration

20.1.1. Overview

The configuration of any preprocessing is done via format policies, in the dedicated XML files (see page 109, « Format policies »). However, the execution of these preprocessings is carried out by the Arcsys Application Agent and is entirely independent of the ArcFF format control module. So, it is not necessary to configure or start ArcFF format control module for the execution of preprocessings.

20.1.2. Description

As explained earlier, preprocessings are defined in the XML files that describe format policies. A root tag `PRE_PROCESSINGS` can thus be defined at the same level as the `FORMAT_RULES` tag:

```
<?xml version="1.0"?>
<FORMAT_POLICY>
    <CODE>Advanced</CODE>
    <LABEL>Advanced policy with backup and antivirus pre-processings</LABEL>
    <CREATOR>USER</CREATOR>
    <PRE_PROCESSINGS>
        <PRE_PROCESSING>
            <TYPE>default</TYPE>
            <COMMAND>
                <PATH>/script/path/backup.sh</PATH>
                <ARGS>
                    <ARG>%object_path%</ARG>
                    <ARG>/home/backup/</ARG>
                    <ARG>%repository_code%</ARG>
                    <ARG>%collection_code%</ARG>
                    <ARG>%lot_id%</ARG>
                    <ARG>%object_id%</ARG>
                </ARGS>
                <TIMEOUT>300</TIMEOUT>
            </COMMAND>
            <FAILURE_CONDITIONS>
                 <FAILURE_CONDITION>NONZERORETURNCODE</FAILURE_CONDITION>
            </FAILURE_CONDITIONS>
        </PRE_PROCESSING>
        <PRE_PROCESSING enforce-integrity-check="true">
            <TYPE>fileNameMatch</TYPE>
            <PATTERN>(.*)\.(?i)(zip)</PATTERN>
            <COMMAND>
                <PATH>/script/path/antivirus.sh</PATH>
                <ARGS>
```

```
                    <ARG>%object_path%</ARG>
                </ARGS>
                <TIMEOUT>3600</TIMEOUT>
            </COMMAND>
            <FAILURE_CONDITIONS>
                <FAILURE_CONDITION>NONZERORETURNCODE</FAILURE_CONDITION>
                <FAILURE_CONDITION>TIMEOUTREACHED</FAILURE_CONDITION>
            </FAILURE_CONDITIONS>
        </PRE_PROCESSING>
    </PRE_PROCESSINGS>
    <FORMAT_RULES>
        ...
    </FORMAT_RULES>
</FORMAT_POLICY>
```

The following information must be specified for each preprocessing:

- Its type: As with format rules, to define a preprocessing as active for the files whose path corresponds to a certain Java regular expression, its value is `fileNameMatch`. To define a preprocessing as active for the files that do not correspond to any `fileNameMatch` type preprocessing, its value is `default`.
- A Java regular expression (only for `fileNameMatch`) type preprocessing: The preprocessing will be active for all files whose path corresponds to this regular expression.

Important

If the path of a file to be archived does not correspond to any Java regular expression of the different preprocessings, and if there is no preprocessing with `default` type, no preprocessing will be performed.

- A command (optional): It is currently the only type of preprocessing that can be defined.

The following must be specified for each command:

- A path to the external script to be executed.

Important

Since this script is executed by Arcsys Application Agent, the module must have access to it via the specified path and have the necessary permissions for this execution.

- A set of arguments to transmit to the script (optional). In these arguments, a set of variables related to the processed object can be used. These variables should be specified between '%' characters and are as follows (case-insensitive):
  - `repository_id`: The identifier of the repository where the object will be archived.

- `repository_code`: The code of the repository where the object will be archived.
- `collection_id`: The identifier of the collection where the object will be archived.
- `collection_code`: The code of the collection where the object will be archived.
- `lot_id`: The identifier of the lot containing the object.
- `lot_code`: The code of the lot containing the object.
- `object_id`: The object identifier.
- `object_path`: The object file path.

- A timeout in seconds (optional). The behavior of this timeout will depend on the definition of failure conditions described below.
- A list of failure conditions (optional) that will trigger an archiving error. Possible error cases are (case-insensitive):

Value	Description
NONZERORETURNCODE	Archiving will fail for files on which a preprocessing script has been executed and returned a non-zero exit code.
TIMEOUTREACHED	Archiving will fail for files on which a preprocessing script was executed but did not complete before the time specified by the `TIMEOUT` element.

Important

Return code `999` is used internally to indicate that the script has reached the specified timeout and should therefore not be used by external scripts.

Note

The return code of the executed script is logged in the Arcsys Application Agent logs at the INFO level, while their output is logged at the DEBUG level.

If the failure condition `NONZERORETURNCODE` is not specified for a command, but the return code of the executed script is non-zero, this will not cause the archiving to fail; a warning message will simply be logged in the Arcsys Application Agent logs.

If the error condition `TIMEOUTREACHED` is not specified for a command, but a `TIMEOUT` element is specified, if the script's execution time reaches this timeout, the archiving will not fail. Instead, the script's execution will be

interrupted, and a warning message will simply be logged in the Arcsys Application Agent logs.

If no failure condition is specified, execution of the preprocessing script will not block archiving, regardless of its execution time or return code.

20.1.3. Fixity check

If the checksum information for an object is provided at the time of archiving, the Arcsys Application Agent performs a standard check to ensure that the checksum matches the data being archived. This check occurs after the preprocessing script has been executed.

However, if the checksum information is not provided and must be calculated by the Arcsys Application Agent during archiving, you can enforce an initial checksum calculation before running the scripts. This ensures that the data has not been altered after the scripts are executed. To enable this, set the `enforce-integrity-check` attribute to `true` on the relevant `PRE_PROCESSING` element.

21. Configuring JDBC Access to the Relational Database

21.1. Description

For the Arcsys Engine, the Arcsys RMI, TCP/IP and SOAP API and the Arcsys REST API, a `REFERENTIAL_JDBC.properties` file enables JDBC connection to the Arcsys Database to be configured.

Since version 5.0, the connection pool used is HikariCP. https://brettwooldridge.github.io/HikariCP/

21.2. Parameter description

21.2.1. General parameters

Parameter name	**arcsys.typebase**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Type of database used for the Arcsys Database. Possible values: `MariaDB` / `Oracle` / `SQLServer` / `PostgreSQL`. Note: Values are case-sensitive.
Shipped value	`Oracle`

21.2.2. Configuring the pool

Parameter name	**arcsys.jdbc_url**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	JDBC access URL to the relational database. For Oracle Database with the format `jdbc:oracle:thin:@[HOST]:[PORT]:[DATABASE]` (or `jdbc:oracle:thin:[USER/PASSWORD]@//[HOST][:PORT]/SERVICE` with Oracle 12c); for MariaDB, with the format `jdbc:mariadb://[HOST]:[PORT]:[DATABASE]` ; for Microsoft SQL Server, with the format `jdbc:jtds:sqlserver://[HOST]:[PORT]:[DATABASE];instance=[INSTANCE_NAME]` ; for PostgreSQL, with the format `jdbc:postgresql://[HOST]:[PORT]/[DATABASE]`. **Caution** For MariaDB, you must specify the `serverTimezone` explicitly in the connection string: for example `jdbc:mariadb://myhost.mydomain.com:3306/arc_pcl027?serverTimezone=UTC`
Shipped value	`jdbc:oracle:thin:@localhost:1521:ARCSYS`

Parameter name	**arcsys.jdbc_user**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	User name for connecting to the relational database.
Shipped value	`ARCSYS`

Parameter name	**arcsys.jdbc_password**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	User password for connecting to the relational database.
Shipped value	No shipped value.

Parameter name	**arcsys.jdbc_maximumPoolSize**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Maximum number of connections that this pool is authorized to reach, including the connections available and in use. This value determines the maximum number of real connections to the backend of the database. An appropriate value depends on the runtime environment. When the pool reaches this size and no idle connections are available, connection attempts will be blocked until *`arcsys.jdbc_connectionTimeout`* milliseconds.
Shipped value	`30`

Parameter name	**arcsys.jdbc_connectionTimeout**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	This property controls the maximum number of milliseconds for which a client will wait for a connection from the pool. If this time period is exceeded without a connection becoming available, an error is triggered. The lowest acceptable connection timeout is 250 ms.
Shipped value	`30000`

Parameter name	**arcsys.jdbc_validationTimeout**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	This property controls the maximum time period during which a connection will be tested. This value must be less than *`arcsys.jdbc_connectionTimeout`*. The lowest acceptable connection timeout is 250 ms.
Shipped value	`5000`

Parameter name	**arcsys.jdbc_idleTimeout**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	This property controls the maximum time period during which a connection is authorized to remain inactive in the pool. This parameter applies only when *`arcsys.jdbc_minimumIdle`* is defined as less than *`arcsys.jdbc_maximumPoolSize`*. For a connection to be removed as inactive or not is subject to a maximum variation of 30 seconds, and an average of 15 seconds. A connection will never be removed as inactive before this timeout. A value of 0 means that the inactive connections are never deleted from the pool. The minimum value authorized is 10000 ms (10 seconds).
Shipped value	`30000`

Parameter name	**arcsys.jdbc_maxLifetime**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	This property controls the maximum duration of a connection in the pool. When a connection reaches this timeout, it will be removed from the pool, subject to a maximum variation of +30 seconds. A connection in use will never be removed. It is highly recommended you define this value and it must be at least 30 seconds less than all connection timeouts defined at the level of the database. A value of 0 specifies that there is no maximum timeout (unlimited time period), subject of course to the *`arcsys.jdbc_idleTimeout`*parameter. When no value is specified, the default value of 1800000 (30 minutes) will be used.
Shipped value	empty (therefore equivalent to `1800000`)

Parameter name	**arcsys.jdbc_minimumIdle**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	This property controls the minimum number of inactive connections that HikariCP tries to maintain in the pool. If the inactive connections fall below this value, HikariCP makes an effort to add additional connections quickly and efficiently. However, for high performance and maximum reactivity to peak requests, it is recommended you set this value to the same value as *`arcsys.jdbc_maximumPoolSize`*.
Shipped value	`10`

Parameter name	**arcsys.jdbc_connectionTestQuery**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	If your JDBC driver manages JDBC4, it is highly recommended you do not configure this property. This is required only for databases that do not handle the JDBC4 API `Connection.isValid()`. This is the request that is executed just before a connection is given via the pool to validate that the connection to the database is still active.
Shipped value	space

Parameter name	**arcsys.jdbc_aliveBypassWindow**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	This parameter corresponds to the property `com.zaxxer.hikari.aliveBypassWindow`. It is reserved for advanced use.
Shipped value	`300000`

Parameter name	**arcsys.jdbc_housekeepingPeriod**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	This parameter corresponds to the property `com.zaxxer.hikari.housekeeping.periodMs`. It is reserved for advanced use.
Shipped value	`300000`

Parameter name	**SIGNATURE_PROFILES_FILENAME**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	This parameter is only used on the API when using the external signature check. In this context, this parameter specifies the XML definition file of the signature profile that will be used. This file must comply with the corresponding XSD supplied in the XML directory of the Arcsys API.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcSign Option

22. Managing Files in Native Format

The native archiving consists in injecting a native file (whether AFP, ZIP or LZMA file) directly into Arcsys. This file is considered as a native envelope. Its fingerprint value is preserved and can be verified.

The native ZIP or LZMA file is a particular object, on which the retrieval is not possible (because it does not make sense to retrieve such a file). Retrieving a native ZIP/LZMA file extracts the documents contained in it. By default, if no folder is specified for the retrieving operation, the documents are retrieved in the directory specified by the *NATIVE_DEFAULT_DIRECTORY* parameter (see Page 373 for more details).

To handle native objects, you must include a format handler in Arcsys that describes how Arcsys interprets each type of native envelope.

Note

The injection of native files is available in the administration interface of the Arcsys Web Agent or in the Arcsys API (RMI, SOAP or REST) and by optional external modules using the Arcsys API (RMI, SOAP or REST) (like ArcIP, for example).

22.1. Format Handler

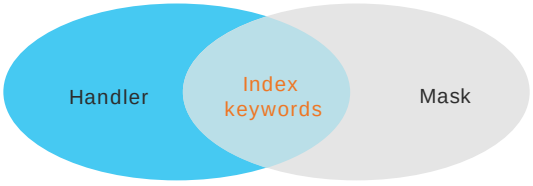
The format handler enables the documents to be analyzed and automatically indexed during the archiving stage. It must contain all information useful to Arcsys to interpret a file of a given type. It is in XML format and is defined by a DTD contained in the `xml` folder of the Arcsys Engine.

22.2. Configuration

22.2.1. Mapping the Handler with the Mask

To be able to index the documents, the keywords found in the handler must match those found in the mask used.

More specifically, the keywords that are indexed are those found in the mask and in the format handler.



Mandatory keywords of the mask must be used in the handler, otherwise an error will occur.

Important

If no keyword is mandatory at mask level and the intersection between the mask and the handler is blank (e.g. the handler given does not match the file), the file will be archived but not indexed.

22.2.2. Configuring Arcsys Engine

The storage directory of the handler files is set in the `ENGINE.properties` file in the *`FORMAT_FILES_PATH`* parameter.

The name of the "Format" field in the object creation interface matches the name of the handler file (see Arcsys Web Interface User Manual Managing Objects section).

22.2.3. Configuring Arcsys Application Agent

A default retrieval directory for the agent is set in the *`NATIVE_DEFAULT_DIRECTORY`* parameter in the `APPAGENT.properties` file. It is used to retrieve the files from the record in native format if a destination directory is not specified.

22.2.4. Configuring the Arcsys Web Agent

Support of native envelopes at interface level can be enabled or disabled using the *`ENABLE_NATIVE_MOVER`* parameter. This option is disabled by default and can be enabled by setting the parameter as true.

22.3. Verification of the hash value

The verification of the hash value that you may specify for the native object involves both *`MANIFEST_TAR_HASH_FUNCTION`* parameter from the Arcsys Engine configuration file and *`VERIFY_NATIVE_REFERENTIAL_HASH_VALUE`* parameter from the Arcsys Application Agent configuration file.

When you specify a hash function and a hash value for the native (AFP or ZIP) object:

- if the hash function that you specify differs from *`MANIFEST_TAR_HASH_FUNCTION`*: Arcsys computes the hash value with *`MANIFEST_TAR_HASH_FUNCTION`*. The hash value that Arcsys computed overwrites the hash value that you specified for the native object, which is ignored.
- if the hash function that you specify is the same that the one set in *`MANIFEST_TAR_HASH_FUNCTION`* parameter:

- if *VERIFY_NATIVE_REFERENTIAL_HASH_VALUE*=`true`, Arcsys raises an error if the hash value computed differs from the one you specified
- if *VERIFY_NATIVE_REFERENTIAL_HASH_VALUE*=`false`, Arcsys Application Agent does not raise an error if the hash value computed differs from the one you specified, but Arcsys Transfer Server does.

22.4. Managing AFP Files (ArcAFP Option)

This solutions enables AFP file support. These files contain sub-documents that can eventually have metadata. For this file type, a format handler is required to analyze the file and extract the metadata.

AFP files can be structured in two different ways, either as a NOP-type structure or a TLE structure, depending on the software with which they were generated.

The format handler enables the documents to be analyzed and automatically indexed during the archiving stage.

Note

Only self-sufficient AFP files are supported (the set of resources must be found in the file, i.e. external resources are not supported).

22.4.1. NOP-structured AFP File

22.4.1.1. Format Handler

A NOP is a purely informative hexadecimal tag that does not appear in the useful data of the documents. The user must define the structure of the NOP to be considered in the AFP files through the format handler.

22.4.1.2. XML File Structure

The format handlers are XML files. These files are stored in the `FORMAT_FILES_PATH/afp/nop/` directory. *FORMAT_FILES_PATH* parameter is set in the Arcsys Engine parameter file (for more details, see page 284, « ENGINE.properties File » file description).

The format handlers must match the NOP XSD that can be found in `xml/afp/nop/` directory of Arcsys Engine installation directory.

For NOP structuring, the structure must be the following:

```
<descriptor encoding="Cp1047">
  <required>
    <type>Hexadecimal standard tag (X'D3EEEE')</type>
    <data name="Tag" offset="Numerical value">Comparison characters</data>
    <data name="Tag" offset="Numerical value">Comparison characters</data>
    <data offset="Numerical value">Comparison characters</data>
  </required>
  <metadata>
    <data name="keyword" offset="Numerical value" length="Number of characters" />
    <data name="keyword" offset="Numerical value" length="Number of
 characters" format="Decoding format"/>
</metadata>
</descriptor>
```

This file contains two parts: the mandatory values and the metadata. The mandatory values are the elements that enable the NOP type to be identified. The metadata provide the indexing information.

22.4.1.3. Mandatory Elements

1. type: The hexadecimal tag, defined by the AFP standard, is always `D3EEEE`. This code identifies a NOP type element.

2. data: These are the additional tags that differentiate the NOP. As these tags do not belong to the AFP standard, their values (offset) must be indicated in bytes. They are found in the information area, always starting with the offset 9.

22.4.1.4. Metadata Elements

The metadata elements are found in the "information" area of the NOP. To pinpoint them, Arcsys is based on a source offset starting at least with the value 9 and on a length in number of characters. According to the search type, a deciphering format will be used as, for example, for the date where the year, month and day information must be determined.

22.4.1.5. Example

For the following XML file:

```
<descriptor encoding="Cp1047">
  <required>
      <type>d3eeee</type>
      <data name="constant" offset="9">ARC</data>
      <data name="NOP version" offset="12">007</data>
      <data name="object type code" offset="15">002</data>
       <data offset="36">MLG</data>
  </required>
  <metadata>
    <data name="ContractNumber" offset="18" length="6" />
    <data name="IssueDate" offset="23" length="8" format="yyyyMMdd"/>
</metadata>
</descriptor>
```

This NOP must be indicated with the standard D3EEEE tag. With header structure defined, you can pinpoint the position of the first byte of the NOP structure and hence analyze the additional tags.

At offset 9, there is a three-letter succession: the constant "ARC".

At offset 12, there is a three-letter succession: the NOP version "007".

At offset 15, there is a three-letter succession: the object type code "002".

At offset 36, there is a three-letter succession: "MLG".

If all these conditions are valid, the associated metadata can be recovered.

At offset 18, 6 characters should be retrieved corresponding to the "Contract Number".

At offset 23, 8 characters should be retrieved corresponding to the "Issue Date". For a date type, the format attribute enables you to decipher the information: "yyyy" for the year, "MM" for the month and "dd" for the day.

22.4.2. AFP File Structured by TLE

22.4.2.1. Format Handler

A TLE is a hexadecimal tag used to define the logical structure of an AFP flow. An AFP flow document is identified by the appearance of TLE. A TLE contains an identifier and a value. This indexing mechanism will be mapped with that of Arcsys. The TLE identifier matches the identifier of an Arcsys metadata item and the value of the TLE matches the metadata value. The format handler must therefore identify the TLEs that Arcsys must accept.

22.4.2.2. XML File Structure

The format handlers are specified in XML files. These files are stored in the `FORMAT_FILES_PATH/afp/tle/` directory. *`FORMAT_FILES_PATH`* parameter is set in the Arcsys Engine parameter file (for more details, see page 284, « ENGINE.properties File » description).

The TLE format handlers have the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xs:element name="descriptor">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="required"/>
      <xs:element ref="metadata"/>
    </xs:sequence>
    <xs:attribute name="encoding" use="required" type="xs:NCName"/>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="required">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="type"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="type" type="xs:NCName"/>
  <xs:element name="metadata">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="unbounded" ref="data"/>
      </xs:sequence>
    </xs:complexType>
   </xs:element>
<xs:element name="data">
    <xs:complexType>
      <xs:attribute name="format" type="xs:string"/>
      <xs:attribute name="name" use="required" type="xs:NCName"/>
    </xs:complexType>
</xs:element>
</xs:schema>
```

The above XSD file is available in the file `xml/afp/tle/native_descriptor_v1.xsd` in the Arcsys Engine installation directory. This file is composed of two parts: the mandatory values and the metadata.

22.4.2.3. Mandatory Elements

type: The standard tag matches the hexadecimal value of a TLE in the AFP structure

22.4.2.4. Metadata Elements

The data between the metadata tags match the TLE identifiers that Arcsys must take into account. It is also the joining point with Arcsys metadata.

22.4.2.5. Example

For the following XML file:

```
<descriptor encoding="Cp1047">
  <required>
    <type>d3a090</type>
  </required>
  <metadata>
    <data name="IssueDate" format="dd/MM/yy" />
    <data name="EditTypeCode" />
    <data name="InvoiceNumber" />
  </metadata>
</descriptor>
```

This TLE must be indicated with the standard `D3A090` tag.

Arcsys takes into consideration `IssueDate`, `EditTypeCode` and `InvoiceNumber` TLEs. If there are other TLEs, they are ignored.

22.4.3. Configuration

22.4.3.1. Mapping the Handler with the Mask

Mandatory keywords in the mask must be used in the handler, otherwise an error will occur.

Important

If no keyword is mandatory at mask level and the intersection between the mask and the handler is blank (e.g. the handler given does not match the file), the file will be archived but not indexed.

22.4.3.2. Configuring Arcsys Engine

For NOP-type AFP files, the storage directory of the handler files is the following: `FORMAT_FILES_PATH/afp/nop/`.

For TLE-type AFP files, the storage directory of the handler files is the following: `FORMAT_FILES_PATH/afp/tle/`.

22.5. Managing ZIP Files (ArcPak Option)

With this solution, you can insert ZIP or LZMA files directly in Arcsys. To do this, you must instruct Arcsys on how to index files contained in this type of record.

This solution involves including a mapping file in the ZIP/LZMA archives containing information on files contained in the ZIP/LZMA file.

In order to archive ZIP/LZMA files, three steps are needed, that are detailed in the following chapters:

1. In the `<FORMAT_FILES_PATH/>zip/csv/` directory, you must create an XML file that will be the format handler for your ZIP/LZMA files.
2. The ZIP/LZMA file must be formed as expected by Arcsys:
    1. Each ZIP/LZMA file must contain a CSV file that Arcsys will use for mapping with the mask used when archiving the ZIP/LZMA file.

Important

The encoding of the CSV file must be the same as the encoding of the system where the Arcsys Application Agent is located.

For example, if the Arcsys Application Agent runs on a Windows system, the CSV file must be in the CP-1252 format. If the Arcsys Application Agent runs on a Linux system with UTF-8 encoding, the CSV file must be in the UTF-8 format.

Important

You should also take care of the encoding of the file names inside the ZIP files. For example if you create the ZIP file on Windows and inject it in Arcsys on Linux, the file names having special characters would prevent archiving to succeed if the encoding of your ZIP tool is not defined to UTF-8 (for example on 7zip you may use the `-mcu` switch to force UTF-8). More generally in the case of ZIP creation on Windows, the use of the 7z format is recommended.

2. All included files must be at the root of the ZIP/LZMA file; directories or subdirectories are not supported.

3. When creating the archive, you must specify which format handler will apply to the objects contained in this archive (see Page 138).

22.5.1. Format Handler in XML Format

This file is an XML file that will define the format of the CSV file used as mapping file. The format handler defines the name of the mapping file, as well as the separator character for the CSV file columns or the metadata.

The XML files of format handlers are stored in `<FORMAT_FILES_PATH>/zip/csv/` directory. *<FORMAT_FILES_PATH>* parameter is set in the Arcsys Engine parameter file (for more details, see page 284, « ENGINE.properties File » file description).

The format handlers must match the ZIP XSD that can be found in `xml/zip/csv/` directory of Arcsys Engine installation directory. It has the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
 <descriptor name="name of the descriptor file in the ZIP file">
    <headerlinenumber value="numerical value" />
    <splitcharacter value="character" />
    <fileinfos>
      <filename columnnumber="numerical value" />
      <filehashfunction columnnumber="numerical value" />
      <filehashvalue columnnumber="numerical value" />
    </fileinfos>
    <metadata>
      <data columnnumber="numerical value" name="keyword name" />
      <data columnnumber="numerical value" format="decoding format" name="keyword name" />
    </metadata>
 </descriptor>
```

The format handler is formed by a general `descriptor` tag. This tag can have a `name` attribute, used to specify the name of the descriptor file included in each ZIP/LZMA file. The `descriptor` tag includes four tags:

- The `headerlinenumber` tag is optional. It is used to specify the number of lines used for headers in the CSV file. When no header is used in the CSV file, the `headerlinenumber` can be omitted or set to zero. When `n` header lines are used in the CSV file, the format handler should indicate `<headerlinenumber value="n" />`.

- The `splitcharacter` tag is required. It is used to specify the character used to separate the different columns of the mapping file.

- The `fileinfos` tag is used to indicate in which column is set the needed information. The `fileinfos` tag must include the following mandatory elements:

  - `filename`: identifies the number of the column that contains the file name of each file to be archived present in the ZIP/LZMA file.

  - `filehashfunction`: identifies the column number corresponding to the hashing algorithm used for each file handled.

  - `filehashvalue`: identifies the column number corresponding to the hash value of the file handled.

Note

The numbering of the columns starts from 0.

For example, to indicate that the file name is specified in the first column, the file hash function in the second column, and the file hash value in the third column, the `fileinfos` tag is:

```
<fileinfos>
          <filename columnnumber="0"/>
          <filehashfunction columnnumber="1"/>
          <filehashvalue columnnumber="2"/>
</fileinfos>
```

- The `metadata` tag contains metadata elements. Each Arcsys metadata, as well as the column number from which to retrieve it, is identified by a `data` tag.

  At least one `data` tag must be set.

  Each `data` tag contains the following attributes:

  - `name`: the name of the keyword. This name must match the keyword name set in Arcsys. It is required.

- `columnnumber`: the column number from which to retrieve the keyword value. This information is required.

Note

The numbering of the columns starts from 0.

- `format`: the format of the keyword. This information is optional. It can be used to specify a date format.

22.5.1.1. Example

For the following format handler XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
 <descriptor>
    <splitcharacter value=";" />
    <fileinfos>
      <filename columnnumber="0" />
      <filehashfunction columnnumber="1" />
      <filehashvalue columnnumber="2" />
    </fileinfos>
    <metadata>
      <data columnnumber="3" name="ContractNumber" />
      <data columnnumber="4" name="PortfolioCode" />
      <data columnnumber="5" format="yyyy-MM-dd" name="IssueDate" />
    </metadata>
 </descriptor>
```

This format handler specifies that the mapping file has the same name as the ZIP/LZMA file (no `name` is specified in `descriptor` element).

The mapping file, in CSV format, has six columns numbered from 0 to 5:

1. The column 0 corresponds to the file name.
2. Column 1 corresponds to its hash algorithm.
3. Column 2 corresponds to the hash value.
4. Column 3 corresponds to the ContractNumber metadata.
5. Column 4 corresponds to the PortfolioCode metadata.
6. Column 5 corresponds to the IssueDate metadata.

If there are other columns, they are ignored.

22.5.2. Mapping File in CSV Format

This file is a CSV file whose name is defined by the format handler: the name of the mapping file can be specified by the *name* attribute of the *descriptor* tag of the format handler. If this is not the case, the name of the mapping file must have the same name as the ZIP/LZMA file that it describes (`archive.zip` -> `archive.csv`).

Important

If Arcsys does not find this mapping file, it generates an error.

The mapping file contains a line for each file contained in the ZIP/LZMA file. For each line, it contains as much columns as defined in the format handler.

Important

The ZIP/LZMA file must only contain files; directories or subdirectories are not supported

Important

The encoding of the CSV file must be the same as the encoding of the system where the Arcsys Application Agent is located.

For example, if the Arcsys Application Agent runs on a Windows system, the CSV file must be in the CP-1252 format. If the Arcsys Application Agent runs on a Linux system with UTF-8 encoding, the CSV file must be in the UTF-8 format.

The column corresponding to the hashing algorithm used must be filled with one of the following values:

- `M` or `MD5`, for MD5 hash function
- `S` or `SHA1`, for SHA1 hash function
- `U` or `SHA256`, for SHA256 hash function
- `T` or `SHA512`, for SHA512 hash function

The hashing algorithm can not be left blank or set to a value corresponding to 'no hash value'.

The column corresponding to the hash value may be empty. If it is filled in, it ensures the fixity of the file because Arcsys checks this value; otherwise the hash value is calculated by Arcsys.

22.5.3. Associating a format handler

When archiving a ZIP/LZMA file, the format handler corresponding to the mapping file has to be specified in the *format* field of the characteristics of the object.

Note

The *format* field is only available in the administration interface of the Arcsys Web Agent.

Mandatory keywords in the mask must be used in the handler, otherwise an error will occur.

Important

If no keyword is mandatory at mask level and the intersection between the mask and the handler is blank (e.g. the handler given does not match the file), the files in the zip will be archived without metadata.

23. Configuring ArcRFT Option

23.1. Installing GenericSearch

To use ArcRFT Option, you must first install GenericSearch as described in the GenericSearch manual. These binaries are released in a distinct installer.

23.2. Configuring Arcsys to use GenericSearch

The configuration file `THIRD_PARTY_SEARCH_ENGINE.properties` enables you to give the characteristics of GenericSearch (which is considered as a "Third Party" search engine) to Arcsys Core modules. This configuration file can be duplicated in the modules using it, i.e Arcsys REST API, Arcsys Engine, Arcsys Application Agent, and Arcsys Web Agent.

23.2.1. Parameter description

23.2.1.1. Third party search engine parameters

Parameter name	**THIRD_PARTY_URL**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	URL to the Third Party Search Engine. This parameter must be defined if an indexing policy using an external Search Engine is configured.
Shipped value	No shipped value.

Parameter name	**THIRD_PARTY_CLIENT_ID**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Identifier of the client application for OAuth2 authentication. For GenericSearch it is the value of *`clientDetails.clientId`*.
Shipped value	arcsys

Parameter name	**THIRD_PARTY_CLIENT_SECRET**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Secret shared by the client applications of the Third Party Search Engine (OAuth 2.0). For GenericSearch it is the value of *`clientDetails.clientSecret`*.
Shipped value	No shipped value.

23.3. Defining an indexing policy

To use GenericSearch, an indexing policy must be properly configured. For more details, refer to the Arcsys Functional Description Manual.

23.4. Using GenericSearch in GUI

By default, the search page does not allow you to conduct a search using GenericSearch. For that purpose, you must modify the `WEBAGENT_guipreferences.xml` configuration file either globally or for a specific user or group (see page 417, « etc/WEBAGENT_guipreferences_example.xml File »):

- `DISPLAY_SEARCH_ENGINE_SELECTION` enables you to choose whether the selection of the search engine is possible in the GUI or not;
- `SEARCH_ENGINE_SELECTION_DEFAULT_VALUE` enables you to choose the default search engine that will be used for all criteria (not only for full text criteria);
- `DISPLAY_CRITERIA_THIRD_PARTY_SEARCH` enables you to display the full text input field on all criteria;
- `DISPLAY_CONTENT_THIRD_PARTY_SEARCH` enables you to display the full text input field on document contents.

24. Configuring Arcsys Engine REST callbacks

When an Arcsys request arrives in terminal status, it is possible to send an HTTP request containing some information about this Arcsys request to one of your internal web services.

24.1. Configuration

The parameters about the request destination URL and the way to transmit Arcsys requests information are described in page 284, « ENGINE.properties File » (check the parameters with the prefix `SEND_REST_REQUESTS`).

Note

To perform HTTPS requests, you must also specify the information about the keystore and the truststore in these parameters.

25. Attestations

The attestations are documents imposed by the `NFZ-42-013` standard which are intended to attest that an action has been correctly carried out in an electronic archiving system. In this sense, it is not like a proof folder, which is intended to demonstrate the evidentiary value when the need arises. A certain number of these attestations are therefore described in the standard and Arcsys is able to generate three types of attestations:

- `Archival` attestation;
- `Disposal` attestation;
- `Restitution` attestation.

These attestations can be generated by the Arcsys Engine directly following a Arcsys operation (successful archival, disposal or restitution request) or afterwards by calls to the Arcsys REST API.

25.1. Attestation Policies

25.1.1. Definition

An attestation policy allows to define which type of attestation can be generated as well as a set of parameters concerning their generation.

Important

For an attestation policy to be applied to a record, it must be associated with the record class (via ArcWeb, the Arcsys Web Agent or the Arcsys REST API).

25.1.2. Description File for an Attestation Policy

The description file for an attestation policy is an XML file that complies with the following structure:

```
<?xml version="1.0"?>
<ATTESTATION_POLICY>
    <!-- Standard attestation policy, all the attestation types are enabled -->
    <CODE>Attestation_Policy_EN</CODE>
    <LABEL>Standard attestation policy</LABEL>
    <CREATOR>USER</CREATOR>
    <ATTESTATIONS>
        <!-- Common parameters -->
```

```
        <PARAMETERS>
            <PARAMETER name="company.logo.path">attestations/policies/img/infotel.png</
PARAMETER>
            <PARAMETER name="company.information">Infotel Blagnac
                Avenue de l'Escadrille Normandie Niemen
                Parc Le Millénaire
                31700 Blagnac</PARAMETER>
            <PARAMETER name="identification.title">Identification</PARAMETER>
            <PARAMETER name="lot.id.label">Archive identifier:</PARAMETER>
            <PARAMETER name="lot.code.label">Archive code:</PARAMETER>
            <PARAMETER name="lot.label.label">Archive label:</PARAMETER>
            <PARAMETER name="classification.title">Classification information</PARAMETER>
            <PARAMETER name="repository.label">Repository:</PARAMETER>
            <PARAMETER name="collection.label">Collection:</PARAMETER>
            <PARAMETER name="class.label">Class:</PARAMETER>
            <PARAMETER name="operation.details.title">Details of the operation</PARAMETER>
            <PARAMETER name="operation.author.label">By:</PARAMETER>
            <PARAMETER name="operation.date.label">On:</PARAMETER>
            <PARAMETER name="content.title">Content</PARAMETER>
            <PARAMETER name="object.label">Document</PARAMETER>
            <PARAMETER name="attestation.date.label">Made by Infotel on</PARAMETER>
            <PARAMETER name="date.pattern">dd/MM/yyyy HH:mm:ss</PARAMETER>
        </PARAMETERS>
        <ATTESTATION type="archival" enabled="true">
            <!-- Specific parameters for archival attestation, all common parameters can be
 overloaded here -->
            <PARAMETERS>
                <PARAMETER name="attestation.title">Archival attestation</PARAMETER>
                <PARAMETER name="attestation.text">This document certifies the filing of
 documents in the Arcsys electronic archiving system. Below is a summary of the deposit
 made.</PARAMETER>
            </PARAMETERS>
        </ATTESTATION>
        <ATTESTATION type="disposal" enabled="true">
            <!-- Specific parameters for disposal attestation, all common parameters can be
 overloaded here -->
            <PARAMETERS>
                <PARAMETER name="attestation.title">Disposal attestation</PARAMETER>
                <PARAMETER name="attestation.text">You are hereby informed that the
 archive described below that you entrusted to us has been deleted in accordance with the
 contractually agreed retention rules.</PARAMETER>
            </PARAMETERS>
        </ATTESTATION>
        <ATTESTATION type="restitution" enabled="true">
            <!-- Specific parameters for restituion attestation, all common parameters can
 be overloaded here -->
            <PARAMETERS>
                <PARAMETER name="attestation.title">Restitution attestation</PARAMETER>
                <PARAMETER name="attestation.text">This document certifies the return of
 the documents for which you have entrusted us with responsibility. Please note that this
 document also notifies the transfer of responsibility for the permanent conservation of
 the said documents.
                    You will find below a summary of the restitution made.</PARAMETER>
            </PARAMETERS>
        </ATTESTATION>
    </ATTESTATIONS>
</ATTESTATION_POLICY>
```

The following must be specified for each attestation policy:

- A code corresponding to the Arcsys code for the attestation policy and which must be unique for a given Arcsys repository. A label and a user can also be specified and will be inserted in the Arcsys Database, but they are optional.

- A list of parameters that will be used to generate the attestations. These parameters and their use are described below.
- One or more attestation configuration (via `ATTESTATION` tag) with attributes specifying their type and indicating if the attestation policy allows to generate this type of attestation. The possible types for an attestation are `archival`, `disposal`, `restitution`. Each attestation configuration must be unique for a given type.

The parameters mentioned above particularly concern labels that will be displayed in the produced attestations. Each of these parameters is optional and can be specified at two levels:

- At the root `ATTESTATIONS` tag: these parameters will be considered as common to all types of attestations and will be used for the generation of an attestation if its configuration does not overload it;
- In a specific `ATTESTATION` tag: these parameters will be used for all attestation generations of the defined type.

The parameters that can be used are the following:

- *`company.logo.path`*: Path to the logo that will be displayed at the top of an attestation (the following suffixes are supported: jpg, jpeg, tif, tiff, gif, bmp and png);

Important

The logo must be accessible from the same path for the components generating the attestation: the Arcsys Engine, but also the Arcsys REST API, if it is used to generate attestations (for example via ArcWeb Module).

- *`company.information`*: Additional information that will be displayed just below the logo;
- *`attestation.title`*: Attestation title;
- *`attestation.text`*: Attestation text;
- *`identification.title`*: Title of the record identification part;
- *`lot.id.label`*: Label preceding the record identifier. If this parameter is not specified, the information will not be retrieved or displayed;
- *`lot.code.label`*: Label preceding the record code. If this parameter is not specified, the information will not be retrieved or displayed;
- *`lot.label.label`*: Label preceding the record label. If this parameter is not specified, the information will not be retrieved or displayed;

- *classification.title*: Title of the archive classification part;
- *repository.label*: Label preceding the label of the record repository. If this parameter is not specified, the information will not be retrieved or displayed;
- *collection.label*: Label preceding the label of the record collection. If this parameter is not specified, the information will not be retrieved or displayed;
- *class.label*: Label preceding the code of the record class. If this parameter is not specified, the information will not be retrieved or displayed;
- *operation.details.title*: Title of the operation details part;
- *operation.author.label*: Label preceding the author of the operation that made the generation of the attestation possible (archival, restitution or disposal request). If this parameter is not specified, the information will not be retrieved or displayed;
- *operation.date.label*: Label preceding the date of the operation that made the generation of the attestation possible (archival, restitution or disposal request). If this parameter is not specified, the information will not be retrieved or displayed;
- *content.title*: Title of the record content part;
- *object.label*: Label preceding each document of the record. If this parameter is not specified, the information will not be retrieved or displayed;
- *attestation.date.label*: Label preceding the date of the attestation generation. If this parameter is not specified, the information will not be retrieved or displayed;
- *date.pattern*: Format of the dates displayed in the attestation. If this parameter is not specified, the format by default will be `dd/MM/yyyy`.

Below is an example of rendering of an archival attestation generated from the attestation policy shown above.

Infotel Blagnac
Avenue de l'Escadrille Normandie Niemen
Parc Le Millénaire
31700 Blagnac

Archival attestation

This document certifies the filing of documents in the Arcsys electronic archiving system. Below is a summary of the deposit made.

Identification

Archive identifier: 22
Archive code: DOC_LOT
Archive label: Lot for documentation

Classification information

Repository: Default base
Collection: Default collection
Class: DEFAULT_CLASS1/DEFAULT_CLASS2

Details of the operation

By: arcsys42super
On: 12/05/2021 14:10:19

Content

Document 63, facture1.pdf, 81109bd534ca2ed86aa19ccb12cb2ad0646183ad00eb2605b31fab94bc4f27beaaa0e37fcf24cdbf6993f0ed7ceebccb7771b7ec41f7ffade619de55eb91bb03 (SHA512)
Document 64, facture2.pdf, 81109bd534ca2ed86aa19ccb12cb2ad0646183ad00eb2605b31fab94bc4f27beaaa0e37fcf24cdbf6993f0ed7ceebccb7771b7ec41f7ffade619de55eb91bb03 (SHA512)
Document 65, facture3.pdf, 81109bd534ca2ed86aa19ccb12cb2ad0646183ad00eb2605b31fab94bc4f27beaaa0e37fcf24cdbf6993f0ed7ceebccb7771b7ec41f7ffade619de55eb91bb03 (SHA512)

Made by Infotel on 12/05/2021 14:10:19

Page 1/1

Figure 25.1. Example of a generated archival attestation policy

25.1.3. Default Attestation Policies

Default attestation policies are supplied in the `attestations/policies` directory of the Arcsys Engine. These policies can be directly imported in the repositories. In these attestation policies, all types of attestation are enabled and all possible labels are filled in. These are the following policies:

- **Attestation_Policy_EN**: Standard policy with English labels.
- **Attestation_Policy_FR**: Standard policy with French labels.

25.1.4. Importing an attestation policy

In order to be accessible from Arcsys, attestation policies must be imported using the `attestationpolicy` script delivered in the Arcsys Engine (or possibly by calling the Arcsys REST API). The use of the script is detailed page 355, « attestationpolicy.sh (or .bat) Script ».

25.1.5. Attaching an attestation policy to an Arcsys class

In order to generate attestations on a record, it must be linked to a class which must itself have an attestation policy attached to it (for the class creation, see Interface Guide).

To attach an attestation policy to a class, you must open the `class edition` menu. Then you can select the desired attestation policy via the field `Attestation policy`.

Note

If you have a single attestation policy for a given collection, you can use the field `Default class for the new lots` in the collection edition menu, that will avoid selecting the desired class for each lot.

26. Workflow

This part describes the different steps to create and use a workflow policy with Arcsys.

You need first to determine the workflow policies that can be used by the different archiving processes.

A workflow policy is an XML file used to define a set of rules concerning the workflow validation. These rules are used to specify the steps requiring validation.

Important

For a workflow policy to be applied to a record, it must be associated with the record class.

26.1. Workflow policy description file

The description file for a workflow policy is an XML file that uses the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<WORKFLOW_POLICY version="2.0">
    <CODE>STANDARD_WORKFLOW_POLICY</CODE>
    <LABEL>Standard Workflow Policy</LABEL>

    <TASK_SCHEMES>
        <TASK_SCHEME type="validation"
                     step="pre_disposal"
                     group-mode="parallel"
                     min-groups="2">
            <CODE>PRE_DISPOSAL_VALIDATION</CODE>
            <LABEL>Pre-disposal validation task scheme</LABEL>
            <GROUPS>
                <GROUP>ARCHIVE</GROUP>
                <GROUP min-validators="2">LEGAL</GROUP>
                <GROUP>MANAGEMENT</GROUP>
            </GROUPS>
        </TASK_SCHEME>

        <TASK_SCHEME type="validation"
                     step="restitution"
                     group-mode="sequential"
                     min-groups="0">
            <CODE>RESTITUTION_VALIDATION</CODE>
            <LABEL>Restitution validation task scheme</LABEL>
            <GROUPS>
                <GROUP order="1">ARCHIVE</GROUP>
                <GROUP order="2">LEGAL</GROUP>
                <GROUP order="3">MANAGEMENT</GROUP>
            </GROUPS>
        </TASK_SCHEME>
    </TASK_SCHEMES>
</WORKFLOW_POLICY>
```

The first task scheme uses parallel validation. All specified groups can act at the same time, and the workflow is approved when two of the three groups have approved it.

The `LEGAL` group requires approvals from two distinct users before it is considered approved.

The second task scheme uses sequential validation. Since *`min-groups`* is set to `0`, all specified groups must approve the workflow. The groups become active one after another, according to their *`order`* value.

Each workflow policy must define the following information:

- A `CODE` that matches the code of the workflow policy and that is unique in a given repository.
- A `LABEL` that describes the workflow policy.
- One or more task schemes, each corresponding to a breakpoint in the workflow.

Each task scheme must define the following information:

- A `type` attribute that defines the action to handle at this step. Currently, only the `validation` type is available. A `validation` task requires one or more user approvals before the workflow can continue.
- A `step` attribute that identifies the point in the workflow at which the validation task is triggered. Currently, the following steps are available:
  - `pre_disposal`, which corresponds to a breakpoint triggered when a disposal request is created for the record. The disposal request is suspended until the required approvals are obtained.
  - `restitution`, which corresponds to a breakpoint triggered during a restitution request, after data retrieval and before data deletion. The restitution request is suspended until the required approvals are obtained.

Important

When a workflow is rejected:

- **For a `pre_disposal` workflow, the disposal request stops and Arcsys applies a disposal hold to the record. To proceed with the disposal, remove the disposal hold and create a new disposal request.**
- **For a `restitution` workflow, the request stops after data retrieval. To complete the restitution up to data deletion, create a new restitution request.**

- An optional `group-mode` attribute that specifies how the configured validator groups are processed:

  - `parallel` (default): all configured groups can approve or reject the workflow simultaneously.
  - `sequential`: groups are activated according to their *`order`* attribute.
- An optional `min-groups` attribute that defines the minimum number of groups that must approve the workflow.

  If this attribute is omitted or set to `0`, all configured groups must approve the workflow.

  If this attribute is greater than `0`, the workflow is approved as soon as the specified number of groups have approved it.

  For example:

  - Three groups with `min-groups="0"`: all three groups must approve.
  - Three groups with `min-groups="2"`: any two groups must approve.
  - Five groups with `min-groups="3"`: any three groups must approve.
- A `CODE` element that defines the task scheme code. This code must be unique in a given repository.
- A `LABEL` element that describes the task scheme.
- A `GROUPS` element containing one or more `GROUP` elements. Each `GROUP` identifies a group of users authorized to approve or reject the workflow. Its value must match an existing group name in the LDAP directory.

  Each `GROUP` element can define the following optional attributes:

  - `min-validators`: minimum number of distinct users from the group required for the group to be considered approved. The default value is `1`.
  - `order`: activation order of the group when `group-mode="sequential"` is used. Groups with the same *`order`* value are activated simultaneously during the same validation phase. The default value is `1`. This attribute is ignored when `group-mode="parallel"` is used.

26.1.1. Workflow approval and rejection rules

A workflow is approved when the required number of configured groups has approved it. This number is defined by the `min-groups` attribute.

- If `min-groups` is omitted or set to `0`, all configured groups must approve the workflow.

- If `min-groups` is greater than `0`, the workflow is approved as soon as this number of groups has approved it.

Each group has its own approval rule. By default, one approval is enough for a group to be considered approved. This can be changed with the optional `min-validators` attribute.

When a group requires several validators, the group is approved only after the required number of distinct users has approved it. However, the group still counts as only one approved group toward the `min-groups` requirement.

A workflow is automatically rejected when it becomes mathematically impossible to reach the required number of approved groups.

A group can also be rejected. As long as the group has not already been approved, a single rejection from an authorized user rejects the group. This rejection may reject the whole workflow immediately if the required approval quorum can no longer be reached.

Example:

```
<TASK_SCHEME type="validation"
             step="restitution"
             group-mode="parallel"
             min-groups="3">
    <GROUPS>
        <GROUP>DIRECTION</GROUP>
        <GROUP>LEGAL</GROUP>
        <GROUP>ARCHIVE</GROUP>
        <GROUP>ADMINISTRATOR</GROUP>
    </GROUPS>
</TASK_SCHEME>
```

This workflow contains four groups and requires approvals from three groups.

1. `DIRECTION` rejects the workflow. The workflow remains in `PENDING` status because the three remaining groups can still approve it.

2. `LEGAL` rejects the workflow. Only two groups remain available. Since three approvals are required, the workflow is immediately marked as `REJECTED`.

The rejection is therefore triggered automatically as soon as the approval quorum can no longer be satisfied.

26.1.2. Workflow actions and users belonging to multiple groups

Workflow approval and rejection actions are available in ArcWeb and through the Arcsys REST API. The legacy Web application is not modified for this feature, except explicit exceptions.

A user cannot act more than once for the same group in the same workflow instance.

However, if a user belongs to several groups involved in the workflow, they may approve or reject once for each group to which they belong, provided that the group is currently eligible. A validation performed as one group does not prevent the same user from validating as another group.

When multiple workflow groups are available for the same user, ArcWeb requires the user to select the group on whose behalf the action is performed. When the action is performed through the Arcsys REST API, the acting group must be specified when several groups are eligible.

Note

The `min-validators` value must be consistent with the number of distinct users who can act for the group. If this value is greater than the number of eligible users for the group, the group cannot be approved unless the configuration, group membership, or permissions are changed.

26.2. Workflow policy examples

The following examples illustrate how workflow validation behaves under different policy configurations.

26.2.1. Example 1: Default behavior (all groups must approve)

When the `min-groups` attribute is omitted or set to `0`, all configured groups must approve the workflow. This is the default behavior.

```
<TASK_SCHEME type="validation" step="pre_disposal">
    <CODE>PRE_DISPOSAL_VALIDATION</CODE>
    <LABEL>Pre-disposal validation</LABEL>
    <GROUPS>
        <GROUP>ARCHIVE</GROUP>
        <GROUP>LEGAL</GROUP>
        <GROUP>MANAGEMENT</GROUP>
    </GROUPS>
</TASK_SCHEME>
```

In this example, the workflow is approved only after the `ARCHIVE`, `LEGAL`, and `MANAGEMENT` groups have approved it.

26.2.2. Example 2: Approval quorum

The `min-groups` attribute defines the minimum number of groups required to approve the workflow. In the following example, approval from any two groups out of the three configured groups is sufficient.

```
<TASK_SCHEME type="validation"
             step="restitution"
             group-mode="parallel"
             min-groups="2">
    <CODE>RESTITUTION_VALIDATION</CODE>
    <LABEL>Restitution validation</LABEL>
    <GROUPS>
        <GROUP>ARCHIVE</GROUP>
        <GROUP>LEGAL</GROUP>
        <GROUP>MANAGEMENT</GROUP>
    </GROUPS>
</TASK_SCHEME>
```

Since the workflow uses `parallel` mode, all configured groups can act at the same time. The workflow is approved as soon as any two groups have approved it.

Example scenarios:

- `ARCHIVE` approves, `LEGAL` approves → workflow approved.
- `ARCHIVE` approves, `MANAGEMENT` approves → workflow approved.
- `LEGAL` rejects, `ARCHIVE` approves, `MANAGEMENT` approves → workflow approved.

26.2.3. Example 3: Multiple validators in the same group

The `min-validators` attribute specifies the minimum number of distinct users required to validate a group.

```
<TASK_SCHEME type="validation"
             step="pre_disposal">
    <CODE>PRE_DISPOSAL_VALIDATION</CODE>
    <LABEL>Pre-disposal validation</LABEL>
    <GROUPS>
        <GROUP>ARCHIVE</GROUP>
        <GROUP min-validators="2">LEGAL</GROUP>
        <GROUP>MANAGEMENT</GROUP>
    </GROUPS>
</TASK_SCHEME>
```

Since the `group-mode` attribute is not specified, the workflow uses the default `parallel` mode and all configured groups can act simultaneously.

In this example, all groups must approve the workflow because `min-groups` is not specified. However, the `LEGAL` group is considered approved only after two distinct users belonging to this group have validated the workflow.

Note

The two validations required for the `LEGAL` group make `LEGAL` one approved group. They do not count as two groups toward the workflow approval rule.

Example:

1. User A validates as `LEGAL` → progress: 1/2.
2. User B validates as `LEGAL` → progress: 2/2.
3. The `LEGAL` group is now approved.

If User B rejects instead of validating, the `LEGAL` group is rejected because it has not yet reached its two required approvals. Since this example does not define `min-groups`, all configured groups must approve the workflow. Therefore, the rejection of `LEGAL` rejects the whole workflow.

26.2.4. Example 4: Sequential validation

The `group-mode` and `order` attributes define a validation sequence.

```
<TASK_SCHEME type="validation"
             step="pre_disposal"
             group-mode="sequential"
             min-groups="2">
    <CODE>PRE_DISPOSAL_VALIDATION</CODE>
    <LABEL>Pre-disposal validation</LABEL>
    <GROUPS>
        <GROUP order="1">ARCHIVE</GROUP>
        <GROUP min-validators="2" order="2">LEGAL</GROUP>
        <GROUP order="3">MANAGEMENT</GROUP>
    </GROUPS>
</TASK_SCHEME>
```

In this example, the `ARCHIVE` group becomes active first. Once this group has approved or rejected the workflow, the `LEGAL` group becomes active. The `LEGAL` group requires approvals from two distinct users. The `MANAGEMENT` group becomes active only if the workflow has not already reached a final status.

Since this workflow requires approvals from two groups (`min-groups="2"`), the workflow may be completed before all configured groups become active.

Note

The two validations required for the `LEGAL` group make `LEGAL` one approved group. They do not count as two groups toward the workflow approval rule.

Example scenarios:

- `ARCHIVE` approves → `LEGAL` is approved by two distinct users → workflow approved. Since two groups have approved the workflow, the approval quorum is reached and `MANAGEMENT` never becomes active.
- `ARCHIVE` approves → one user validates as `LEGAL` → another authorized user rejects as `LEGAL` → `LEGAL` is rejected and `MANAGEMENT` becomes active:
  - If `MANAGEMENT` approves, the workflow is approved because two groups, `ARCHIVE` and `MANAGEMENT`, have approved it.

  - If `MANAGEMENT` rejects, the workflow is rejected. Only one group has approved the workflow, and no other group remains available to reach the required two approvals.

- `ARCHIVE` rejects → `LEGAL` is approved by two distinct users → `MANAGEMENT` becomes active:
  - If `MANAGEMENT` approves, the workflow is approved because two groups, `LEGAL` and `MANAGEMENT`, have approved it.
  - If `MANAGEMENT` rejects, the workflow is rejected. Only one group has approved the workflow, and no other group remains available to reach the required two approvals.
- `ARCHIVE` rejects → one user validates as `LEGAL` → another authorized user rejects as `LEGAL` → workflow rejected. No group has approved the workflow and only `MANAGEMENT` remains available, so the required approval from two groups can no longer be reached.

26.3. Importing a Workflow policy

In order to be accessible from Arcsys, workflow policies must be imported using the `workflowpolicy.sh(.bat)` script delivered in the Arcsys Engine, or by calling the Arcsys REST API. The use of the script is detailed page 357, « workflowpolicy.sh (or .bat) Script ».

Workflow policies and task schemes can also be created separately, and then linked together, by calling the Arcsys REST API.

26.4. Default workflow policy

A default workflow policy is supplied in the `workflows/policies` directory of the Arcsys Engine. This policy can be imported in the repositories with the **workflowpolicy.sh(.bat)** command.

Note

Each workflow policy defines a group of users authorized to validate or reject the workflow. The example set in the default workflow policy must be updated before importing it in Arcsys.

26.5. Attaching a workflow policy to an Arcsys class

In order to activate a workflow policy on a record, the record must be linked to a class which must itself have a workflow policy attached to it (for the class creation, see Interface Guide).

To attach a workflow policy to a class, you must open the `class edition` menu. Then you can select the desired workflow policy via the field `Workflow policy`.

Note

If you have a single workflow policy for a given collection, you can use the field `Default class for the new lots` in the collection edition menu, that will avoid selecting the desired class for each record.

Part 2. Configuration files and scripts description

1. Overview

This section of the document provides information on the following elements for each product module:

- Architecture: Organization of the directories, file names, number of files, etc.;
- Configuration files: List of the parameters for each file, definition of each parameter, etc.;
- Script files: Behavior, parameters, return code, etc.

2. Parameter Files

2.1. Separators

The parameter files can be changed. However, it is important that the Unix separation format is respected for the parameters that contain paths. In other words, the ‘slash’ (/) even if the module is installed on a Windows workstation.

Example: C:/Program Files/Infotel

3. Arcsys RMI, TCP/IP and SOAP API

3.1. Overview

The various Arcsys APIs are installed in a directory called `ArcsysApi`.

3.2. Directory Structure

Type	Name	Description
	`launch_arcsys_api.sh (or .bat)`	Module launch script.
	`status_arcsys_api.sh (or .bat)`	Module status script.
	`shutdown_arcsys_api.sh (or .bat)`	Module stop script.
	`installService_api.cmd (non systematic)`	Module Windows service installation script.
	`removeService_api.cmd (non systematic)`	Module Windows service uninstallation script.
	`etc`	Directory containing the module configuration files.
	`arcsysJAAS.conf`	This file contains the parameters of the Java authorization and authentication service.
	`ARCSYS_RMI_API.properties`	Main configuration file of the RMI API module.
	`ARCSYS_TCPIP_API.properties`	Main settings file of the TCP/IP API module.
	`ARCSYS_WS_API.properties`	Main settings file of the API Web services module.
	`log4j2.xml`	log4j configuration file.
	`policy`	This file contains the parameters of the module's Java class permissions.
	`REFERENTIAL_JDBC.properties`	This file contains the parameters for accessing the Arcsys Database in JDBC mode.

	jar	Directory containing the module Arcsys Java libraries (JAR).
	libext	Directory containing the different external Java libraries required in the module.
	log	Directory containing a list of all the log files. By default, these files are automatically rotated each day.
	trace	Directory containing a list of all the trace files. By default, these files are automatically rotated each day.
	ws	This directory contains elements required to manually install web services.
	xml	Directory containing the signature profile definition XSD file and a sample signature profile XML file.

3.3. etc/log4j2.xml file

3.3.1. Definition

This file contains the settings for the log file. The logs are generated with the log4J Apache library.

3.3.2. Parameters

The parameters of this file depend on the information desired in the logs. Hence please consult the standard documentation of Log4J for any changes.

See: https://logging.apache.org/log4j/2.x/

3.4. etc/arcsysJAAS.conf File

This file contains the parameters of the Java authorization and authentication service. See the description in JAAS configuration file.

3.5. etc/REFERENTIAL_JDBC.properties File

This file contains the parameters for accessing the Arcsys Database in JDBC mode. See the description in page 123, « Configuring JDBC Access to the Relational Database ».

3.6. etc/ARCSYS_RMI_API.properties File

3.6.1. Definition

This file contains the parameters for providing availability and subsequent access to the RMI service of the Arcsys API.

3.6.2. Arcsys RMI API Parameters

Parameter name	**COMPONENT_API_PORT_NUMBER**
Parameter type	Port number
Parameter change assessment	After module reboot.
Description	RMI service port of the Arcsys RMI API. This port is only used if the RMI API is deployed (*RMI_API_DEPLOY*= `true`) which is the default case after installation.
Shipped value	`25040`

Parameter name	**COMPONENT_API_SERVICE_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Name of the Arcsys RMI API service.
Shipped value	`ArcsysRmiAPI`

Parameter name	**SITE_CODE**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Site code to which the module must belong to. This site must be created in the Arcsys Web beforehand (by default, the installer creates systematically a `PRIMARY_SITE` code site).
Shipped value	`PRIMARY_SITE` (primary site created by default during the installation step).

Parameter name	**RMI_API_DEPLOY**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies whether the RMI API is deployed on launching of this module.
Shipped value	`true`

Parameter name	**SEC_API_TRANSACTIONS_TIMEOUT**
Parameter type	Number of seconds.
Parameter change assessment	After module reboot.
Description	Timeout after which an inactive transaction will be cancelled and closed.
Shipped value	`86400`(1 day)

Parameter name	**SIMULTANEOUS_API_TRANSACTIONS_NUMBER**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Maximum number of transactions allowed. After this limit, a new attempt to create a transaction will result in error.
Shipped value	`1000`

Parameter name	**CTRLPROC_LOOPSLEEPTIME**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	Timeout of transaction control process. The lower the limit, the quicker it will detect a transaction timeout.
Shipped value	`1000`(1 second)

Parameter name	**SEC_TIME_REFRESH_LDAP_USER**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	If ALWAYS_CACHE_LDAP_USER is `true`, specifies the time in seconds after which it should clean the cache so that the next connection is done in a new call to the LDAP.
Shipped value	`21600`(1 second)

Parameter name	**DEFAULT_DOMAIN_FOR_METADATA_LIST**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	When the methods for obtaining metadata lists (`getMetadataLotListByLot` for example) are called without specifying a domain, this parameter specifies from which domain the metadata must be sent: `BOTH`(for all domains), `BUSINESS` (business unit only),`TECHNICAL` (technical only)
Shipped value	`BOTH`

Parameter name	**ONLINE_RESTORE_REQUEST**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	`true` if an synchronous retrieval request must be created in the relational database during synchronous retrieval by the RMI API or web service, otherwise `false`. If ArcHP Option is used, you should set this parameter to `false`.
Shipped value	`true`

Parameter name	**TEST_FILE_AVAILABILITY_FOR_ONLINE_RESTORE**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies whether the API must test the status of the presence of files to synchronously retrieve on the transfer servers before sending the request. When you know with certainty, via the configuration, that the files to display are still present, set this parameter to `false`. For example, if the files are on file systems and the "Authorize direct access for synchronous retrieval" option is activated on the file systems.
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**USE_CONSTANT_TYPE_FOR_ALL_ONLINE_RESTORE**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies whether the Arcsys type (native, AFP, ZIP, etc.) of files to synchronously retrieve with this API is still the same. This prevents loading in the database on each display.
Shipped value	`false`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**NATIVE_TYPE_FOR_ALL_ONLINE_RESTORE**
Parameter taken into account only if:	*`USE_CONSTANT_TYPE_FOR_ALL_ONLINE_RESTORE`*`= true`
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Specifies the Arcsys constant type to be used by this API for the synchronous retrievals. Possible values: `AFP_NOP`, `AFP_TLE`, `ZIP_CSV`, `NOT_NATIVE`.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**SIZE_FOR_CACHE_ONLINE_RESTORE_MOVER_REFERENCE**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Number of envelope cache entries for the synchronous retrieval (0, by default). If 0, there is no cache. If > 0, a cache limited by this value is implemented (everything depends on the size of the memory given to the API). In this way, the characteristics of envelopes containing the object to retrieve are not reloaded systematically if an object of this envelope has already been viewed. The oldest entries are not contained in the cache.
Shipped value	`0`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**LOAD_ONLINE_STATUS_FOR_SEARCH_ARCHIVES**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies during a call to the `searchArchives` method whether you should load the "online" status of each record from the Arcsys Database. Setting it to `false` will improve search performance.
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**LOAD_ARCHIVED_STATUS_FOR_SEARCH_ARCHIVES**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies during a call to the `searchArchives` method whether you should load the "archived" status of each record from the Arcsys Database. Setting it to `false` will improve search performance.
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**ALWAYS_CACHE_LDAP_USER**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies whether an LDAP user cache should be implemented. If this is the case, in the next connection with the same user in the API, it does not reconnect to the LDAP, but simply checks the hash of the password transmitted. If ArcHP Option is used, you should set this parameter to `true`.
Shipped value	`false`

3.6.3. General parameters for communication with the Arcsys Transfer Server

Parameter name	**TRANSFER_SERVERS**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	List of the transfer server codes with which the API module must communicate for synchronous retrieval and creating/deleting file systems. For synchronous retrieval, the request is tried successively on the transfer servers of the list, until it succeeds, in the order given by *TRANSFER_SERVER_STRATEGY*. For creating/deleting file systems, one of the reachable transfer servers is contacted. This parameter is not mandatory. If left blank, all transfer servers of the site will be taken into consideration, in the order given by *TRANSFER_SERVER_STRATEGY*.
Shipped value	No shipped value.

Parameter name	**HEALTH_CHECK_TRANSFER_SERVERS_PROC_LOOPSLEEPTIME**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	Control interval for checking the presence of transfer servers, for failover or load balancing management.
Shipped value	No shipped value.

Parameter name	**TRANSFER_SERVER_STRATEGY**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Definition of the access mechanism to the transfer servers. Possible values are `FIRST_AVAILABLE` and `RANDOM`. `FIRST_AVAILABLE` means that the request is always sent to the first transfer servers, then to the second if the request fails, and so on. `RANDOM` operates in the same way, but on each request, a transfer server is chosen randomly from the list of transfer servers available. If ArcHP Option is used, you should set this parameter to `RANDOM`.
Shipped value	`RANDOM`

3.6.4. Performance-related API parameters

Parameter name	**ONLINE_RESTORE_REQUEST**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	`true` if an synchronous retrieval request must be created in the relational database during synchronous retrieval by the RMI API or web service, otherwise `false`. If ArcHP Option is used, you should set this parameter to `false`.
Shipped value	`true`

Parameter name	**TEST_FILE_AVAILABILITY_FOR_ONLINE_RESTORE**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies whether the API must test the status of the presence of files to synchronously retrieve on the transfer servers before sending the request. When you know with certainty, via the configuration, that the files to display are still present, set this parameter to `false`. For example, if the files are on file systems and the "Authorize direct access for synchronous retrieval" option is activated on the file systems.
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**USE_CONSTANT_TYPE_FOR_ALL_ONLINE_RESTORE**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies whether the Arcsys type (native, AFP, ZIP, etc.) of files to synchronously retrieve with this API is still the same. This prevents loading in the database on each display.
Shipped value	`false`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**NATIVE_TYPE_FOR_ALL_ONLINE_RESTORE**
Parameter taken into account only if:	*`USE_CONSTANT_TYPE_FOR_ALL_ONLINE_RESTORE`*`= true`
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Specifies the Arcsys constant type to be used by this API for the synchronous retrievals. Possible values: `AFP_NOP`, `AFP_TLE`, `ZIP_CSV`, `NOT_NATIVE`.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**SIZE_FOR_CACHE_ONLINE_RESTORE_MOVER_REFERENCE**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Number of envelope cache entries for the synchronous retrieval (0, by default). If 0, there is no cache. If > 0, a cache limited by this value is implemented (everything depends on the size of the memory given to the API). In this way, the characteristics of envelopes containing the object to retrieve are not reloaded systematically if an object of this envelope has already been viewed. The oldest entries are not contained in the cache.
Shipped value	`0`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**LOAD_ONLINE_STATUS_FOR_SEARCH_ARCHIVES**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies during a call to the `searchArchives` method whether you should load the "online" status of each record from the Arcsys Database. Setting it to `false` will improve search performance.
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**LOAD_ARCHIVED_STATUS_FOR_SEARCH_ARCHIVES**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies during a call to the `searchArchives` method whether you should load the "archived" status of each record from the Arcsys Database. Setting it to `false` will improve search performance.
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**ALWAYS_CACHE_LDAP_USER**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies whether an LDAP user cache should be implemented. If this is the case, in the next connection with the same user in the API, it does not reconnect to the LDAP, but simply checks the hash of the password transmitted. If ArcHP Option is used, you should set this parameter to `true`.
Shipped value	`false`

3.6.5. LDAP Directory Parameters

For LDAP parameters (preset by LDAP_), see Description of Different LDAP Parameters.

LDAP_CAS_SINGLE_SIGN_ON: If `true`, indicates whether you want to use CAS ticket (Central Authentication Service) authentication to the API (`connect` method) instead of the user/password combination. In this case, the *LDAP_CAS_SERVER_URL* parameter will be used to determine the CAS server to use.

LDAP_CAS_SERVER_URL: Indicates the URL of the CAS server where *LDAP_CAS_SINGLE_SIGN_ON* is true.

3.7. etc/ARCSYS_TCPIP_API.properties File

3.7.1. Definition

This file contains the parameters for providing availability and subsequent access to the Arcsys API TCP/IP service. This API is intended for internal use in Arcsys only (Arcsys Transfer Server, Arcsys Web Agent).

3.7.2. Arcsys TCP/IP API parameters

TCPIP_API_DEPLOY : `true` if the Arcsys TCP/IP API service must be deployed during API startup; otherwise `false` . By default, its value is: `true` ;

COMPONENT_API_TIMEOUT : Timeout of the connection to the API. Default value: `3200` ;

3.7.3. Communication Parameters with the Arcsys Transfer Server

For parameters setting the communication protocol with the Arcsys Transfer Server (preset by TRANSFER_SERVER), see Encryption Method with the Transfer Server.

3.8. etc/ARCSYS_WS_API.properties File

3.8.1. Definition

This file contains the parameters for providing availability and subsequent access to the web services of the Arcsys API.

3.8.2. Arcsys SOAP API API parameters

Parameter name	**WS_API_PORT_NUMBER**
Parameter type	Port number
Parameter change assessment	After module reboot.
Description	Port for the web services of the Arcsys API. This port is only used if the Web Services API is deployed (*WS_API_DEPLOY*= `true`).
Shipped value	`50060`

WS_API_DEPLOY : `true` if the Arcsys API web services must be deployed when launching the APIs, otherwise `false`. By default, its value is: `true` ;

WS_API_ONLINE_RESTORE_SERVICE_NAME Name of the MTOM web service of the Arcsys API for synchronous retrieval methods. By default, its value is: `ArcsysOnlineRestoreWS` ;

WS_API_INFER_CONTENT_TYPE_FROM_FILENAME `true` if the value of the Content-type field of the HTTP header in the SOAP response must be deduced from the extension of the retrieved file (for synchronous retrieval). By default, its value is: `true` .

3.9. launch_arcsys_api.sh (or .bat) Script

This script deploys the Arcsys APIs (RMI, TCP/IP, etc.) on local host. The API clients can then recover them and run their methods.

3.10. shutdown_arcsys_api.sh (or .bat) Script

This script stops the Arcsys APIs (RMI, TCP/IP).

Hence, the Arcsys RMI API service can only be recovered by a lookup and API clients that have already recovered this service can only make new calls to the API methods. However, the calls in progress to the API methods will continue to work normally. Once all the calls in progress have been completed, this script finishes, as well as the API launch script, which finishes in the same way.

3.11. status_arcsys_api.sh (or .bat) Script

This script recovers a return code matching the current status of the Arcsys API.

Return code value	Agent status	Description
64	STARTED_AND_WORKING	The Arcsys APIs are deployed and ready for use (`launch_arcsys_api` script launched).
66	SHUTDOWN	The Arcsys APIs are stopped and not ready for use (except for calls in progress) (`shutdown_arcsys_api` script launched).
69	GET_STATUS_ERROR	Error in retrieving the status.

3.12. installService_api.cmd Script

This script deploys the Arcsys RMI, TCP/IP and SOAP API as a Windows service.

3.13. removeService_api.cmd Script

This script uninstalls the application service as a Windows service.

4. Arcsys REST API

4.1. Overview

The Arcsys REST API is installed in a directory called `ArcsysApiRestWS`.

4.2. Directory Structure

Type	Name	Description
	`launch_apiRestWS.sh (or .cmd)`	Module launch script.
	`status_apiRestWS.sh (or .cmd)`	Module status script.
	`shutdown_apiRestWS.sh (or .cmd)`	Module stop script.
	`installService_apiRestWS.cmd (non systematic)`	Module Windows service installation script.
	`removeService_apiRestWS.cmd (non systematic)`	Module Windows service uninstallation script.
	`etc`	Directory containing the module configuration files.
	`arcsys-rest.properties`	Main configuration file of the REST API module.
	`log4j2.xml`	Main log4j configuration file.
	`log4j2_status.xml`	log4j configuration file for the status command.
	`log4j2_shutdown.xml`	Log4j configuration file for the stop command.
	`policy`	This file contains the parameters of the module's Java class permissions.
	`REFERENTIAL_JDBC.properties`	This file contains the parameters for accessing the Arcsys Database in JDBC mode.
	`THIRD_PARTY_SEARCH_ENGINE.properties`	This file contains the settings of the third party search engine.
	`jar`	Directory containing the module's Java libraries (jar).

	`log`	Directory containing a list of all the log files. By default, these files are automatically rotated each day.
	`trace`	Directory containing a list of all the trace files. By default, these files are automatically rotated each day.

4.3. etc/log4j2.xml, etc/log4j2_status.xml and etc/log4j2_shutdown.xml files

4.3.1. Definition

These files contain the settings for the log files. The logs are generated with the Apache library log4j.

4.3.2. Parameters

The parameters of this file depend on the information desired in the logs. Therefore, consult the standard documentation of Log4J for any changes.

See: https://logging.apache.org/log4j/2.x/

4.4. etc/arcsys-rest.properties file

4.4.1. Definition

This file contains the main parameters for the module.

4.4.2. Base parameters

Parameter name	**HTTP_PORT**
Parameter type	Port number
Parameter change assessment	After module reboot.
Description	Port for the REST web services of the Arcsys API.
Shipped value	`8090`

Parameter name	**HTTPS_PORT**
Parameter type	Port number
Parameter change assessment	After module reboot.
Description	Port for the REST web services of the Arcsys API for use with SSL protocol.
Shipped value	`8090`

Parameter name	**CLIENT_DETAILS_CLIENT_ID**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Identifier of the client application (OAuth 2.0).
Shipped value	`arcsys_rest`

Parameter name	**CLIENT_DETAILS_CLIENT_SECRET**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Secret shared by the client application (OAuth 2.0).
Shipped value	No shipped value.

Parameter name	**ACCESS_TOKEN_VALIDITY**
Parameter type	Number of seconds.
Parameter change assessment	After module reboot.
Description	Validity period of access tokens.
Shipped value	`300`

Parameter name	**TOKEN_IDLE_TIME**
Parameter type	Number of seconds.
Parameter change assessment	After module reboot.
Description	Idle time after which the token is not refreshed and the user needs to authenticate again in the REST API. If this time is exceeded, the user must re-authenticate himself by starting a new authentication process.
Shipped value	`1800`

Parameter name	**SWAGGER_ENABLED**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Flag to indicate if the REST API Swagger documentation must be activated. It is recommended to disable Swagger documentation (with value `false`) in production for security reasons.
Shipped value	`false`

Parameter name	**WORKSPACE_DIRECTORY**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Defines the workspace directory. This workspace is used to store temporary files (i.e. between uploading a file and archiving it). **Caution** In cluster mode, the workspace directory must be shared between all the Arcsys REST API instances. **Note** This directory is shared with Arcsys Application Agent. Both modules must have access to it. See Setting rights on directories shared between Arcsys modules for more information.
Shipped value	`/tmp`

Parameter name	**CLEAN_EXPIRED_WORKSPACE_FILES**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Flag to indicate if files that are expired (see *`WORKSPACE_FILES_LIFESPAN`*) should be deleted from workspace directory.
Shipped value	`true`

Parameter name	**EXPIRED_WORKSPACE_FILES_CLEANER_INTERVAL**
Parameter type	List of intervals in CRON format for launching the creation (except for copy) and treatment process, each interval being separated by the ";" character. The paragraph page 353 (« Managing CRON Format ») gives more details about this format. **Note** The first value of the interval corresponds to the seconds and must be set to `*`. If anything else than `*` (for example `0`) is specified for this first value, an issue can occur, as a check is made that the current time matches the CRON expression specified. In the example, if `0` is specified, it would only match when the number of seconds of the current time is equal to 0.
Parameter change assessment	After module reboot.
Description	Defines the cron interval to delete expired workspace files.
Shipped value	`0 0 0 * * ?`

Parameter name	**WORKSPACE_FILES_LIFESPAN**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Defines the lifespan (in days) of a workspace file before it expires.
Shipped value	`7`

Parameter name	**AGT_CODES**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	Defines the codes of the Arcsys Application Agents list to be utilized for uploads in the Arcsys REST API (this requires using ArcBWS APIs). These agents must possess the capability to access the *`WORKSPACE_DIRECTORY`* folder of the Arcsys REST API.
Shipped value	No shipped value.

Parameter name	**AGT_STRATEGY**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the strategy used to get an application agent from the list specified in *AGT_CODES*. Possible values are: • `FIRST_AVAILABLE`: First active agent found in the list. • `ROUND_ROBIN`: Circular queue of active agents found in the list. • `RANDOM`: Random active agent found in the list.
Shipped value	`FIRST_AVAILABLE`

Parameter name	**STOP_JETTY_TOKEN**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Stop token of the embedded Jetty server.
Shipped value	`9lFVfB9ES30=`

Parameter name	**SSL_KEYSTORE_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the keystore containing the certificate used for SSL/TLS exchanges. For this component, the keystore file is required if SSL is enabled. In this case, ensure that the server hostname matches exactly with the hostname specified in the keystore. If this is not possible, you can add the parameter `-Darcsys.rest.hostname=<hostname>` in the `status_apiRestWS` and `shutdown_apiRestWS` scripts to "force" the hostname to match the one in the certificate.
Shipped value	No shipped value.

Parameter name	**SSL_KEYSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**SSL_KEYSTORE_TYPE**
Parameter type	Characters string. Possible values are: `jks` or `pkcs12`.
Parameter change assessment	After module reboot.
Description	Type of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	`jks`

Parameter name	**SSL_KEYSTORE_CERTALIAS**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Alias of the certificate in the keystore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**SSL_TRUSTSTORE_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**SSL_TRUSTSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password for the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**SSL_TRUSTSTORE_TYPE**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Type of the truststore used for SSL/TLS exchanges.
Shipped value	`jks`

4.4.3. OAuth2 multi-client parameters

The following parameters are used to define multiple OAuth2 clients.

Parameter name	**OAUTH2_CLIENTS**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.

Description	Enables OAuth2 multi-client mode and defines the list of OAuth2 client identifiers. When this parameter is present, the OAuth2 configuration switches from legacy single-client mode to multi-client mode. Each client listed must have its own dedicated configuration section.
	
	For example, if you define `OAUTH2_CLIENT=client1,client2` you must define the following lines:
	
	`OAUTH2_CLIENT.client1.CLIENT_ID`
	`OAUTH2_CLIENT.client1.CLIENT_SECRET`
	`OAUTH2_CLIENT.client1.GRANT_TYPES`
	`OAUTH2_CLIENT.client1.LDAP_USER`
	`OAUTH2_CLIENT.client1.LDAP_PASSWORD`
	`OAUTH2_CLIENT.client1.TOKEN_VALIDITY`
	`OAUTH2_CLIENT.client2.CLIENT_ID`
	`OAUTH2_CLIENT.client2.CLIENT_SECRET`
	`OAUTH2_CLIENT.client2.GRANT_TYPES`
	`OAUTH2_CLIENT.client2.LDAP_USER`
	`OAUTH2_CLIENT.client2.LDAP_PASSWORD`
	`OAUTH2_CLIENT.client2.TOKEN_VALIDITY`
	
	Note
	
	If this parameter is defined together with legacy OAuth2 parameters (`CLIENT_DETAILS_CLIENT_ID/CLIENT_DETAILS_CLIENT_SECRET`), multi-client mode takes priority.

Shipped value	No shipped value.
Parameter available since version (if applicable)	26.2.STS

All parameters below are defined per OAuth2 client, using the prefix: OAUTH2_CLIENT.<xxx>.

Parameter name	**OAUTH2_CLIENT.<client_name>.CLIENT_ID**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the OAuth2 client identifier used during authentication. This value is exposed as the client_id during OAuth2 token requests.
Shipped value	No shipped value.
Parameter available since version (if applicable)	26.2.STS

Parameter name	**OAUTH2_CLIENT.<client_name>.CLIENT_SECRET**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Defines the OAuth2 client secret associated with the client identifier. This value is required for `client_credentials` grant type.
Shipped value	No shipped value.
Parameter available since version (if applicable)	26.2.STS

Parameter name	**OAUTH2_CLIENT.<client_name>.GRANT_TYPES**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	Defines the OAuth2 grant types supported by the client. Multiple grant types can be configured for the same client.
Possible values	`password`, `client_credentials`
Shipped value	No shipped value.
Parameter available since version (if applicable)	26.2.STS

Parameter name	**OAUTH2_CLIENT.<client_name>.TOKEN_VALIDITY**
Parameter type	Number of seconds.
Parameter change assessment	After module reboot.
Description	Defines the access token validity duration in seconds for the OAuth2 client. This parameter is optional; if it is not set, the value `ACCESS_TOKEN_VALIDITY` is used instead.
Shipped value	No shipped value.
Parameter available since version (if applicable)	26.2.STS

The following parameters are mandatory only when the client_credentials grant type is enabled.

Parameter name	**OAUTH2_CLIENT.<client_name>.LDAP_USER**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the LDAP user associated with the OAuth2 client when using the client_credentials grant type. This user is used to authenticate the client and determine its security context.
Shipped value	No shipped value.
Parameter available since version (if applicable)	26.2.STS

Parameter name	**OAUTH2_CLIENT.<client_name>.LDAP_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Defines the LDAP password associated with the configured LDAP user for the client_credentials grant type.
Shipped value	No shipped value.
Parameter available since version (if applicable)	26.2.STS

4.4.4. General parameters for communication with the Arcsys Transfer Server

Parameter name	**TRANSFER_SERVERS**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	List of the transfer server codes with which the API module must communicate for synchronous retrieval and creating/deleting file systems. For synchronous retrieval, the request is tried successively on the transfer servers of the list, until it succeeds, in the order given by *TRANSFER_SERVER_STRATEGY*. For creating/deleting file systems, one of the reachable transfer servers is contacted. This parameter is not mandatory. If left blank, all transfer servers of the site will be taken into consideration, in the order given by *TRANSFER_SERVER_STRATEGY*.
Shipped value	No shipped value.

Parameter name	**HEALTH_CHECK_TRANSFER_SERVERS_PROC_LOOPSLEEPTIME**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	Control interval for checking the presence of transfer servers, for failover or load balancing management.
Shipped value	No shipped value.

Parameter name	**TRANSFER_SERVER_STRATEGY**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Definition of the access mechanism to the transfer servers. Possible values are `FIRST_AVAILABLE` and `RANDOM`. `FIRST_AVAILABLE` means that the request is always sent to the first transfer servers, then to the second if the request fails, and so on. `RANDOM` operates in the same way, but on each request, a transfer server is chosen randomly from the list of transfer servers available. If ArcHP Option is used, you should set this parameter to `RANDOM`.
Shipped value	`RANDOM`

4.4.5. Cluster management parameters

The following parameters enable the Arcsys REST API to operate in cluster mode:

Parameter name	**JWT_JWS_KEY_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the file containing the signature key for tokens; the key is randomly generated on startup if no file is specified. The signature key must correspond to a file containing 256 random bits and must be generated by `generate-key` script. **Important** **This parameter is mandatory for cluster mode: the file must be the same for all APIs involved in cluster mode.**
Shipped value	No shipped value.

Parameter name	**CLEAN_EXPIRED_TOKENS**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Flag to indicate if the REST API must delete expired tokens in database. In a cluster context, it is recommended to keep it to `true` on all APIs.
Shipped value	`true`

Parameter name	**TOKEN_CLEANER_PERIOD**
Parameter taken into account only if:	*CLEAN_EXPIRED_TOKENS*= `true`
Parameter type	Number of seconds.
Parameter change assessment	After module reboot.
Description	The time to wait between each deletion of expired tokens in database.
Shipped value	`1800` (30 min)

4.4.6. SSO Parameters

The following parameters are required when using SSO with Arcsys REST API

Parameter name	**SSO_PROTOCOL**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Type of SSO protocol used.
Possible values	`SAML2` or empty
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_PROTOCOL_VERSION**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the protocol version for SSO.
Possible values	`2.0`
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_PROTOCOL_BINDING**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the protocol binding for SSO. This parameter is optional. If no value is set, the default value is `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`
Possible values	• `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST` • `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_AUTHN_CONTEXT**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The authorization context class. It indicates how a user authenticates to an Identity Provider. This parameter is optional. For more information, see your Identity Provider's configuration or the OASIS documentation section 3 for the Security Assertion Markup Language (SAML) V2.0 Specification (https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf).
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS
Example	*SSO_AUTHN_CONTEXT*=`urn:oasis:names:tc:SAML:2.0:ac:classes:Password`

Parameter name	**SSO_SERVICE_PROVIDER_ID**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The identifier of the Arcsys REST API in the identity provider configuration (Client ID).
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_IDENTITY_PROVIDER_ID**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The identifier of the Identity Provider (the service that serves as the source of identity information and authentication decision).
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_LOGIN_REQUEST_URL**
Parameter type	URL, formed like: `<protocol>://<domain_name or ip_address>[:<port_number>][/<uri>]`
Parameter change assessment	After module reboot.
Description	The request URL for the login. This is the URL used to send authentication requests.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_LOGOUT_REQUEST_URL**
Parameter type	URL, formed like: `<protocol>://<domain_name or ip_address>[:<port_number>][/<uri>]`
Parameter change assessment	After module reboot.
Description	The URL used to send logout requests.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_REST_API_URL**
Parameter type	URL, formed like: `<protocol>://<domain_name or ip_address>[:<port_number>][/<uri>]`
Parameter change assessment	After module reboot.
Description	The Arcsys REST API base URL used for the SSO callbacks. This corresponds to the internal mechanism used by Arcsys for the login and the logout.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_USER_ID_KEY**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Attribute Name which identifies the user in the SSO response.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_DEFLATER_ENABLED**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Indicates if the SSO request must be compressed.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_INFLATER_ENABLED**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Indicates if the SSO response must be decompressed.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_NAME_ID_FORMAT**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The name id format. Controls the format of the subject of the SAML2 assertion. Select a value supported by your IDP.
Possible values	• `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified` • `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress` • `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName` • `urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName` • `urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos` • `urn:oasis:names:tc:SAML:2.0:nameid-format:entity` • `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` • `urn:oasis:names:tc:SAML:2.0:nameid-format:transient` • `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted`
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

4.4.6.1. Signature configuration

These parameters are optional. They have to be set when using SSO with signature.

Parameter name	**SSO_SIGNATURE_ALGORITHM**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The signature algorithm to sign SSO requests.
Possible values	`MD5withRSA`, `SHA1withDSA`, `SHA256withDSA`, `SHA1withRSA`, `SHA224withRSA`, `SHA256withRSA`, `SHA384withRSA`, `SHA512withRSA`, `HmacSHA1`, `HmacSHA224`, `HmacSHA256`, `HmacSHA384`, `HmacSHA512`
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_KEYSTORE_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("`/`"), even on Windows.
Parameter change assessment	After module reboot.
Description	The keystore file to generate the signature in the SSO request.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_KEYSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	The keystore password to generate the signature in the SSO request.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_KEYSTORE_TYPE**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The keystore type to generate the signature in the SSO request.
Possible values	`jceks`, `jks` or `pkcs12`
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_KEYSTORE_KEY_ALIAS**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The alias to use in the keystore to generate the signature in the SSO request.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_KEYSTORE_KEY_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	The keystore key password to generate the signature in the SSO request.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

4.4.6.2. Signature verification

These parameters are optional. They have to be set when using SSO with signature verification.

Parameter name	**SSO_CERT_SUBJECT_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The subject name of the certificate used for SSO. This parameter is used to verify the signature in the SSO response. This parameter is mandatory if `SSO_TRUSTSTORE_FILE` is not empty.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_TRUSTSTORE_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	The truststore file to check the signature in the SSO response.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_TRUSTSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	The truststore password to check the signature in the SSO response.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_TRUSTSTORE_TYPE**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The truststore type to check the signature in the SSO response.
Possible values	`jceks`, `jks` or `pkcs12`
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

4.4.6.3. Requests Cleaner

These parameters are used to specify the behavior of the requests cleaner when using SSO. At least one Arcsys REST API must have these parameters set when using SSO.

Parameter name	**CLEAN_EXPIRED_SSO_REQUESTS_TASK**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Flag to indicate if the Arcsys REST API must delete expired SSO requests.
Shipped value	false
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**EXPIRED_SSO_REQUESTS_CLEANER_INTERVAL**
Parameter type	List of intervals in CRON format for launching the creation (except for copy) and treatment process, each interval being separated by the ";" character. The paragraph page 353 (« Managing CRON Format ») gives more details about this format. **Note** The first value of the interval corresponds to the seconds and must be set to `*`. If anything else than `*` (for example `0`) is specified for this first value, an issue can occur, as a check is made that the current time matches the CRON expression specified. In the example, if `0` is specified, it would only match when the number of seconds of the current time is equal to 0.
Parameter change assessment	After module reboot.
Description	Cron interval to clean expired SSO requests.
Shipped value	0 0 0 * * ?
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SSO_REQUESTS_LIFESPAN_MINUTES**
Parameter type	Number of minutes.
Parameter change assessment	After module reboot.
Description	The lifespan of SSO requests.
Shipped value	5
Parameter available since version (if applicable)	2024.1.LTS

4.4.7. Performance-related API parameters

Parameter name	**ONLINE_RESTORE_REQUEST**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	`true` if an synchronous retrieval request must be created in the relational database during synchronous retrieval by the RMI API or web service, otherwise `false`. If ArcHP Option is used, you should set this parameter to `false`.
Shipped value	`true`

Parameter name	**TEST_FILE_AVAILABILITY_FOR_ONLINE_RESTORE**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies whether the API must test the status of the presence of files to synchronously retrieve on the transfer servers before sending the request. When you know with certainty, via the configuration, that the files to display are still present, set this parameter to `false`. For example, if the files are on file systems and the "Authorize direct access for synchronous retrieval" option is activated on the file systems.
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**USE_CONSTANT_TYPE_FOR_ALL_ONLINE_RESTORE**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies whether the Arcsys type (native, AFP, ZIP, etc.) of files to synchronously retrieve with this API is still the same. This prevents loading in the database on each display.
Shipped value	`false`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**NATIVE_TYPE_FOR_ALL_ONLINE_RESTORE**
Parameter taken into account only if:	*`USE_CONSTANT_TYPE_FOR_ALL_ONLINE_RESTORE`*`= true`
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Specifies the Arcsys constant type to be used by this API for the synchronous retrievals. Possible values: `AFP_NOP`, `AFP_TLE`, `ZIP_CSV`, `NOT_NATIVE`.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**SIZE_FOR_CACHE_ONLINE_RESTORE_MOVER_REFERENCE**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Number of envelope cache entries for the synchronous retrieval (0, by default). If 0, there is no cache. If > 0, a cache limited by this value is implemented (everything depends on the size of the memory given to the API). In this way, the characteristics of envelopes containing the object to retrieve are not reloaded systematically if an object of this envelope has already been viewed. The oldest entries are not contained in the cache.
Shipped value	`0`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**LOAD_ONLINE_STATUS_FOR_SEARCH_ARCHIVES**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies during a call to the `searchArchives` method whether you should load the "online" status of each record from the Arcsys Database. Setting it to `false` will improve search performance.
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**LOAD_ARCHIVED_STATUS_FOR_SEARCH_ARCHIVES**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies during a call to the `searchArchives` method whether you should load the "archived" status of each record from the Arcsys Database. Setting it to `false` will improve search performance.
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcHP Option

Parameter name	**ALWAYS_CACHE_LDAP_USER**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Specifies whether an LDAP user cache should be implemented. If this is the case, in the next connection with the same user in the API, it does not reconnect to the LDAP, but simply checks the hash of the password transmitted. If ArcHP Option is used, you should set this parameter to `true`.
Shipped value	`false`

4.4.8. Last logon Parameters

Parameter name	**LAST_LOGON_ENABLED**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Flag to indicate whether the last logon information of a given user should be saved in the database.
Shipped value	`true`
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**LAST_LOGON_EXCLUDED_CLIENTS**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	Defines the list of client names for which the last logon information will not be saved in the database. This client name is retrieved during authentication via the HTTP header `Arcsys-Client-Id`.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**LAST_LOGON_EXCLUDED_USERS**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	Defines the list of usernames for which the last logon information will not be saved in the database.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**LAST_LOGON_EXCLUDED_GROUPS**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	Defines the list of group names for which the last logon information will not be saved in the database.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

4.4.9. LDAP Directory Parameters

For LDAP parameters (preset by LDAP_), see Description of Different LDAP Parameters.

LDAP_CAS_SINGLE_SIGN_ON: If `true`, indicates whether you want to use CAS ticket (Central Authentication Service) authentication to the API (`connect` method) instead of the user/password combination. In this case, the *LDAP_CAS_SERVER_URL* parameter will be used to determine the CAS server to use.

LDAP_CAS_SERVER_URL: Indicates the URL of the CAS server where *LDAP_CAS_SINGLE_SIGN_ON* is true.

4.5. etc/REFERENTIAL_JDBC.properties File

This file contains the parameters for accessing the Arcsys Database in JDBC mode. See the description in page 123, « Configuring JDBC Access to the Relational Database ».

4.6. etc/THIRD_PARTY_SEARCH_ENGINE.properties File

4.6.1. Definition

This file contains the parameters for accessing the Third Party Search Engine for indexing or search. See its description in page 139, « Configuring ArcRFT Option ».

4.7. launch_apiRestWS.sh (or .cmd) Script

This script is used to start the Arcsys REST API on localhost.

4.8. shutdown_apiRestWS.sh (or .cmd) Script

This script is used to stop the Arcsys REST API on the canonical host name of the server. The host name must match the certificate configuration in case SSL is used. Alternatively, you can customize the host name by setting an additional parameter `-Darcsys.rest.hostname=...` in the command line of the script.

The calls in progress to the REST API services will continue to run normally until they end normally. Once all the calls in progress have been completed, this script finishes, as well as the API launch script, which finishes in the same way.

4.9. status_apiRestWS.sh (or .cmd) Script

This script is used to recover the Arcsys REST API status on the canonical host name of the server. The host name must match the certificate configuration in case SSL is used. Alternatively, you can customize the host name by setting an additional parameter `-Darcsys.rest.hostname=...` in the command line of the script.

Return code value	Agent status	Description
74	STARTED_AND_WORKING	The Arcsys REST APIs are deployed and ready for use (`launch_apiRestWS` script launched).
76	SHUTDOWN	The Arcsys REST APIs are stopped and not ready for use (except for calls in progress) (`shutdown_apiRestWS` script launched).
79	GET_STATUS_ERROR	Error in retrieving the status.

4.10. installService_apiRestWS.cmd Script

This script deploys the Arcsys REST API as a Windows service.

4.11. removeService_apiRestWS.cmd Script

This script uninstalls the application service as a Windows service.

4.12. Generating a JWS signature key for cluster mode (ArcHP Option)

The module may use two cryptographic keys: as other modules, the encryption key for the settings of the `etc/arcsys-rest.properties` file, but also (optionally) another key which is the JWS signature key for JWT tokens.

By default, the JWS signature key of JWT tokens is generated randomly each time the module is started and stored in the memory while the module is in use. It is used to sign the JWT tokens used to authenticate the REST API by using JWS.

For cluster mode, it is necessary to set the token signature key. This allows a module deployed on a cluster node to accept the tokens generated by a module deployed on another node. For this, you should create a file containing this key, then reference it in the *`JWT_JWS_KEY_FILE`* parameter mentioned above. For this key to be accepted on other nodes in a cluster, the file containing the key must be copied on each cluster node and the *`JWT_JWS_KEY_FILE`* parameter must also be entered for each node.

The JWS signature key is used in an HMAC-SHA-256 algorithm and can be generated by using the `generate-key` script. By default, this key will be called `secret-key` (settings encryption). You should rename it `signer-key` and move it to the `etc` directory if it is a signature key.

5. Arcsys Transfer Server

The Arcsys Transfer Server is installed in a directory called `ArcsysTransferServer`.

5.1. Directory Structure

Type	Name	Description
	`launch_transferServer.sh` (or .bat)	Module launch script.
	`status_transferServer.sh` (or .bat)	Module status script.
	`shutdown_transferServer.sh` (or .bat)	Module stop script.
	`clean_cache_transferServer.sh` (or .bat)	This script cleans the content of the Arcsys Transfer Server stage directory.
	`reload_transferServer.sh` (or .bat)	This script is used to reload specific parts of the Arcsys Transfer Server configuration.
	`rotate_logs_transferServer.sh` (or .bat)	This script triggers the rotation of the Arcsys Transfer Server logs.
	`monitor_transferServer.sh` (or .bat)	This script returns a monitoring report of the Arcsys Transfer Server.
	`keymanager_delete.sh` (or .bat)	This script deletes a key manager that is no more in use.
	`register_transferServer.sh` (or .bat)	This script records the Arcsys Transfer Server in the Arcsys Database.
	`installService_transferServer.cmd`	This script installs the Arcsys Transfer Server as a Windows service. This script is specific to the Windows platform.
	`removeService_transferServer.cmd`	This script uninstalls the Arcsys Transfer Server as a Windows service. This script is specific to the Windows platform.
	`bin`	Contains the executables for the installation platform.
	`lib`	Contains the shared and dynamic libraries used by the module.

	`log`	Directory containing a list of all the log files. By default, these files are automatically rotated each day.
	`restore-post-processing`	File post-processing directory. This directory is automatically purged after each synchronous retrieval involving post-processing.
	`stage`	Directory tree where files sent or received and files being recycled are stored.
	`xml`	Directory containing the last version of the XML schema for describing the manifests (also present in the application agent).
	`etc`	Directory containing the module configuration files.
	`LogParam.cfg`	This file contains the parameters of the logs.
	`transferServer.conf`	Main settings file of the module.
	`arcmover`	[ArcMover only] Directory containing the settings files for ArcMover.
	`arcMover.conf`	This file contains the parameters for ArcMover.
	`filesystems.conf`	Configuration file for file systems. This configuration file is only taken into account if it is set in `[changers]` section of `arcMover.conf` file. See "changers" Section of ArcMover.conf File (see Page 236).
	`<peripheral>.conf`	Main configuration file of the tape library (or changer).
	`mediatypes`	Directory containing media type files for ArcMover.
	`generic`	Directory containing the parameters of all the Generic media managers.

	`<configuration_name>`	Name to associate with the Generic configuration. If the default configuration is chosen for the Arcsys Transfer Server, the server name is `default` . This name corresponds to what is set in the Arcsys Web Agent in the `Configuration` field of the zone.
	`cloud`	Directory containing the configuration of all Cloud servers.
	`s3`	Directory containing the configuration of all S3 Cloud servers (for ArcMover S3 Option).
	`<configuration_name>`	Name to associate with the S3 Cloud configuration (for ArcMover S3 Option). If the default configuration is chosen, the configuration name is `default`. This name corresponds to what is set in the Arcsys Web Agent in the `Configuration` field of the zone.
	`cloud.conf`	S3 Cloud server configuration file (for ArcMover S3 Option).
	`layers`	Contains the layers used optionally in post-processing of an synchronous retrieval.
	`scripts`	Contains scripts that concerns specific options.
	`arcmover`	Contains scripts specific to ArcMover Tape Option.
	`drive_delete.sh`	This script removes a drive from the Arcsys Database (ArcMover Tape Option). This script is specific to the Unix platform.
	`drive_replace.sh`	This script replaces a drive by another (ArcMover Tape Option). This script is specific to the Unix platform.
	`flush_spool.sh` (or .bat)	This script flushes the zones of the Arcsys Transfer Server grouped by `Spooler`.

	`mailslot.sh` (or .bat)	This script moves a tape from or to a mailslot (ArcMover Tape Option).
	`tape_calibration.sh`	This script is used to request a tape calibration (ArcMover Tape Option). This script is specific to the Unix platform.
	`tape_delete.sh`	This script removes a tape from the Arcsys Database (ArcMover Tape Option). This script is specific to the Unix platform.
	`tape_dismount.sh`	This script unloads a tape from a drive. This script is specific to the Unix platform.
	`tape_library_delete.sh`	This script removes a tape library from the Arcsys Database (ArcMover Tape Option). This script is specific to the Unix platform.
	`tape_library_replace.sh`	This script moves tapes from a tape library to another one in the Arcsys Database (ArcMover Tape Option). This script is specific to the Unix platform.
	`tapecheck_cancel.sh`	This script is used to cancel a tapecheck operation (ArcMover Tape Option). This script is specific to the Unix platform.
	`tapecheck_dump.sh`	This script extracts the content of a tape on a drive managed by ArcMover in a specified directory (ArcMover Tape Option). This script is specific to the Unix platform.
	`tapecheck_history.sh`	This script is used to show an history of tapecheck operations (ArcMover Tape Option). This script is specific to the Unix platform.
	`tapecheck_integrity.sh`	This script is used to check the integrity of the envelopes on a given tape (ArcMover Tape Option). This script is specific to the Unix platform.
	`tapecheck_struct.sh`	This script is used to check the integrity of the envelopes on a given tape (ArcMover

		Tape Option). This script is specific to the Unix platform.
	`tapecheck_verify.sh`	This script is used to check the integrity of the envelopes and of objects inside the envelopes on a given tape (ArcMover Tape Option). This script is specific to the Unix platform.

5.2. transferServer.conf File

This file contains the parameters for the Arcsys Transfer Server.

5.2.1. Component code and host name parameters

To make the system open, the application agents, engines, Transfer Servers and Transfer Services have a unique code that is used to identify them in the configuration files or in the Arcsys interface. This way, the host of a module can be changed without it impacting the other modules. This code is determined by the administrator, who will choose a code with meaning for the context.

Parameter name in `transferServer.conf`	**component_code**
Parameter type	Characters string
Description	Name of this component. To avoid duplication, this name must be unique (on all sites). It is recommended to use a "logical" name distinct from a physical host name for example. Unique component code. The code must identify the module in the Arcsys installation.
Shipped value	`ArcsysTransferServerPrimary`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferServer.conf`	**component_port_number**
Parameter type	Characters string
Description	Transfer Server port number Port number on which the Transfer Server is deployed. For compatibility, this parameter also accepts a service name defined in /etc/services.
Shipped value	`25010`

Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferServer.conf`	**component_host**
Parameter type	Characters string
Description	Optional component host (to force the address of the component). If not empty, this host overrides the host chosen automatically. Here you can set an IP, a host name... This address will be the one used by the other modules. Module host. This parameter is optional. This is used to "force" the name under which the module will be registered in the Arcsys Database, overwriting the default name chosen by Arcsys. Use, for example, a localhost, an IP, a host name, etc.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**external_component_host**
Parameter type	Characters string
Description	Optional external component host. If set, this host will be used instead of the `component_host`. It will be used by components that are not in the same site. This optional parameter defines the address with which the Arcsys Transfer Server or the Arcsys Transfer Service will be contacted from a module present on a different site than the Arcsys Transfer Server or the Arcsys Transfer Service.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**site_code**
Parameter type	Characters string
Description	Code of the Transfer Server site Site code to which the module must belong. This site must be created in the Arcsys Web beforehand (by default, the installer creates systematically a PRIMARY_SITE site code).

Shipped value	`PRIMARY_SITE`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

5.2.2. Stage directories

These are the working spaces

Parameter name in `transferServer.conf`	**server_cache_dir**
Parameter type	Directory path
Description	This directory is the cache directory path. Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ('/'), even on Windows. Path to the cache zone of the server used as the transfer zone to and from the agent. **Important** **The path must correspond to an existing directory, with adequate access rights (read, write and execute).**
Shipped value	`${TARGET_FOLDER}/ArcsysTransferServer/stage/serv`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferServer.conf`	**cache_max_size**
Parameter type	Size in gigabytes (GiB)
Description	This parameter is the maximum amount of data that can be stored in cache. The decimal separator is the dot ('.'). Example(s): `1.5` for a cache of 1.5Gb **Note** The deletion of extra data from the cache area occurs after the end of the processing of a request. This means that the value

	specified as `cache_max_size` may occasionally be exceeded. This is the case when the Arcsys Transfer Server is processing a request on a lot which size is larger than the cache size, or when processing simultaneously several lots which total size exceeds the `cache_max_size`.
Shipped value	`1000`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.
Minimum value	1

Parameter name in `transferServer.conf`	**stage_delete_mode**
Parameter type	State variable. Possible values: `SIZE`, `DATE`
Description	Defines the priority order when deleting from cache is needed. This parameter defines the priority in deleting stage zones when extra space is required. This deletion takes place after the end of the processing of the request on the lot. The data concerned by an ongoing processing are not eligible for deletion. When this parameter is set to `DATE`, it gives priority to the deletion of the oldest lots. In this case, the oldest files have no reason to persist. When set to `SIZE`, it gives priority to the deletion of the smallest lots. It is therefore possible and normal for old files that are large enough to persist. The choice of the `stage_delete_mode` must be made considering the interest of the stage. The use of the files it contains makes it possible to optimize operations (mainly retrieval or recovery requests) subsequent to the archiving of a lot. This is possible because the envelopes are available on a medium (the stage area) which has been ideally chosen with better performance than those of the media on which the archives are permanently stored. The cache is all the more efficient when the files kept there: * are large: the gain provided by the cache area is all the more important as consequent I/O operations are involved. * are often used for retrieval operations. The `SIZE` choice makes it possible to favor the conservation of large files while `DATE` to preserve the files of the most recent archiving. The `SIZE` choice is particularly relevant if the size of the archives varies a lot (thus preserving files whose presence in the cache represents a particularly significant potential gain). If this is not the case, you might as well prefer the `DATE` choice unless the post-archiving operations generally concern batches archived a long time ago. In this case, the `SIZE` option, which performs a cleaning independent of the date, leaves a chance to old batches to be kept.

Shipped value	`SIZE`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferServer.conf`	**server_synchro_cache_dir**
Parameter type	Directory path
Description	This directory is the cache directory path. Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ('/'), even on Windows. Path to the cache zone of the server used to hold the files when synchronizing the manifest.
Shipped value	`${TARGET_FOLDER}/ArcsysTransferServer/stage/sync`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferServer.conf`	**server_online_dir**
Parameter type	Directory path
Description	This directory is the online retention directory path. Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ('/'), even on Windows. Path to the cache zone of the server used to hold the files. This parameter can be set to the same path as `server_cache_dir` if you want to use the same directory for the transfer zone to and from the agent as well as for online retention.
Shipped value	`${TARGET_FOLDER}/ArcsysTransferServer/online`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**online_max_size**
Parameter type	Size in gigabytes (GiB)

Description	Maximum size of the file retention zone.The decimal separator is the dot ('.'). This parameter is the maximum amount of data that can be stored in online retention directory (in Gb). This parameter is used only when server_online_dir is set Example(s): `1.5` for a online cache of 1.5Gb
Shipped value	`1000`
Parameter taken into account only if:	`server_online_dir` is different from
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	0

Parameter name in `transferServer.conf`	**online_storage_mandatory**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Defines if online storage is mandatory. This parameter is used only when `server_online_dir` and `server_cache_dir` are different This parameter is used to define whether you want archiving with online retention to give an error if the online retention zone is full. This parameter is ignored when the cache zone and the online retention zone are merged.
Shipped value	`false`
Parameter taken into account only if:	`server_online_dir` is different from
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferServer.conf`	**input_stage_dir**
Parameter type	Directory path
Description	It is only useful when the application agent is running on the same host as the transferServer. It must respectively match the `ARCH_REPOSITORY_PATH` application agent parameters. Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ('`/`'), even on Windows.

	Directory for copying files to the server's client side (used only when the server is used as a service, i.e. it is in the same machine as an application agent). This must be the same path as set in `ARCH_REPOSITORY_PATH` parameter of the Application agent located on the same machine. **Note** This directory may be shared with Arcsys Application Agent. In this case, both modules must have access to it.
Shipped value	`${TARGET_FOLDER}/ArcsysTransferServer/stage/cli/in`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferServer.conf`	**output_stage_dir**
Parameter type	Directory path
Description	It is only useful when the application agent is running on the same host as the transferServer. It must respectively match the `REST_REPOSITORY_PATH` application agent parameters. Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ('/'), even on Windows. Directory for receiving files from the server's client side (used only when the server is used as a service, i.e. it is in the same machine as an application agent). This must be the same path as set in `REST_REPOSITORY_PATH` parameter of the Application agent located on the same machine.
Shipped value	`${TARGET_FOLDER}/ArcsysTransferServer/stage/cli/out`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferServer.conf`	**recycling_stage_dir**
Parameter type	Directory path
Description	Recycling work directory. Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ('/'), even on Windows. A parameter determines the work directory where tape recycling will take place. This directory must contain disk space equal to the maximum capacity

	to be recycled multiplied by the number of potential simultaneous recycling processes (these recycling processes triggered by instant only by the Arcsys interface).
Shipped value	`${TARGET_FOLDER}/ArcsysTransferServer/stage/recycling`
Option(s)	`ArcMover Tape`: You must have this option to use this parameter.
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

5.2.3. Server adjustment

Parameter name in `transferServer.conf`	**retention_pool_factor**
Parameter type	Positive integer
Description	Defines the retention pool structure level. A factor of n ensures that there will not be more than 10 to the power of n files per directory into the retention pool area. Factor used in creating retrieval directories. Example(s): 2 insures 100 files max, 3 insures 1000 files max, etc...
Shipped value	`2`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1
Maximum value	19

Parameter name in `transferServer.conf`	**async_archive_session**
Parameter type	Positive integer
Description	Defines the number of threads used to handle storage-related operations (such as archiving and retrieval tasks).
Shipped value	`10`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**max_tcp_cli**
Parameter type	Positive integer
Description	Defines the maximum number of simultaneous TCP clients managed by the Transfer Server
Shipped value	25
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**max_tcp_retry**
Parameter type	Positive integer
Description	Maximum number of retry before considering all clients TCP has failed Maximum number of attempts for communication with a remote Arcsys Transfer Server (for registering purpose, at launch of the Arcsys Transfer Service).
Shipped value	3
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**delay_tcp_retry**
Parameter type	Number of seconds
Description	Retry delay before new attempt connection to server daemon. Delay between two contact attempts to a remote Arcsys Transfer Server (for registering purpose, at launch of the Arcsys Transfer Service).
Shipped value	5
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**nb_net_client_thread**

Parameter type	Positive integer
Description	Defines the number of threads used for data transfer operations (between different Arcsys Transfer Servers or with Arcsys Transfer Services).
Shipped value	`10`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**max_net_retry**
Parameter type	Positive integer
Description	Maximum number of retry before considering the file transfer failed Maximum number of attempts of file transfer between the Arcsys Transfer Server and the Arcsys Transfer Service.
Shipped value	`3`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**retry_delay**
Parameter type	Number of seconds
Description	Retry delay before new attempt to process the request. Delay between two attempts of file transfer between the Arcsys Transfer Server and the Arcsys Transfer Service or between the the Arcsys Transfer Server and the storage media, or of a recycling operation.
Shipped value	`15`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**up_interval**
Parameter type	Number of seconds
Description	Interval between each update of the up status in the database

	This parameter specifies the interval between each update of the up status of the Arcsys Transfer Server in the database. The up status is used for health monitoring of the Arcsys Transfer Server. It corresponds to a "I am alive" information sent by the Arcsys Transfer Server. This information is materialized by a database field, containing the date and time of the last update at which the Arcsys Transfer Server was running. When the last update date exceeds the refresh interval, the Arcsys Transfer Server can be considered as off or in error state. This status is used by the Arcsys Engine when sending requests, to privilege started Arcsys Transfer Servers. Setting this parameter at a low value allows a better reactivity in case of failure of a Arcsys Transfer Server. However, this involves more frequent connections to the database.
Shipped value	`30`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.
Minimum value	1
Maximum value	32767

5.2.4. Media Manager

Archiving Media Parameters

Parameter name in `transferServer.conf`	**arcmover_enabled**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Defines if ArcMover is enabled or not
Shipped value	`true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**fork_session_timeout**
Parameter type	Number of seconds
Description	Time-out for external media manager session Time-out for communications between the Arcsys Transfer Server and external media manager. When set to 0, no time-out is used.

Shipped value	`0`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**write_speed**
Parameter type	Positive integer
Description	Media writing speed threshold above which warnings are logged (MiB/s) This parameter specifies the speed of writing to media (disk, tape, etc.) below which a warning message appears in WARN. This is used to detect a hardware configuration problem. **Note** This parameter is only used for an ArcMover media
Shipped value	`1`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**jobactions_enabled**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Enables job actions for detailed archiving tracking in the web interface When set to true, `jobactions_enabled` activates the use of job action to track activity details on media managed by ArcMover (Tape or Filesystem). Job actions represent basic processes that the Arcsys Transfer Server can perform to carry out a job. When activated, they are displayed in the web interface and stored in the database. Their storage can significantly increase database volume. Enabling job actions also introduces additional exchanges between Arcsys Transfer Servers and Arcsys Engines, and possibly between multiple Arcsys Transfer Servers, which slightly slows down processing. Each read or write operation on a media generates around ten job actions per envelope. For example, archiving a lot of three envelopes on two zones creates about 60 job actions. Copying the same lot generates around 30 job actions for reading and 30 for writing.

	Note The default value for this parameter is `false` upon installation (to limit the amount of data stored in the database) and `true` during updates (to keep the same behavior as before).
Shipped value	`false`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

5.2.5. Arcsys API TCP/IP Connection properties

Parameters for connecting with the database and TCP/IP API

Parameter name in `transferServer.conf`	**api_host_name**
Parameter type	Characters string
Description	Host name of TCP/IP API
Shipped value	`localhost`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**api_port_number**
Parameter type	Characters string
Description	Port number of TCP/IP API Port number of the Arcsys TCP/IP API. For compatibility, this parameter also accepts a service name defined in `/etc/services`.
Shipped value	`25050`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**database_pool_connections_size**

Parameter type	Positive integer
Description	Number of always active connections in pool
Shipped value	`10`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**database_max_pool_connections_size**
Parameter type	Positive integer
Description	Maximum number of authorized connections.
Shipped value	`60`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**database_retry_get_connection**
Parameter type	Positive integer
Description	Number of attempts to get a connection from the pool.
Shipped value	`3`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**database_retry_get_connection_wait_ms**
Parameter type	Number of milliseconds
Description	Delay between two attempts to get a connection from the pool.
Shipped value	`1000`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**database_refresh_connections**
Parameter type	Number of seconds
Description	Time between each refresh of the connection pool A refresh checks each connection is still alive and reduces the number of connections down to database_pool_connections_size in case of inactive connections Time period after which all the connections perform a `select 1` to stay active. Keep in mind that any firewalls can cut off connections after x seconds. Consequently, a value lower than that set for the firewalls should be set.
Shipped value	`60`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

5.2.6. Communication protocol

Security Parameters

Important

The security parameters (protocol type, certificates) must be coherent on the Arcsys Transfer Server with those of the modules communicating with it (Arcsys API (RMI, SOAP or REST), Arcsys Web Agent, Arcsys Engine, Arcsys Transfer Services).

Parameter name in `transferServer.conf`	**security_protocol**
Parameter type	State variable. Possible values: `NONE`, `INTERNAL`, `SSL`
Description	Security protocol Indicates if communication to and from the Transfer Server is encrypted or not, and the type of encryption (symmetrical for INTERNAL: Blowfish algorithm is used; or asymmetrical for SSL). Disabling encryption increases performance but may be forbidden depending on the required security level. **Note** The value must be the same between all clients and servers exchanging data. It is not mandatorily the same on a global Arcsys installation.

Shipped value	`NONE`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferServer.conf`	**ssl_keystore_filename**
Parameter type	File path
Description	Path of the PKCS#12 keystore. Absolute file path. Use Unix like separators ('`/`'), even on Windows. Location of the keystore containing the certificate used for SSL/TLS exchanges. The file must mandatorily be in PKCS12 format.
Shipped value	**No shipped value.**
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**ssl_keystore_password**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Encrypted password of the PKCS#12 keystore Password of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	**No shipped value.**
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**ssl_truststore_filename**
Parameter type	File path

Description	Path of the PKCS#12 truststore. Absolute file path. Use Unix like separators ('/'), even on Windows. Location of the truststore used for SSL/TLS exchanges. The file must mandatorily be in PKCS12 format.
Shipped value	**No shipped value.**
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**ssl_truststore_password**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Encrypted password of the PKCS#12 truststore Password for the truststore used for SSL/TLS exchanges.
Shipped value	**No shipped value.**
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**ssl_library_libssl**
Parameter type	File path
Description	Optional path of the not provided dynamic library libssl if not standard. By default: on Linux, the library will be loaded by the path `libssl.so`. On Windows, the library will be loaded first by the path `ssleay32.dll` (Microsoft library), then `libssl32.dll` and then `libssl.dll`. Absolute file path. Use Unix like separators ('/'), even on Windows. Optional path of the libssl library used for SSL communication. This path will not be taken into account for other features requiring libssl (ArcCrypt Option uses the system installation). The version of this libssl library must be between 1.1.0 and 1.1.2. By default, the loaded libssl library path is `libssl.so` on Linux, and on Windows `ssleay32.dll` is first loaded, then in case of failure, `libssl32.dll` followed by `libssl.dll`.

	It is recommended to use this parameter if there are several versions of libssl installed on the system. Example(s): /usr/lib64/libssl.so.3
Shipped value	**No shipped value.**
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**ssl_library_libcrypto**
Parameter type	File path
Description	Optional path of the not provided dynamic library libcrypto if not standard. By default: on Linux, the library will be loaded by the path `libcrypto.so`. On Windows, the library will be loaded first by the path `libeay32.dll` (Microsoft library), then `libcrypto32.dll` and then `libcrypto.dll`. Absolute file path. Use Unix like separators ('`/`'), even on Windows. Optional path of the libcrypto library used for SSL communication. By default, the loaded libcrypto library path is `libcrypto.so` on Linux, and on Windows `libeay32.dll` first, then in case of failure, `libcrypto32.dll` followed by libcrypto.dll. Example(s): /usr/lib64/libcrypto.so.3
Shipped value	
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**ssl_module_path**
Parameter type	Characters string
Description	Optional path to the directory containing legacy cryptographic modules (e.g., `legacy.dll /legacy.so`) used by the SSL module. This may be required if your system relies on older cryptographic algorithms or libraries for compatibility reasons. Absolute file path. Use Unix like separators ('`/`'), even on Windows.
Shipped value	

Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**ip_whitelist**
Parameter type	List of string separated by commas
Description	List of IPs authorized to contact the Transfer Server. If empty, all IPs are authorized. This list defines the workstations that can send requests to the Arcsys Transfer Server. This parameter is composed of a comma-separated list of addresses. If this parameter has a value, only the requests from the workstations whose IP appears in the list will be authorized. Other requests will not receive a response. The ip_whitelist parameter can be left blank, in which case all connection requests will be authorized. If this parameter is used, the IP of the site on which the server is launched must be added to run the status, reload and shutdown scripts. According to the site configuration, this IP will be the local IP (generally, 127.0.0.1) or the network IP (e.g. 192.168.40.120). Example(s): To restrict communication between an Arcsys Transfer Server on a site where the IP is 192.168.40.120 and a service on a site where the IP is 192.168.40.130, the parameters are: - for the server: ip_whitelist=192.168.40.120 - for the service: ip_whitelist=192.168.40.130 The Arcsys Transfer Server can contact the service on site 192.168.40.130. In any case, since the IP of the server is not on its list of authorized IPs, this service will not answer the request.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

5.2.7. Cluster management parameters

The following parameters enable the Transfer Server to operate in cluster mode.

Important

Stage areas of the Transfer Servers must be separated. They must not be shared, as they are not managed by the clustering mode.

Sharing the same folders would cause anomalies during the requests processing.

`ArcHP`: You must have this option to use these parameters.

Section of parameters taken into account only if: `OS` = `linux`

Parameter name in `transferServer.conf`	**clustering_enabled**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Enables the cluster behavior of Transfer Server Specifies whether you authorize operation of the Transfer Server in cluster mode. This mode authorizes in particular media sharing. **Important** **All Transfer Servers of a given site that have `clustering_enabled` set to true are in the same cluster. The Transfer Servers of another site will be in another cluster: a cluster cannot be distributed over several sites.**
Shipped value	`false`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**cluster_port_number**
Parameter type	Positive integer
Description	Port number of the Transfer Server inside the cluster Port number of the Transfer Server on the cluster (TCP), for incoming and outcoming cluster messages. This port number is not mandatorily the same for all nodes of the cluster.
Shipped value	`25011`
Parameter taken into account only if:	`clustering_enabled` = `true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1000
Maximum value	65535

Parameter name in `transferServer.conf`	**cluster_nodes_connect_timeout**
Parameter type	Number of seconds
Description	Timeout for trying to contact each node (started or not) of the cluster. When this timeout is reached, the node will not be able to start.
Shipped value	`30`
Parameter taken into account only if:	`clustering_enabled` = `true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**cluster_nodes_ready_timeout**
Parameter type	Number of seconds
Description	Timeout for waiting validation for each started node of the cluster. When this timeout is reached, the node will not be able to start.
Shipped value	`15`
Parameter taken into account only if:	`clustering_enabled` = `true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**cluster_synchronization_retry**
Parameter type	Positive integer
Description	Number of "retry" for cluster synchronization attempts Number of attempts to synchronize cluster resources when starting up the Transfer Server.
Shipped value	`5`
Parameter taken into account only if:	`clustering_enabled` = `true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**cluster_synchronization_timewait_ms**
Parameter type	Number of milliseconds
Description	Time to wait between two "retry" for synchronization attempts Use this parameter to specify the waiting time in milliseconds between two synchronization attempts on a cluster node after a failure.
Shipped value	`2500`
Parameter taken into account only if:	`clustering_enabled` = `true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**cluster_synchronization_time_processing**
Parameter type	Number of seconds
Description	Max time to wait when processing the cluster synchronization before getting an error Maximum working time between the connection to a node and the end of the synchronization step with this node.
Shipped value	`30`
Parameter taken into account only if:	`clustering_enabled` = `true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**cluster_resource_get_timeout**
Parameter type	Number of seconds
Description	Timeout for getting a resource inside the cluster When this timeout is reached, the operation requesting the resource (for example, the archiving request) will fail
Shipped value	`60`
Parameter taken into account only if:	`clustering_enabled` = `true`

Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**cluster_submit_timeout**
Parameter type	Number of seconds
Description	Max time to wait for a wakeup message Maximal time the Arcsys Transfer Server waits for an acknowledgment message after submiting a message on the cluster.
Shipped value	`20`
Parameter taken into account only if:	`clustering_enabled` = `true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferServer.conf`	**cluster_submit_retry**
Parameter type	Positive integer
Description	Number of "retry" to submit a wakeup message before getting an error Number of times the Arcsys Transfer Server tries again to submit a message on the cluster before getting an error.
Shipped value	`5`
Parameter taken into account only if:	`clustering_enabled` = `true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

5.2.8. Encryption parameters

The following parameter is necessary to define the list of key managers available to the Transfer Server.

`ArcCrypt`: You must have this option to use these parameters.

Parameter name in `transferServer.conf`	**key_managers_conf**
Parameter type	File path
Description	Key managers configuration file for encryption. Absolute file path. Use Unix like separators ('/'), even on Windows. Example(s): key_managers_conf=/opt/Infotel/ArcsysTransferServer/etc/key_managers.conf
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

5.2.9. Code set

The following parameters are only taken into account in Windows environments.

Section of parameters taken into account only if: `OS` = `windows`

Parameter name in `transferServer.conf`	**local_charset**
Parameter type	Characters string
Description	Local system codeset Factor indicating the local codeset of the character strings of the platform containing the Arcsys Transfer Server. This factor is required to convert the character strings used by the Arcsys Transfer Server on Windows to the local codeset. To display the system locale used, type "chcp" in DOS.
Shipped value	`850`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**daemon_charset**
Parameter type	Characters string
Description	Daemon char strings used codeset - usually UTF-8 Factor indicating the codeset to which the character strings transmitted by the network should be converted.

Shipped value	`UTF-8`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

5.2.10. Alerts

To monitor Arcsys operation

Parameter name in `transferServer.conf`	**notify_conf**
Parameter type	Characters string
Description	Alert configuration file path. Absolute file path. Use Unix like separators ('/'), even on Windows. To monitor Arcsys operation, the various modules can be configured to issue alerts to a monitoring server (Centreon, Nagios, etc.) in the event of error or a change in status.
Shipped value	`${TARGET_FOLDER}/ArcsysTransferServer/etc/notify.conf`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

5.2.11. Post processing

An optional post-processing can be done to a file during synchronous retrieval.

Parameter name in `transferServer.conf`	**online_restore_post_processing**
Parameter type	List of string separated by commas. Possible values: `NONE`, `MergePDFLayers`
Description	List of online restore post tasks Sorted list of the post-processing to be done to the files during synchronous retrieval. This parameter is optional. When the field is MergePDFLayers, the application can be asked using an Arcsys API method to "layer" a PDF file. These layers are found in the layer directory of the Arcsys Transfer Server.
Shipped value	`NONE`

Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferServer.conf`	**online_restore_post_processing_timeout_sec**
Parameter type	Number of seconds
Description	Timeout for the post processing task. Defines the maximum time lapse during which post-processing must be performed. Post-processing is abandoned when the time is over (a post-processing error is returned via the Arcsys API). The child process responsible for post-processing is then killed by the Arcsys Transfer Server.
Shipped value	`300`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

5.3. etc/<key_managers>.conf File

5.3.1. Definition

This file is used for encryption purposes. It defines the list of key managers available to encrypt or decrypt master keys by this transfer server. The encryption policies, defined at collection level, are attached to the key managers. Each key manager is uniquely identified by its own code. This configuration file is used for two distinct goals:

- Define the association between these key managers and the transfer servers. This association is done through a shared library and may be configured differently (PIN code, slot id...) on the same key manager on two different transfer servers.
- Define the label of these key managers. The label is used to identify the key managers in the GUI screen of encryption policy.

The transfer server loads the configuration file and compares it to the database:

- if a key manager is defined in the file and does not exist in the database (search by code), it is automatically created in the database and attached to this transfer server;
- if a key manager is defined in the file and already exists in the database: the label of the key manager in the database is updated with the label defined in the configuration file;

- if a key manager is defined in the database for this transfer server and does not exist anymore in the configuration file, an error is reported in the log file of the transfer server.

Important

All transfer servers associated to a media used by a collection with an encryption policy using a given key manager must be attached to this key manager.

Each key manager is configured by a section named by the code given to the key manager:

```
[Code of a key manager]
...list of parameters defining the key manager...
[Code of another key manager]
...list of parameters defining the key manager...
```

5.3.2. Parameters for all key managers

The following settings apply whatever the key manager type:

Parameter name	**label**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Label of the key manager. The Encryption policy screen allows to define an attached key manager through a list of labels of key managers. Each time a transfer server starts and encounters a key manager with a given code, the label read in this parameter overwrites the previously registered label for this key manager. If several transfer servers are attached to the same key manager code with a different label, the label of the key manager will be the one defined in the last started transfer server.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcCrypt Option

Parameter name	**type**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Type of the key manager.
Possible values	`PKCS11`, `PKCS12`
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcCrypt Option

Parameter name	**out_of_validity_range_behaviour**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	This parameter defines the date validation behavior of keys used for encryption. The date validation is processed when loading the keys from the key manager and when the Arcsys Transfer Server uses them. When set to `ERROR`, an error is raised when the key start date is not reached or when the key end date is expired. When this error occurs while loading the keys from the key manager, an error message is printed in the log file, and the key is not loaded in the Arcsys Transfer Server memory. As a result, requests that require this key will end in error. When the date validation error occurs when using a key, the associated request ends in error. When set to `WARNING`, the Arcsys Transfer Server prints a warning message in the log file when the key start date is not reached or when the key end date is expired. The key can still be used. When set to `NONE`, the date validation is disabled. **Note** The date validation uses the start date and end date information from the key manager. When those fields are empty or missing, the date validation is ignored. When only one of these two fields is set, the date validation is done only on this field, and is ignored on the other.
Possible values	`ERROR`, `WARNING`, `NONE`
Shipped value	WARNING
Only for Arcsys option	This parameter can only be used with the option: ArcCrypt Option

5.3.3. Parameters for PKCS11 key manager

The following settings apply only to PKCS11 key managers:

Parameter name	**module**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Path to the PKCS11 dynamic library provided by the HSM constructor to access the device.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcCrypt Option
Example	*module*=/usr/local/lib64/opensc-pkcs11.so

Parameter name	**token_label**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	PKCS11 token label name. Either `token_label` or `slot_id` must be defined to identify the token slot to use. If none of these parameters is specified, or if both parameters are specified, an error will occur.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcCrypt Option

Parameter name	**slot_id**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	PKCS11 slot id. Either `token_label` or `slot_id` must be defined to identify the token slot to use. If none of these parameters is specified, or if both parameters are specified, an error will occur.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcCrypt Option

Parameter name	**user_pin**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	PKCS11 user pin.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcCrypt Option

Parameter name	**hybrid_mode**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	This parameter is optional. Defining this parameter to `true` makes the transfer server use the `openssl` library to encrypt the key instead of the PKCS11 device. This is necessary if the PKCS11 device does not support native encryption. The decryption is always done by the PKCS11 device. By default, it is set to `false`.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcCrypt Option

Here is an example of configuration of a PKCS11 key manager:

```
[softhsm_rsa]
label=SoftHSM PKCS#11 RSA Tokens
type=PKCS11
module=/usr/local/lib64/softhsm/libsofthsm2.so
token_label=rsa
user_pin=enc(E4BBE5934925956C)
```

5.3.4. Parameters for PKCS12 key manager

The following settings apply only to PKCS12 key managers:

Parameter name	**keystore**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Path to the PKCS12 file.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcCrypt Option
Example	*keystore*=etc/rsa1.p12

Parameter name	**password**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password to access the PKCS12 keystore.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcCrypt Option

Here is an example of configuration of a PKCS12 key manager:

```
[pkcs12_rsa1]
label=PKCS#12 RSA1
type=PKCS12
keystore=etc/rsa1.p12
password=enc(8944CE07176988D6F30CC021471EBC9A)
```

5.4. LogParam.cfg File for Arcsys Transfer Server

This section describes the configuration of the log files generated by the Arcsys Transfer Server. Arcsys uses a proprietary standard for these components, which is based on the Log4J standard. The `LogParam.cfg` file found in the `etc/` directory of the Arcsys Transfer Server and the Arcsys Transfer Service enables configuration of these logs. Any configuration error is indicated in the output file of the logs. The rows starting with # and empty rows are ignored. The definition of each variable is optional. Where a variable is not defined, a default value will be associated.

Parameter name in `LogParam.cfg`	**LogLevel**
Parameter type	State variable. Possible values: `DEBUG`, `INFO`, `NOTICE`, `WARN`, `ERROR`, `FATAL`
Description	Defines the level of the logs to write. If the level configured is `ERROR`, only `ERROR` or `FATAL` logs are taken into account.
Shipped value	`INFO`
Default value	`NOTICE`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `LogParam.cfg`	**ConsoleAppender**
Parameter type	State variable. Possible values: `yes`, `no`
Description	Output on stdout of the main process
Shipped value	`no`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `LogParam.cfg`	**FileAppender**
Parameter type	State variable. Possible values: `yes`, `no`
Description	Writes logs in a file
Shipped value	`yes`
Default value	`no`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `LogParam.cfg`	**RotateLogDaily**
Parameter type	State variable. Possible values: `yes`, `no`

Description	Indicates whether logs should be rotated each day at midnight
Shipped value	`yes`
Default value	`no`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `LogParam.cfg`	**File**
Parameter type	File path
Description	Defines the name and path of the file where the logs will be recorded. The path can be absolute or relative. If this variable is not defined or the file defined is incorrect, the default log file will be used.
Shipped value	`${TARGET_FOLDER}/ArcsysTransferServer/log/arcsys-transfer-server.log`
Default value	`arcsys-transfer-server.log`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `LogParam.cfg`	**WinEventLogAppender**
Parameter type	State variable. Possible values: `yes`, `no`
Description	Writes logs in a Windows event log
Shipped value	`no`
Parameter taken into account only if:	`OS = Windows`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `LogParam.cfg`	**WinEventLogSource**
Parameter type	Characters string

Description	Defines the source name of the logs to write in the event log.
Shipped value	`ELEMNISCATE`
Parameter taken into account only if:	`WinEventLogAppender` = `yes`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	True, you have to fill this parameter.

Parameter name in `LogParam.cfg`	**layout.ConversionPattern**
Parameter type	Characters string
Description	Defines the content of the format of the logs generated. The different formats allowed are as follows: %t: Call thread identifier; %i: Call process identifier (PID); %p: Priority level of the message to be logged; %m: Message to log; %n: Line skip character; %d: Date and time in ISO 8601 format (e.g. 2000-09-07 14:07:41); %d(format_date): Date and/or time in the format defined by format_date; Authorized expressions to set the date/time format: - yyyy: Year in 4 digits, - yy: Year in 2 digits, - MM: Month in 2 digits, - MMM: Month abbreviation, - MMMM: month full name, - dd: Day of the month in 2 digits (01-31), - D: Number of the day in the year (001-365), - HH: Time in 24 hours (00-23), - K: Time in 12 hours (0-11), - mm: Minutes (00-59), - ss: Seconds (00-59).

	%u: milliseconds; **Important** **Curly brackets cannot be used in the date format because this character finishes the date format.**
Shipped value	`%p %d{dd-MM-YYYY HH:mm:ss},%u [TID:%t PID:%i] %m`
Default value	`%i.%t.%d.%p %m`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

5.5. etc/arcmover/arcMover.conf File

5.5.1. Definition

This file contains the parameters for ArcMover. Its name cannot be changed.

5.5.2. "settings" Section

This section is optional. It contains the parameter settings at global level for ArcMover.

The general format is:

parameter=value

The global parameters at tape library (see page 241, « Global Settings ») and drive levels (see page 247, « Global Settings ») apply as well as the following settings.

These settings are optional.

5.5.2.1. Settings at ArcMover level

These settings are only valid in the `[settings]` section of the ArcMover configuration file.

The following settings apply:

Parameter name	**arcmover_trace_file**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	This parameter allows to specify a log file that will receive operations issued by the Arcsys Transfer Server on the tape library. The log entries affected by this parameter are a formated part of those generated by default by the Arcsys Transfer Server in its log file (see page 229, « LogParam.cfg File for Arcsys Transfer Server »). Unlike the Arcsys Transfer Server log file, which lists all operations processed by the Arcsys Transfer Server and the different stages of these operations (requests, lots and media concerned, etc.), the ArcMover log file only traces operations carried out on the tape library (mounting and dismounting of tapes, reading and writing operations). When this parameter is not set, or empty, no specific log file is generated for ArcMover Tape operations. The operations carried out on the tape library are always part of the Arcsys Transfer Server log file, in a non-formatted pattern that may be subject to change between different versions of Arcsys. **Note** The automatic rotation of this log file is not supported by Arcsys Transfer Server. Although this trace file contains a limited number of entries, you may consider setting up your own rotation mechanism using system tools if its size grows significantly in your environment. See page 83, « Setting up specific traces for ArcMover Tape ».
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcMover Tape Option
Example	*arcmover_trace_file*= <transfer_server_installation_path>/log/arcmover.log

Parameter name	**invalid_space_left_as_error**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	This parameter controls the behavior of the Arcsys Transfer Server when an inconsistency in the space left on a filesystem is detected (when the value set in the Arcsys Database differs from the one in memory). When set to `false`, the Arcsys Transfer Server ignores the inconsistency error and continues to process the request. This behavior allows the Arcsys Transfer Server to continue processing requests, even when the free space on the filesystem in the Arcsys Database differs from the one in memory. When set to `true`, the application considers the inconsistency as an error. As a result, the affected filesystem is locked to prevent any further write operations on it. Additionally, diagnostic information is logged to assist Arcsys technical support in identifying the root cause of the inconsistency.
Shipped value	`false`

Parameter name	**expiration_fs_nb_threads**
Parameter type	Positive integer.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Number of threads used during filesystem expiration processing. Higher values increase filesystem I/O usage.
Minimum value	`1`
Maximum value	`64`
Shipped value	`5`
Parameter available since version (if applicable)	26.2.STS

Parameter name	**expiration_fs_select_max_movers**
Parameter type	Positive integer.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Maximum number of movers (envelopes) selected in a single batch during filesystem expiration processing. `0` means "no limit", unless *`expiration_fs_select_size_max`* is also `0`. If both *`expiration_fs_select_max_movers`* and *`expiration_fs_select_max_size`* are set to `0`, *`expiration_fs_select_max_movers`* is automatically set to `250`.
Shipped value	`250`
Parameter available since version (if applicable)	26.2.STS

Parameter name	**expiration_fs_select_max_size**
Parameter type	Positive integer.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Maximum cumulative size of movers selected in a single batch during filesystem expiration processing. `0` means "no limit", unless *`expiration_fs_select_max_movers`* is also `0`. If both *`expiration_fs_select_max_movers`* and *`expiration_fs_select_max_size`* are set to `0`, *`expiration_fs_select_max_movers`* is automatically set to `250`.
Unit	bytes
Shipped value	`0`
Parameter available since version (if applicable)	26.2.STS

Parameter name	**expiration_fs_select_max_loop**
Parameter type	Positive integer.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Maximum number of batch iterations during filesystem expiration processing. `0` means "no loop limit": all eligible movers (envelopes) will be processed. **Important** **Using `0` may result in a very long operation.**
Shipped value	`0`
Parameter available since version (if applicable)	26.2.STS

5.5.3. "changers" Section

This section is optional. It contains the definition for the different tape libraries (or changers) accessible with ArcMover. If this option is not present, ArcMover will only manage file system type media.

The general format is:

peripheral=configuration_file

For each tape library, a parameter is given a value: the name of the peripheral uniquely identifies the tape library. The configuration file specifies which configuration file to use. The path specified relates to the (i.e. `etc/arcmover/`) directory.

Example: Loading the configuration of two tape libraries:

```
[changers]
MySCSIChanger=MySCSIChanger.conf
```

Note

An optional *fs_changer_name* parameter is also available. It specifies a name to give to the media changer that manages the file systems.

The default value of this setting is `ArcmoverFSChanger`.

Use the value of this setting in the `[changers]` section to specify an initialization file at tape library level, specific to the file system management.

Example of loading the `filesystems.conf` configuration file for file systems using the default value of the `fs_changer_name` parameter:

```
[changers]
ArcmoverFSChanger=filesystems.conf
```

Example of loading the `filesystems.conf` configuration file for file systems by changing the default value of the `fs_changer_name` parameter:

```
[settings]
fs_changer_name=filesystems

[changers]
filesystems=filesystems.conf
```

5.6. etc/arcmover/filesystems.conf file

5.6.1. Definition

This file contains the settings specific to file systems. Its name is the same that is specified in the `arcMover.conf` file for the changer that manages the file systems. See "changers" Section of ArcMover.conf File (Page 236) for more details.

5.6.2. "settings" Section

This section is mandatory. It is used to define the configuration for the file systems.

The general format is:

parameter=value

5.6.2.1. Settings for file systems

Parameter name	**type**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	This setting specifies the changer type. It is mandatory.
Possible values	`fs`
Shipped value	No shipped value.

Parameter name	**mode_sync**
Parameter taken into account only if:	the operating system is Linux
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Defines if the writing of the files is synchronous or not. When set to `true`, `O_SYNC` flag is activated in all `open()` functions. **Note** This is recommended for a better reliability, but could in some configurations cause a slight performance degradation, relative to the size of the envelopes and of the system buffer.
Shipped value	`true`
Parameter available since version (if applicable)	2024.1.LTS

5.7. etc/arcmover/<peripheral>.conf file

5.7.1. Definition

This file contains the settings specific to a tape library. Its name is the same as that specified in the `arcMover.conf` file for the peripheral concerned.

5.7.2. "settings" Section

This section is mandatory. It is used to define the configuration for the tape library.

The general format is:

parameter=value

5.7.2.1. Settings at tape library level

These settings are only valid in the `[settings]` section of the configuration file of the tape library.

The following settings apply:

Parameter name	**type**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	This setting specifies the tape library type. It is mandatory.
Possible values	`scsi`
Shipped value	No shipped value.

Parameter name	**path**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("`/`"), even on Windows.
Parameter change assessment	After module reboot.
Description	This setting specifies the system access path to the changer of the tape library. The parameters *`path`* and *`serial`* are mutually exclusive; one must be specified exclusively, and not both simultaneously.
Shipped value	No shipped value.

Parameter name	**serial**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	This setting specifies the serial of the changer of the tape library. The parameters *`path`* and *`serial`* are mutually exclusive; one must be specified exclusively, and not both simultaneously.
Shipped value	No shipped value.
Parameter available since version (if applicable)	24.1.STS
Example	*`serial`*`=QUANTUMFQC1920006_LLA`

Parameter name	**media_types**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Specifies the file name where media types to use are defined. The filename must be set without its `.dat` extension and must be relative to the directory `${TARGET_FOLDER} (special variable that automatically takes the value of the parameter of the -DTARGET_FOLDER command line)/ArcsysTransferServer/etc/arcmover/media_types` This setting is mandatory.
Shipped value	No shipped value.
Example	In the following example, the mapping file loaded by the Arcsys Transfer Server is `etc/arcmover/mediatypes/L180.dat`: *media_types*=L180

Parameter name	**drives**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	This parameter specifies the drives to configure for the tape library. This is a list of drive codes; each code has a unique name (maximum length: 16 characters per name). It is optional but the Arcsys Transfer Server cannot use the tape library if no drive is configured. The unique name of the drive will be used to define a specific section of the configuration file, which will contain the configuration parameters of the drive in question (see "drive_xxx" section below).
Shipped value	No shipped value.
Example	Example of definition of two drives: *drives*=DR0001,DR0002

Parameter name	**stop_on_init_error**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	This parameter determines whether the Arcsys Transfer Server should stop or not when an error is detected on all or on a specific tape library (depending on where this parameter is set) when initializing. This parameter specifies one or several mandatory tape libraries, and prevents the Arcsys Transfer Server from starting if this hardware material is not reachable. When set to `false` and an error is encountered while initializing a tape library, a message in `WARN` level is added into the log file. The tape library is then set offline and the Arcsys Transfer Server starts. All operations on the tape library fail. When set to `true`, any error detected on the tape library when initializing the Arcsys Transfer Server stops the launching process. If set in `arcmover.conf` file, any error on any tape library stops the launching process. If set in `<peripheral>.conf` file, only the errors on the tape library concerned by this file stop the launching process.
Shipped value	`false`

5.7.2.2. Global Settings

The global settings at drive level (see List of global settings available at drive configuration level) apply as well as the following settings that have a global impact. Defined at the tape library level, they override all previous global definitions.

These settings are optional.

Settings for All Tape Library Types

Parameter name	**compression_buffer_size_bytes**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Size of compression buffer The specified value must be between `1024` and `523264`.
Shipped value	`131072`
Only for Arcsys option	This parameter can only be used with the option: ArcPak Option

Parameter name	**compression_nb_threads**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Setting used to compress envelopes before archiving. It is used to parallelize the compression process using several threads. The specified value must be between `1` and `32`.
Shipped value	`2`
Only for Arcsys option	This parameter can only be used with the option: ArcPak Option

Parameter name	**recycling_select_max_movers**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	This parameter sets a limit on the number of envelopes considered during a single recycling step of a tape. When set to `0`, this limit is ignored, and all envelopes may be processed in one operation (depending on the value of the *`recycling_select_max_size`* parameter). When set to a non-zero value, the Arcsys Transfer Server processes up to the specified number of envelopes per step, ordered by filemark and offset. The recycling operation is divided into multiple steps, each limited by this parameter's value. The application loops and performs these steps repeatedly until all envelopes have been processed. This parameter can be used alone or with *`recycling_select_max_size`*. When both parameters are used, the recycling processes envelopes until the first of the two limits is reached in each step. The tape is considered empty only when no valid data remains after all recycling steps have been completed.
Shipped value	`0` (ignored)
Only for Arcsys option	This parameter can only be used with the option: ArcMover Tape Option
Parameter available since version (if applicable)	24.1.STS

Parameter name	**recycling_select_max_size**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	This parameter sets a size limit for each recycling step of a tape. When set to `0`, this limit is ignored, and all envelopes may be processed in one operation (depending on the value of the *`recycling_select_max_movers`* parameter). When set to a non-zero value, the Arcsys Transfer Server processes envelopes up to the specified total size per step, ordered by filemark and offset. The recycling operation is divided into multiple steps, each limited by this parameter's value. The application loops and performs these steps repeatedly until all envelopes have been processed. This parameter can be used alone or together with *`recycling_select_max_movers`*. When both parameters are set, the recycling processes envelopes until the first of the two limits is reached in each step. The tape is considered empty only when no valid data remains after all recycling steps have been completed. **Note** This size limit can be exceeded by the size of the last envelope included. The Arcsys Transfer Server selects envelopes until the specified size is reached, and stops after processing the first envelope that exceeds this limit. Please take this into account when choosing the parameter's value.
Unit	bytes
Shipped value	`0` (ignored)
Only for Arcsys option	This parameter can only be used with the option: ArcMover Tape Option
Parameter available since version (if applicable)	24.1.STS

Parameter name	**recycling_select_max_loop**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	This parameter sets a limit for the number of recycling steps of a tape. When set to `0`, this limit is ignored, and all envelopes are processed, in as many steps as necessary. When set to a non-zero value, the Arcsys Transfer Server processes envelopes by steps, with a maximal number of steps as specified by *`recycling_select_max_loop`* parameter. Each step, the Arcsys Transfer Server proceeds envelops up to the limit specified by *`recycling_select_max_movers`* and *`recycling_select_max_size`* parameters.
Shipped value	`0` (ignored)
Only for Arcsys option	This parameter can only be used with the option: ArcMover Tape Option
Parameter available since version (if applicable)	26.2.STS

Settings for SCSI Tape libraries

Parameter name	**scsi_move_timeout**
Parameter type	Number of seconds.
Parameter change assessment	After module reboot.
Description	Maximum time tolerated for tape mount/unmount. Value must be a non-zero positive integer.
Shipped value	`900` (15 minutes)

Parameter name	**scsi_ibm_max_drives**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Maximum number of drives supported for IBM LSM. Value must be a non-zero positive integer.
Shipped value	`255`

Parameter name	**scsi_changer_reload_inventory**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	This parameter allows to enable a quick reload of tape library inventory before loading or unloading a tape. This reload only updates the current slot inventory: there is no detection of new tape or removal of tape. To add or remove a tape from the inventory, use `mailslot_transferServer.sh` script (see page 92, « Managing mailslots »).
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcMover Tape Option
Parameter available since version (if applicable)	2024.1.LTS

5.7.3. "scratchpools" Section

This optional section is used to specify empty media and associated pools.

Only the tapes categorized in the `[scratchpools]` section will be taken into account by ArcMover. A `[scratchpools]` section may be added to `<peripheral>.conf` to control the association between tapes and scratch pools. To use another scratch pool, you must:

- indicate which tapes are to be placed in which pools in the `[scratchpools]` section of the `<peripheral>.conf` file,
- and indicate which scratch pool is used for a given zone in the "zone" section of the Arcsys Web Interface.

The list of media in the Arcsys Web Interface only categorizes tapes when they are used.

Configuration is done by adding a list of lines similar to: `<Tape barcode>=<Scratchpool id>`

Example:

```
[scratchpools]
LAB004L8=1
LAB005L8=1
LAB006L8=1
LAB007L8=101
LAB008L8=101
LAB009L8=2
LAB010L8=2
LAB011L8=55
LAB012L8=55
```

In this example:

- scratch pool id "1" contains the tapes LAB004L8, LAB005L8 and LAB006L8;
- scratch pool id "2" contains the tapes LAB009L8 and LAB010L8;
- scratch pool id "55" contains the tapes LAB011L8 and LAB012L8;
- scratch pool id "101" contains the tapes LAB007L8 and LAB008L8.

5.7.4. "drive_xxx" Section

This section is mandatory for each drive listed in the drive setting in the `[settings]` section. It will have the name "drive" followed by the code chosen in the drive setting in the `[settings]` section.

Example of setup of a tape library with two drives:

```
[settings]
# Tape library parameters
type=scsi
path=/dev/sg5
media_types=SL500
# Define two drives
drives=DR0001,DR0002

[drive_DR0001]
# Parameters for the first drive
type=scsi
path=/dev/sg4

[drive_DR0002]
# Parameters for the second drive
type=scsi
path=/dev/sg3
```

5.7.4.1. Settings at drive level

These settings are only valid in the `[drive_xxx]` section of the tape library configuration file.

Parameter name	**type**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Specifies the drive type. This parameter is optional.
Possible values	`scsi`
Shipped value	`scsi`

Parameter name	**path**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Specifies the system access path to the drive. The parameters *path* and *serial* are mutually exclusive; one must be specified exclusively, and not both simultaneously.
Shipped value	No shipped value.
Example	*path*=/dev/sg4

Parameter name	**serial**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Specifies the serial of the drive. The parameters *path* and *serial* are mutually exclusive; one must be specified exclusively, and not both simultaneously.
Shipped value	No shipped value.
Parameter available since version (if applicable)	24.1.STS
Example	*serial*=80C677F0000

5.7.4.2. Global Settings

All the following settings have a global impact. Defined at drive level, they override all previous global or tape library definitions.

These settings are optional.

Settings for all Drives

Parameter name	**retries_cnt**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Maximum number of unsuccessful attempts before failure of a command to load or unload a tape on a drive.
Shipped value	5

Parameter name	**retry_sleep**
Parameter type	Number of seconds.
Parameter change assessment	After module reboot.
Description	Timeout before a new command attempt to load or unload a tape on a drive. Value must be a positive integer.
Shipped value	`10`

Settings for SCSI Drives

Parameter name	**scsi_cmd_timeout**
Parameter type	Number of seconds.
Parameter change assessment	After module reboot.
Description	Timeout of SCSI commands. Value must be a positive integer.
Shipped value	`120`

Parameter name	**eject_tape_on_error**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Instructs ArcMover whether or not to eject the drive tape if an error occurs.
Shipped value	`false`

Parameter name	**drive_exclusive_use**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Instructs ArcMover whether or not to lock the drive for exclusive use.
Shipped value	`true`

Parameter name	**scsi_drive_autodismount_time**
Parameter type	Number of seconds.
Parameter change assessment	After module reboot.
Description	When set to a value other than `0`, this parameter specifies the idle timeout (in seconds) beyond which a tape is unmounted from the drive. When set to `0`, automatic unmount of tape from a drive is disabled. The tapes are only unloaded by the Arcsys Transfer Server when a job planned on the drive requires another tape than the one currently inserted in it.
Shipped value	`0` (disabled)
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**scsi_force_disable_compression**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	When set to `true` and hardware compression is set on a tape, the Arcsys Transfer Server forces the drive to disable hardware compression. This is done just after loading the tape.
Shipped value	`true`
Parameter available since version (if applicable)	2024.1.LTS

5.8. etc/cloud/s3/<configuration_name>/cloud.conf File

This file is mandatory for each S3 Cloud server and contains the access parameters to the media manager.

5.8.1. Parameters

The following settings apply:

Parameter name	**server**
Parameter type	Characters string.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Cloud server REST API address. For AWS S3, this address must correspond to a valid termination point for the bucket region. For more information, see: http://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region **Important** **If this parameter is missing or left blank, the associated configuration is considered inactive.**
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option

Parameter name	**port_number**
Parameter type	Positive integer.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Cloud server REST API port number.
Shipped value	`80` if `https_enabled` equals to `false` ; otherwise: `443`.
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option

Parameter name	**region**
Parameter type	Characters string.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Bucket hosting area.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option
Example	*region*=us-east-1

Parameter name	**access_key**
Parameter type	Characters string.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Access key ID to the REST API.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option
Example	*access_key*=AKIAIOSFODNN7EXAMPLE

Parameter name	**private_key**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Secret access key to access the cloud provider.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option

Parameter name	**https_enabled**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Specifies whether the REST API of the Cloud media manager should be contacted by https or not.
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option

Parameter name	**certificate_directory**
Parameter type	Characters string.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Directory containing certificates of certification authorities.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option

Parameter name	**proxy**
Parameter type	Characters string.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Proxy access URL, if the access to the Cloud media manager is necessary.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option
Example	*proxy*=127.0.0.1:3128

Parameter name	**proxy_user**
Parameter type	Characters string.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	User used to connect to the proxy, if necessary to access the Cloud media manager.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option

Parameter name	**proxy_passwd**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	User password for proxy login, if necessary to access the Cloud media manager.
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option

Parameter name	**sse_type**
Parameter type	Characters string.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Type of server-side encryption that will be used at request level. By default, this parameter is set to `NONE`. In this case, during S3 API write operations, Arcsys does not set any encryption-specific header. The default setting of the bucket is used. When the parameter is set to `SSE-S3`, Arcsys adds a `x-amz-server-side-encryption` header set to `AES256`. **Note** If you want to use bucket-side encryption features: Since 2023, AWS enforces `SSE-S3` as the default data encryption. If the header is not specified, AWS applies SSE-S3 automatically. This makes the change transparent for Arcsys on AWS. For other compatible cloud providers, this behavior may differ. We recommend the following: • Check with your provider the default behavior when the header is missing, to verify the default value and confirm that missing header is not considered as a request to store data without encryption, even if encryption is enabled on the bucket. • If in doubt, set *`sse_type`* to `SSE-S3` to guarantee encryption.
Possible values	`NONE`, `SSE-S3`
Shipped value	`NONE`
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option

Parameter name	**request_type**
Parameter type	Characters string.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Type of request used to contact the S3 cloud provider. When set to `PATH`, the URL is formed like: `http[s]://<server>:<port_number>/<bucket>/<object>`. When set to `VIRTUAL_HOSTED`, the URL is formed like: `http[s]://<bucket>.<server>:<port_number>/<object>`. **Important** **When using virtual hosted–style buckets with SSL, the SSL wild-card certificate only matches buckets that do not contain dots (".").** **When using buckets with dots, you must either set *`https_enable`*=`false` or *`request_type`*=`PATH`.**
Possible values	`PATH`, `VIRTUAL_HOSTED`
Shipped value	`PATH`
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**versioning_enabled**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Specifies whether versioning features are supported by the cloud provider. The *`versioning_enabled`* parameter allows compatibility with providers that do not support versioning. Setting it to `false` allows to disable the use of versioning features: during archival, the Arcsys Transfer Server does not add the "x-amz-checksum-sha256" header. During disposal, Arcsys performs a simple document deletion instead of deleting each version. If there is a mismatch between the value of *`versioning_enabled`* and the configuration of the accessed bucket, the Arcsys Transfer Server raises an error. These errors only affect operations that use versioning features, namely upload or deletion of a document (for example: archiving requests, the archiving part of copy or migration requests, disposal requests...). **Note** The control of the consistency rule between the bucket's reported versioning state and the *`versioning_enabled`* parameter can be turned off by setting *`check_versioning_consistency`* to `false`. **Note** When Object Lock is configured on the bucket, *`versioning_enabled`* must be set to `true` (as "x-amz-checksum-sha256" header is required).
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option
Parameter available since version (if applicable)	25.3.STS

Parameter name	**check_versioning_consistency**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	Enforces the consistency rule between the bucket's reported versioning state and the *`versioning_enabled`* parameter. This parameter controls the consistency check between the bucket's versioning state and the *`versioning_enabled`* parameter. When *`check_versioning_consistency`*=`true`, the Arcsys Transfer Server raises an error in these cases: • *`versioning_enabled`*=`true` and the storage service does not support versioning. • *`versioning_enabled`*=`false` and the bucket has versioning (enabled or suspended). When *`check_versioning_consistency`*=`false`, the Arcsys Transfer Server skips this check. If *`versioning_enabled`*=`false` but the bucket has versioning set, the Arcsys Transfer Server only creates a delete marker for the object. The physical data is not removed. This behavior is useful when you want Arcsys to avoid physical deletion and manage physical deletion of the objects by yourself. **Note** Setting *`check_versioning_consistency`*=`false` and *`versioning_enabled`*=`false` when bucket versioning is active creates a mismatch between the bucket and Arcsys. If you accept it, you must handle the physical cleanup yourself, for example with a bucket lifecycle policy. Without this, storage costs and unused space may increase, due to the fact that Arcsys will not physically delete the files.
Shipped value	`true`
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option
Parameter available since version (if applicable)	26.1.STS

Parameter name	**cloud_trace_file**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot or using `reload_transferServer` script (see the section reload_transferServer.sh (or .bat) Script).
Description	This parameter allows to specify a trace file that will receive operations issued by the Arcsys Transfer Server on the cloud media manager. The path may correspond to a file that does not exist. In this case, it is created at the first operation on the cloud media manager. The path can be a relative path. In this case, it is relative to the Arcsys Transfer Server installation directory. If the parameter is not set or empty When this parameter is not set, or empty, specific traces are disabled for this cloud media manager. The operations carried out by this cloud media manager are always part of the Arcsys Transfer Server log file, in a non-formatted pattern that may be subject to change between different versions of Arcsys. **Note** The automatic rotation of this log file is not supported by Arcsys Transfer Server. Although this trace file contains a limited number of entries, you may consider setting up your own rotation mechanism using system tools if its size grows significantly in your environment. See page 75, « Traces for Cloud media manager (ArcMover S3 Option) ».
Shipped value	No shipped value.
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option
Parameter available since version (if applicable)	26.2.STS
Example	`cloud_trace_file=log/cloud.log`

Parameter name	**cloud_trace_level**
Parameter taken into account only if:	*cloud_trace_file* parameter is set
Parameter type	Characters string.
Parameter change assessment	After module reboot or using reload_transferServer script (see the section reload_transferServer.sh (or .bat) Script).
Description	Defines the trace level written to the cloud trace file (set by *cloud_trace_file* parameter). STANDARD enables a trace level intended for operational monitoring. The Arcsys Transfer Server records the main processing steps and successful write operations. The generated traces can also be used to collect statistics and calculate average performance indicators over large numbers of operations. VERBOSE provides the most detailed trace output. Use it when investigating performance issues or when additional diagnostic information is required. Due to the large volume of generated traces, this level should only be enabled for troubleshooting purposes and for a limited period. For more information of the traces generated at each level, see page 76, « Format of traces ».
Possible values	STANDARD, VERBOSE
Shipped value	STANDARD
Only for Arcsys option	This parameter can only be used with the option: ArcMover S3 Option
Parameter available since version (if applicable)	26.2.STS

5.9. etc/generic/<configuration_name>/generic.conf File

This file is mandatory for each Generic media manager.

5.9.1. Presentation

The generic media manager connects Arcsys and the storage. It allows to store the envelopes by adding a scripting part around the storage. It can also be used to access any media manager, as soon as this media manager can be accessed in command lines.

The Generic media manager works with Arcsys envelopes, and not directly on Arcsys objects (excepted for native objects).

To use the Generic external media manager:

1. the Arcsys Transfer Server uses a temporary directory to write to or read from. This directory is configured by the *store_dir* parameter.

2. two variables can be used in the scripts:

- the temporary directory, configured in *store_dir* parameter. It is referenced by $STORE_DIR$ in the command to be run.
- the name of the Arcsys envelope. It is referenced by $SFILENAME$ in the command to be run. This filename is of the form lot_<lotid>_seq_<seqid>.<zonename>

3. when the Arcsys Transfer Server writes data to the media manager, the Generic media manager allows to execute a command. This command is specified in the *write_cmd* parameter. The expected return code can be specified in the *write_cmd_success* parameter.
4. the same principle applies for reading or deleting files.

5.9.2. Examples

5.9.2.1. Store the envelopes on a local filesystem and generate a file containing their hash value

This example shows how to store the Arcsys envelopes on a filesystem, and generate a file containing their hash value.

When the envelopes are written to the Generic media manager, the Arcsys Transfer Server stores them in the temporary directory (set by *store_dir* parameter). It then executes the command configured in the *write_cmd* parameter. In the present case, this command computes the MD5 hash value of the envelope, and stores this value in an output file. This file is located in the same directory as the envelope and has the same name with .md5 extension.

The success of the hash generation is checked by comparing its return code to the expected return code in case of success. The expected return code is set in the *write_cmd_success* parameter. If the code returned by the *write_cmd* command differs, the Arcsys Transfer Server sets the Arcsys request in error.

When the hash generation succeeds, the *write_cmd_delete_file* parameters indicates to the Arcsys Transfer Server whether to delete the files in the *store_dir* directory. In the present case, the files must be kept.

Nothing particular has to be done when reading the file. Therefore, all the *read_cmd** parameters are left commented.

When deleting an envelope from the Generic media manager, the associated .md5 file must also be deleted. This is specified by the *delete_cmd* parameter. The success of this command is verified by comparing its return code to the *delete_cmd_success* parameter.

The configuration file for the present case is:

```
store_dir=/opt/Infotel/ExternalStore/

write_cmd=md5sum $STORE_DIR$/$FILENAME$ > $STORE_DIR$/$FILENAME$.md5
write_cmd_success=0
write_cmd_delete_file=false

#read_cmd=
#read_cmd_success=0
#read_cmd_delete_file=false

delete_cmd=rm $STORE_DIR$/$FILENAME$.md5
delete_cmd_success=0
```

5.9.2.2. Transfer the envelopes to a distant host

This example shows how to use the Generic media manager to store the data on a remote host, that does not have mounting point on the server hosting Arcsys.

In the present case, when writing on the media, the *write_cmd* transfers the envelope to a distant host. If the *write_cmd* returns the same code as the one specified in *write_cmd_success* parameter, the operation succeeds. In this case, the Arcsys Transfer Server must delete the files from the *store_dir* temporary directory. This is specified by the parameter *write_cmd_delete_file*, set to `true`.

Before Arcsys Transfer Server can read the file, this one has to be retrieved in the *store_dir* temporary directory. This operation is processed by the *read_cmd* parameter. Its success is verified by comparing its return code to the expected one, set in *read_cmd_success* parameter. Once the Arcsys Transfer Server has read the data, the *read_cmd_delete_file* parameter specifies whether the envelopes must be kept in the *store_dir* temporary directory or not. In the present case, they are deleted (*read_cmd_delete_file* is set to `true`).

When deleting an envelope from the Generic media manager, the distant file must be deleted. This is specified by the *delete_cmd* parameter. The successful result of the delete command is verified by comparing its return code to the *delete_cmd_success* parameter.

The configuration file for the present case is:

```
store_dir=/opt/Infotel/ExternalStore/

write_cmd=scp $STORE_DIR$/$FILENAME$ arcsys@localhost:~/MyStore/$FILENAME$
write_cmd_success=0
write_cmd_delete_file=true

read_cmd=scp arcsys@localhost:~/MyStore/$FILENAME$ $STORE_DIR$
read_cmd_success=0
read_cmd_delete_file=true
delete_cmd=ssh arcsys@localhost "rm -f ~/MyStore/$FILENAME$"
delete_cmd_success=0
```

5.9.2.3. Use Hadoop Distributed File System (HDFS)

This example shows how to use the Generic media manager to access Hadoop Distributed File System (HDFS). HDFS is not supported by Arcsys but can be accessed by command lines, and therefore with the Generic media manager of Arcsys.

In the present case, when writing on the media, the *write_cmd* is used to submit a `put` request of the envelope to the HDFS server. If the *write_cmd* returns the same code as the one specified in *write_cmd_success* parameter, the operation succeeds. In this case, the Arcsys Transfer Server must delete the files from the *store_dir* temporary directory. This is specified by the parameter *write_cmd_delete_file*, set to `true`.

Before Arcsys Transfer Server can access the file for reading, this latter has to be retrieved in the *store_dir* temporary directory. This operation is processed by the *read_cmd* parameter, which submits a `get` request of the envelope to the HDFS server. Its success is verified by comparing its return code to the expected one, set in *read_cmd_success* parameter. Once the Arcsys Transfer Server has read the data, the *read_cmd_delete_file* parameter specifies whether the envelopes must be kept in the *store_dir* temporary directory or not. In the present case, they are deleted (*read_cmd_delete_file* is set to `true`).

When deleting an envelope from the Generic media manager, the corresponding file must be deleted on HDFS. This is performed by the *delete_cmd* parameter. The successful result of the delete command is verified by comparing its return code to the *delete_cmd_success* parameter.

The configuration file for the present case is:

```
store_dir=/opt/Infotel/ExternalStore/

# https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.
html

write_cmd=hdfs dfs -put $STORE_DIR$/$FILENAME$ hdfs://nn.example.com/user/hadoop/$FILENAME$
write_cmd_success=0
write_cmd_delete_file=true

read_cmnd=hdfs dfs -get hdfs://nn.example.com/user/hadoop/$FILENAME$ $STORE_DIR$/$FILENAME$
read_cmd_success=0
read_cmd_delete_file=true

delete_cmd=hdfs dfs -rm hdfs://nn.example.com/user/hadoop/$FILENAME$
delete_cmd_success=0
```

5.9.3. Parameters

The following settings apply:

Parameter name	**store_dir**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Folder where the archived envelopes will be written for the generic media manager. **Important** **If this parameter is missing or left blank, the associated configuration is considered inactive.**
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**write_cmd**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Optional command executed by the media manager after the Arcsys Transfer Server has written an envelope in the *store_dir* folder (for example during an archiving request or during the writing step of a migration, recovery,...).
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**write_cmd_success**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	This parameter specifies the expected return code of the *write_cmd* command. When the return code has a different value than the one defined, the operation is considered in error.
Shipped value	`0`
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**write_cmd_delete_file**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	If *`write_cmd`* is a success, the envelope will be removed from *`store_dir`* if *`write_cmd_delete_file`* is set to `true`.
Shipped value	`false`
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**read_cmd**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Optional command executed by the media manager before the Arcsys Transfer Server reads an envelope in the *`store_dir`* folder (for example during a retrieval request or during the reading step of a migration, recovery...).
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**read_cmd_success**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	If *`read_cmd`* returns a different value from the value defined in *`read_cmd_success`*, the operation will be in error.
Shipped value	`0`
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**read_cmd_delete_file**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	If *`read_cmd`* is a success, the envelope will be removed from *`store_dir`* if *`read_cmd_delete_file`* is set to `true`.
Shipped value	`false`
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**delete_cmd**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Optional command executed by the media manager before the Arcsys Transfer Server deletes an envelope in the *store_dir* folder (for example during a end of life, the deletion step of a migration, the deletion following an archival error...).
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**delete_cmd_success**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	If *delete_cmd* returns a different value from the value defined in *delete_cmd_success*, the operation will be in error.
Shipped value	0
Parameter available since version (if applicable)	2024.1.LTS

5.10. Scripts

The Arcsys Transfer Server is started up or stopped through scripts or in the case of Windows, through a Windows service. In the latter, prior setup should be performed for this operation to function correctly.

5.10.1. launch_transferServer.sh (or .bat) Script

This script starts up the Arcsys Transfer Server.

5.10.2. status_transferServer.sh (or .bat) Script

This script sends a return code matching the current status of the Arcsys Transfer Server.

Return code value	Arcsys Transfer Server status	Description
13	STARTED_NOT_WORKING	The server is started but not handling any requests (`launch_transferServer` script launched).
14	STARTED_AND_WORKING	The server is started and handling requests.
15	SHUTTING_DOWN	The server is shutting down. (`shutdown_transferServer` script launched).
16	SHUTDOWN	The server is stopped.

5.10.3. shutdown_transferServer.sh (or .bat) Script

This script stops the Arcsys Transfer Server.

5.10.4. clean_cache_transferServer.sh (or .bat) Script

This script is used to clean cache directory while the transfer server is running. This action removes any lot that is not currently concerned by a request, regardless of the retention period nor the cache retention politic.

Empty sub-directories are also removed.

Note

It is recommended that you run this script during off-peak periods.

Before deleting a file or a directory, the Arcsys Transfer Server checks that it is not in use. To do this, it uses the list of lots currently being processed. This means that this list of lots cannot be modified during the verification step. This has the consequence of slowing down the processing of requests during the execution of the script.

5.10.5. reload_transferServer.sh (or .bat) Script

This script is used to reload specific parts of the Arcsys Transfer Server configuration. It can be used to reload the log configuration, ArcMover offline file systems, the configuration of external media managers, the `notify.conf` configuration file or the general settings of `transferServer.conf` configuration file.

The script is used as follows:

```
reload_transferServer.sh [<sectionName>]
```

The name of the section is optional. If specified, it is possible to indicate only one each time.

The possible sections are:

- `log`: reloads the log configuration given in the file `LogParam.cfg`. When `log` section is specified, there is no rotation of the Arcsys Transfer Server logs. If the rotation of the Arcsys Transfer Server logs is requested, `rotate` section must be used.

  Note

  The updated log configuration is applied only to the system process of the Arcsys Transfer Server. Logs generated by external libraries (the libraries used to access S3 and generic media managers) are not affected.

  To apply log level changes to external libraries, it is necessary to restart the Arcsys Transfer Server.

- `rotate`: reloads the log configuration given in the file `LogParam.cfg` and triggers the rotation of the Arcsys Transfer Server logs (similar to **rotate_logs_transferServer.sh** (or .bat) script).

- `device`: reloads the devices. The reloads of devices performs the following actions:

  - checks the ArcMover file systems marked offline to set them online when needed. This can be used to set online a file system after fixing a technical issue.

  - sets offline the file systems that are no more handled by the Arcsys Transfer Server (after deleting the association of the path with the Arcsys Transfer Server, or after deleting a zone, for example).

  - reloads the lock status of ArcMover Tape tape libraries, drives and online tapes. This can be used after locking or unlocking a tape library, a drive or a tape in the Arcsys Web Agent.

    Note

    The reload of devices only updates the status of started devices. It can not be used to add a new device.

    Reloading is allowed only when the device is not currently in use.

  - reloads the expiration status of ArcMover Tape tapes. This action is recommended when the Arcsys Transfer Server is left running for a long period of time, during which tapes may be approaching their end-of-life.

The **reload_transferServer.sh** script checks the tapes state and marks as expired those whose mount count, access count or lifetime has reached the expiration threshold.

The expiration threshold is 1% for the mount count and lifetime, and disabled (0%) for the access count. Please contact Arcsys technical support if you need to change these values.

Tapes marked as expired are then subject to a media migration request, processed by the Arcsys Engine according to its configuration (see page 310, « Media Migration »).

- `mediamanager`: reloads external media managers configuration. This can be used, for example, after a change in an external media manager configuration, or for adding a new external media manager while the Arcsys Transfer Server is running.
- `keymanager`: reloads encryption keys configuration. This can be used, for example, to add a new encryption key while the Arcsys Transfer Server is running.
- `notify`: reloads the `notify.conf` settings (see Page 47).
- `settings`: reloads the general section of `transferServer.conf` configuration file. The parameters that are reloaded are: `up_interval`, `max_tcp_retry`, `delay_tcp_retry`, `max_rec_retry`, `max_net_retry`, `retry_delay`, `ip_whitelist`, `fork_session_timeout`.

When no section is specified, the script performs all the actions of `log`, `device`, `mediamanager`, `keymanager`, `notify` and `settings` sections. Otherwise, the script only performs the action corresponding to the specified section.

Note

The reload of `device`, `mediamanager` or `keymanager` configuration can only be performed when no request is being processed by the Arcsys Transfer Server.

5.10.6. rotate_logs_transferServer.sh (or .bat) Script

This script triggers the rotation of the Arcsys Transfer Server logs. This rotation can be done as a planned task or a cron task. Remember that the parameter:

```
RotateLogDaily=yes
```

(default value: `no`) at `LogParam.cfg` level enables automatic rotation each day at midnight.

5.10.7. monitor_transferServer.sh (or .bat) Script

This script generates a monitoring report of the Arcsys Transfer Server.

The generated report consists of one or several sections to monitor: activity, opened logs sockets, requests, online and cache areas, ArcMover devices, external media managers, spooler, key managers and master keys, tapecheck operations or state of the cluster.

An additional section includes general information: host name of the machine hosting the Arcsys Transfer Server, Arcsys Transfer Server site, log-level and connections to the API.

The script is used as follows:

```
monitor_transferServer.sh [<sectionName>[,<sectionName>][…]]
```

The name of the section is optional. If specified, it is possible to indicate a list of section names, separated by commas, with no space character.

The possible sections are:

- `activity`: Arcsys Transfer Server activity (pending and ongoing requests, synchronous retrievals being processed)
- `reqmanager`: request manager
- `log`: opened logs sockets
- `zone`: status of online and cache areas
- `keymanager`: key managers and master keys (for ArcCrypt Option)
- `arcmover`: management of ArcMover devices and association of media by zone (for both ArcMover Tape and ArcMover Filesystem). `arcmover` section includes `filesystem` and `tape` ones.
- `filesystem`: management of ArcMover Filesystem devices and association of media by zone
- `tape`: management of ArcMover Tape devices and association of media by zone (for ArcMover Tape Option)
- `tapecalibration`: status of tape calibration requests (for ArcMover Tape Option). The monitor output lists only tape calibration requests in `waiting` or `processing` status. Once completed, tape calibration requests no longer appear in the monitor output.

- `tapecheck`: lists the running tapecheck operations. For an history of the latest tapecheck operations (including ended ones), use `tapecheck_history.sh` script (see page 276, « tapecheck_history.sh Script »).
- `spooler`: status of the spooler (for ArcMover Tape Option)
- `mediamanager`: external media managers and their associated zones
- `cluster`: state of the cluster (for ArcHP Option)
- `database`: status of database connection
- `json`: change the output of the script. Instead of default output, the monitoring report is printed as a JSON file.

If no section is specified, or if only `json` section is specified, the monitor displays information of the sections `activity, reqmanager, log, zone, arcmover, mediamanager, spooler, keymanager, cluster, database` as well as the general information. All information is displayed one following the others. Otherwise, only the specified sections are displayed, as well as the general information.

5.10.8. keymanager_delete.sh (or .bat) Script

This script deletes a key manager that is no more in use.

The script is used as follows:

```
keymanager_delete.sh <keyManagerCode>
```

The key manager code is mandatory. It specifies the code of the key manager to delete from the Arcsys Database.

For a key manager to be deleted, the following conditions must be met:

- the key manager code must exist in the Arcsys Database.
- the key manager must not be used: its configuration must have been disabled in the `<key_managers>.conf` configuration file (either commented or removed) (see page 225, « etc/<key_managers>.conf File »).
- none of the keys stored in the key manager should be used by an encryption policy.

5.10.9. register_transferServer.sh (or .bat) Script

This script registers the Arcsys Transfer Server in the relational database.

It must be launched after installation of the Arcsys Transfer Server in cluster mode, before its first launch or when configuration on Arcsys Transfer Server is modified to switch from Arcsys Transfer Server to Cluster mode.

Note

To be able to be launched, the API must be started, the relational database must be installed and the site to which the Arcsys Transfer Server belongs must be registered in the relational database.

5.10.10. Specific scripts for ArcMover Tape Option

The specific scripts for ArcMover Tape Option can be found in the `scripts/arcmover` sub-directory of the Arcsys Transfer Server installation directory. They can help you administrate the tape library used with Arcsys.

5.10.10.1. drive_delete.sh Script

This script removes a drive from the Arcsys Database.

The script is used as follows:

```
drive_delete.sh <driveSerial>
```

The drive serial is mandatory. It specifies the serial of the drive to delete from the Arcsys Database.

For a drive to be deleted, the following conditions must be met:

- the serial must exist in the Arcsys Database.
- the deletion must be requested to the Arcsys Transfer Server managing the drive.
- the drive must not be used: its configuration must have been disabled in the `<peripheral>.conf` configuration file (either commented or removed) (see page 238, « etc/arcmover/<peripheral>.conf file »).

If those conditions are met, the drive is deleted from the Arcsys Database.

5.10.10.2. drive_replace.sh Script

This script replaces a drive by another in the Arcsys Database. This impacts the association between the drive and the supported tape types and the association between the drive and its associated zones. The drive to replace is removed from Arcsys Database at the end of the processing.

The script is used as follows:

```
./drive_replace.sh <oldDriveSerial> <newDriveSerial>
```

The drive serials are mandatory. They specify the serial of the drive to replace (`<oldDriveSerial>`) and the serial of the new drive (`<newDriveSerial>`).

For the drive to be replaced, the following conditions must be met:

- the drive to replace and the new drive must exist in the Arcsys Database.
- the drive to replace and the new drive must belong to the same tape library.
- the drive to replace is not used by the Arcsys Transfer Server: its configuration must have been disabled in the `<peripheral>.conf` configuration file (either commented or removed) (see page 238, « etc/arcmover/<peripheral>.conf file »).
- the new drive is known and can be accessed by the Arcsys Transfer Server: it is configured in the `<peripheral>.conf` configuration file (the Arcsys Transfer Server must be re-started for the changes to be taken into account).
- the new drive is not currently in use.

5.10.10.3. flush_spool.sh Script

This script flushes the zones of Arcsys Transfer Server grouped by `Spooler`. When a zone is set to regroup envelopes by `Spooler`, the records are not archived as soon as the request is received by the Arcsys Transfer Server. They will be cached until the actual archiving is triggered, depending on the `Spooler` configuration. The **flush_spool.sh** script allows to force the archiving operation to be triggered (see Arcsys Administration Manual for more information about the `Spooler`).

The **flush_spool.sh** script can either flush all zones grouped by `Spooler` (when launched with no parameter) or a specific list of zones (specified by their identifier, separated by commas).

The script is used as follows:

```
flush_spool.sh [<zoneId>[,<zoneId>][…]]
```

The zone identifier is optional. If specified, it is possible to indicate a list of zone identifiers, separated by commas, with no space character.

An example of **flush_spool.sh** launch would be:

```
flush_spool.sh 41,42,50
```

The flush operation ends current spool transaction(s) (simulate a timeout on the concerned zones) and waits for generated chunks to be processed. This scripts only

deals with archive request launched before the call to **flush_spool.sh** script. All archive request launched after the script will be ignored.

This script returns value `0` when no error occurs. It may return an error in the following cases:

- error on the specified parameters:
  - if one of the zone identifier is not a number: the script returns value `1`.
  - if one of the zone identifier specifiers does not correspond to a zone: the script returns value `2`. However, the script goes on flushing the other zones specified. When returning value `2`, you may check the Arcsys Transfer Server log file to know exactly which zone(s) have failed.
- if a zone is not grouped by `Spooler`, or if no transaction is started for this zone, it is skipped and the flush process goes on processing the next zone and no error is raised.
- if two flush requests are asked at the same time: the second request will return value `3` to indicate that another flush is in progress. Only one request can be processed simultaneously.
- on any other error, the script will return value `4` to indicate that an error occurred.

5.10.10.4. mailslot.sh Script

This script manages the mail slot of a tape library. See the section Managing mail slots (ArcMover Tape Option).

5.10.10.5. tape_calibration.sh Script

Starting with LTO-9 generation of media, it is required to calibrate the media prior to a first use. The calibration process can take up to 2 hours to complete, and thus impacts the archiving performance. It is therefore advisable to perform calibration outside peak periods.

The **tape_calibration.sh** script is used to request a tape calibration. It can be used to prepare the tape calibration when the Arcsys Transfer Server is in period of low activity.

The script is used as follows:

```
./tape_calibration.sh <tapeLabel> <driveSerial>
```

It takes two mandatory parameters:

- the barcode of the tape you want to calibrate,

- the drive serial of the drive to use.

Important

Tape calibration is occurring at the first load of a new tape. This means that the tape is not yet associated to a zone when the script is launched. This implies that it is mandatory to specify a drive to be used, since the Arcsys Transfer Server cannot use the association of drives with zones to find a suitable drive for this operation.

5.10.10.6. tape_delete.sh Script

This script removes a tape from the Arcsys Database.

The script is used as follows:

```
tape_delete.sh <tapeLabel>
```

The tape label is mandatory. It specifies the barcode of the tape to delete from the Arcsys Database.

For a tape to be deleted, the following conditions must be met:

- the barcode must exist in the Arcsys Database.
- there must not be any envelopes belonging to an archived lot present on the tape. If this is the case, you must first move the envelopes to another storage medium. Depending on the context, you may consider moving the envelopes by recycling the tape (in case of faulty tape, for example) or by migrating the zone (for example, if the tape library has to be changed). See page 95, « Recycling a tape » for more details on tape recycling.
- the tape must not be currently in use:
  - the tape must not be inserted in a drive.
  - the tape must not be waiting for a drive operation.

If those conditions are met, the tape is deleted from the Arcsys Database.

5.10.10.7. tape_dismount.sh Script

This script unloads a tape from a drive while the Arcsys Transfer Server is started and running.

The script is used as follows:

```
tape_dismount.sh <tapeLabel> <driveSerial>
```

The tape and drive labels are mandatory. They specify the barcode of the tape to dismount and the serial of the drive where this tape is located.

The script calls the Arcsys Transfer Server, which performs the following steps:

- Checks that the tape and the drive exist in the database and are linked to the Arcsys Transfer Server receiving the command.
- Checks that the tape is loaded in the specified drive.
- Checks that there are no active or scheduled operations for the specified tape.
- Unloads the tape. This step may take some time depending on the type of drive and tape library.

If any of the checks fail, the Arcsys Transfer Server returns an error and the tape is not unloaded from the drive.

5.10.10.8. tape_library_delete.sh Script

This script removes a tape library from the Arcsys Database.

The script is used as follows:

```
tape_library_delete.sh <tapeLibrarySerial>
```

The tape library serial is mandatory. It specifies the serial of the tape library to delete from the Arcsys Database.

For a tape library to be deleted, the following conditions must be met:

- the serial must exist in the Arcsys Database.
- the tape library must not be used: its configuration must have been disabled in the `arcMover.conf` configuration file (either commented or removed) (see page 233, « etc/arcmover/arcMover.conf File »).
- no drive is associated to this tape library. If a drive is still present, use `drive_delete.sh` script (see page 269, « drive_delete.sh Script ») to remove it before asking for the tape library deletion.
- there must not be any envelopes belonging to an archived lot present on any of the tapes associated to the tape library. If this is the case, you must first move the envelopes to another storage medium. This can be done by migrating the zone.

If those conditions are met, the tape library is deleted from the Arcsys Database.

5.10.10.9. tape_library_replace.sh Script

This script replaces a tape library by another in the Arcsys Database. This impacts the association between the tapes and the tape library and the association between the drives and the tape library. The tape library to replace is removed from Arcsys Database at the end of the processing.

The script is used as follows:

```
./tape_library_replace.sh <oldTapeLibrarySerial> <newTapeLibrarySerial>
```

The tape library serials are mandatory. They specify the serial of the tape library to replace (`<oldTapeLibrarySerial>`) and the serial of the new tape library (`<newTapeLibrarySerial>`).

For the tape library to be replaced, the following conditions must be met:

- the tape library to replace and the new tape library must exist in the Arcsys Database.
- the tape library to replace is not used by the Arcsys Transfer Server: its configuration must have been disabled in the `arcMover.conf` configuration file (either commented or removed) (see page 233, « etc/arcmover/arcMover.conf File »).

The **tape_library_replace.sh** script:

- associates the drive from the previous tape library to the new tape library,
- removes the scratch tapes associated to the new tape library,
- associates the tapes from the previous tape library to the new tape library,
- removes the previous tape library from Arcsys Database.

5.10.10.10. tapecheck_cancel.sh Script

This script is used to cancel a tapecheck operation (check, verify, integrity or dump).

The script is used as follows:

```
./tapecheck_cancel.sh <jobId>
```

The script is called with a mandatory parameter corresponding to the identifier of the tapecheck operation to cancel. This identifier can be obtained by calling **tapecheck_history.sh** script (see page 276, « tapecheck_history.sh Script »).
The identifier is also supplied on the standard output, when the operation is created with the **tapecheck_struct.sh**, **tapecheck_integrity.sh**, **tapecheck_verify.sh** or **tapecheck_dump.sh** script.

5.10.10.11. tapecheck_dump.sh Script

This script extracts the content of a tape on a drive managed by ArcMover in a specified directory.

The script is used as follows:

```
./tapecheck_dump.sh <tapeLabel> <pathForTheDump> [<filemarksToDump | all> <driveSerial>]
```

It takes two mandatory and two optional parameters:

- the barcode of the tape you want to extract
- the target directory where the server will copy the tape files
- optionally, the filemark or range of filemarks to be dumped:
  - `all`: Extracts all filemarks.
  - `n`: Extracts only filemark n.
  - `n1-n3`: Extracts the range of filemarks n1, n2, n3.
  - `n1,n3`: Extracts the filemarks n1 and n3.

  If no filemark is specified, all filemarks will be extracted.
- optionally, the drive serial of the drive to use. If you want to specify a drive to use, you must specified the filemark or range of filemarks to dump.

  If no drive serial is set, the Arcsys Transfer Server automatically selects a drive, taking into account various parameters such as drives activity or tape location.

Note

For more information on how to use the **tapecheck_dump.sh** script, see page 85, « Retrieving Expired Data ».

An example of its launch would be:

```
tapecheck_dump.sh 000001L8 /home/ArchivingProduct/extracttape 741,742,750-753
```

The following script enables the filemarks `741, 742, 750, 751, 752, 753` on tape `000001L8` to be extracted in the directory `/home/ArchivingProduct/extracttape`.

The files extracted will be named `tape_<tape label>_file_<filemark>` for the header and inactive or unknown envelopes. Envelopes archived with spooler feature disabled are named `tape_<tape label>_file_<filemark>_lot_<lot identifier>_seq_<sequence`

identifier>. Envelopes archived on a zone grouped by Spooler are grouped in a tar file named tape_<tape label>_file_<filemark>_regroup.tar.

5.10.10.12. tapecheck_history.sh Script

This script is used to show an history of the last 50 tapecheck operations.

The script is used as follows:

```
./tapecheck_history.sh [<tapeLabel>]
```

This script can be called with an optional parameter corresponding to the barcode of the tape for which you want to know the history of tapecheck operations.

5.10.10.13. tapecheck_integrity.sh Script

This script is used to check the integrity of the envelopes on a given tape, and produce a report of the analysis.

The script is used as follows:

```
./tapecheck_integrity.sh <tapeLabel> <outputDirectory> [<driveSerial>]
```

It takes two mandatory and one optional parameters:

- the barcode of the tape you want to extract
- the target directory where the server will log the report of the analysis
- optionally, the drive serial of the drive to use. If no drive serial is set, the Arcsys Transfer Server automatically selects a drive, taking into account various parameters such as drives activity or tape location.

The result of **tapecheck_integrity.sh** is composed of a log file and a csv file, located in the specified target directory.

The log file contains details of the tapecheck operation. In particular, if a filemark found on the tape does not correspond to any record in the database, the first bytes of data are logged. The log file ends with a summary of the operation: duration, number of filemarks and envelopes, used space, number of envelopes found, number of envelopes missing, having an unexpected size or hash value.

The csv file is named tape_<tape label>_env.csv. It contains the summary of the integrity check. The first line of this file contains the header of the columns. Each following line corresponds to a filemark.

The following information is gathered:

- the global status of the envelope (`STATE`). It is `OK` when it corresponds to data expected by the Arcsys Transfer Server. It is `KO` when an error has been detected. It is `N/A` when the Arcsys Transfer Server does not find any associated record in the database.

  The data can be set `KO` if no header is found on the tape, of if the header data does not match what is expected. It can also be in error if the size or hash value of the data does not match what is expected, or if an envelope is missing on the tape.

  When the Arcsys Transfer Server detects an error on an envelope, a check request is created for the lot.

- the filemark concerned (`FILEMARKID`)
- the offset of the envelope after the filemark (`FILEMARKOFFSET`). This offset is different from 0 when the envelopes have been archived on a zone grouped by `Spooler`. It corresponds to the offset of the envelope within the chunk.
- the type of data found (`FILEMARKTYPE`). It can be `H` for Arcsys header, `E` for envelope, `S` for envelopes archived on a zone grouped by `Spooler`, `U` for unknown.
- the size of the data found for this filemark, in bytes (`FILEMARKSIZE`)
- the hash value of the data found at this filemark (`FILEMARKHASH`). This hash value is computed with the hash function of the envelope stored in Arcsys Database.
- fields corresponding to the information of the envelope stored in Arcsys Database:
  - the hash function (`REFHASHFUNC`). This hash function might be `M` for MD5, `S` for SHA1, `T` for SHA512, `U` for SHA256.
  - the identifier of the lot (`REFLOTID`),
  - the envelope number (`REFSEQID`),
  - the name of the envelope (`REFFILENAME`) if known,
  - the type of the envelope (`REFENVTYPE`). It is empty for manifests, header of the tape or unknown data, `0` for tar envelopes, `1` for native NOP-structured AFP file (ArcAFP Option), `2` for native AFP file structured by TLE (ArcAFP Option), `3` for native ZIP file (ArcPak Option), `4` for RAW data.
  - the compress level used (`REFENVCOMPLEVEL`),
  - the size of the envelope or data, in bytes (`REFENVSIZE`),
  - the hash value (`REFENVHASH`),
  - the identifier of the envelope in the database (`REFENVID`).

A typical line of the report will thus be, for an active envelope:

```
OK,1,0,E,2099,20129cb35e6899f73f1359c788ba38db1bea366c1ce9e086043e2381babce7311380e2ae0985
b9e24cb16e14b64a40fcd9c0917650348882ec504518ae058913,T,255,1,lot_255_seq_1,,0,2099,20129cb
35e6899f73f1359c788ba38db1bea366c1ce9e086043e2381babce7311380e2ae0985b9e24cb16e14b64a40fcd
9c0917650348882ec504518ae058913,421
```

for an inactive envelope:

```
N/A,2,0,U,3093,,,,,,,,,
```

and for an envelope in error:

```
KO,8,0,E,2099,cdb1c3e71a2949ac073eed0dc51ee766b2774bdeb64fcacd08c591a6b2542da7794c1f5a9d44
9e884f9d934595312b56fab13e54a510367aca09bbfde3a0d276,T,259,1,lot_259_seq_1,,0,2099,cdb1c3e
71a51138b5aade164631ee766b2774bdeb64fcacd08c591a6b2542da7794c1f5a9d449e884f9d934595312b56f
ab13e54a510367aca09bbfde3a0d276,427
```

5.10.10.14. tapecheck_struct.sh Script

This script is used to list the filemarks present on a tape. If a match is found in the database between a filemark and an envelope or a spool, this match is indicated in the script output.

The script is used as follows:

```
./tapecheck_struct.sh <tapeLabel> <outputDirectory> [<driveSerial>]
```

It takes two mandatory and one optional parameters:

- the barcode of the tape you want to extract
- the target directory where the server will log the tape structure
- optionally, the drive serial of the drive to use. If no drive serial is set, the Arcsys Transfer Server automatically selects a drive, taking into account various parameters such as drives activity or tape location.

The result of **tapecheck_struct.sh** is saved in a log file, located in the specified target directory. This log file lists the size of data (in bytes) for each filemark of the tape.

5.10.10.15. tapecheck_verify.sh Script

This script is used to check the integrity of the envelopes and of objects inside the envelopes, and produce a report of the analysis. The Arcsys Transfer Server checks for objects inside the envelopes when the envelope is neither compressed nor encrypted. In this case (uncompressed and unencrypted data), the content of envelopes archived on a zone grouped by `Spooler` is checked. The check of the content of native ZIP files and native AFP files is not supported.

The script is used as follows:

```
./tapecheck_verify.sh <tapeLabel> <outputDirectory> [<driveSerial>]
```

It takes two mandatory and one optional parameters:

- the barcode of the tape you want to extract
- the target directory where the server will log the report of the analysis
- optionally, the drive serial of the drive to use.

  If no drive serial is set, the Arcsys Transfer Server automatically selects a drive, taking into account various parameters such as drives activity or tape location.

The result of **tapecheck_verify.sh** is composed of a log file and two csv files, located in the specified target directory.

The first csv file is named `tape_<tape label>_env.csv`. It contains the summary of the integrity check on the envelopes. The content of this file is detailed in the **tapecheck_integrity.sh** chapter (see page 276, « tapecheck_integrity.sh Script ».)

The second csv file is named `tape_<tape label>_obj.csv`. It contains the summary of the integrity check on the objects of the envelopes.

The following information is gathered:

- the state of the data found for the filemark (`STATE`). It is `OK` when it corresponds to data expected by the Arcsys Transfer Server. It is `KO` when an error has been detected. It is `N/A` when the Arcsys Transfer Server does not find any associated record in the database.

  When the Arcsys Transfer Server detects an error on an object, a check request is created for the lot.
- the filemark concerned (`FILEMARKID`)
- the offset of the envelope after the filemark (`FILEMARKOFFSET`). This offset is different from 0 when the envelopes have been archived on a zone grouped by `Spooler`. It corresponds to the offset of the envelope within the chunk.
- the identifier of the object in the Arcsys Database (`REFENVID`)
- the size of the object, in bytes (`OBJFILESIZE`)
- the hash value of the object found (`OBJHASH`). This hash value is computed with the hash function of the object stored in Arcsys Database.
- the hash function of the object in the Arcsys Database (`REFHASHFUNC`). This hash function might be `M` for MD5, `S` for SHA1, `T` for SHA512, `U` for SHA256.

- the identifier of the lot containing the object (`REFLOTID`),
- the envelope number containing the object (`REFSEQID`),
- the identifier of the object in the Arcsys Database (`REFOBJID`),
- the name of the object (`REFOBJNAME`),
- the size of the object in the Arcsys Database, in bytes (`REFOBJSIZE`),
- the hash value stored in the Arcsys Database for the object (`REFENVHASH`).

5.10.11. Windows Systems

In Windows, the Arcsys Transfer Server can be run in two ways:

- Windows application
- Windows service

If you wish to start the Arcsys Transfer Server in "Windows Service" mode, you must install the Windows service by starting the **installService_transferServer.cmd** script.

5.10.11.1. installService_transferServer.cmd Script

This script installs the Arcsys Transfer Server as a Windows service. This script is specific to the Windows platform.

For more details, please refer to the Arcsys Installation Manual.

5.10.11.2. removeService_transferServer.cmd Script

This script uninstalls the Arcsys Transfer Server as a Windows service. This script is specific to the Windows platform.

For more details, please refer to the Arcsys Installation Manual.

6. Arcsys Engine

6.1. Overview

The Arcsys Engine is installed in a directory called `ArcsysEngine`.

6.2. Directory Structure

Type	Name	Description
	`attestationpolicy.sh (or .bat)`	Attestation policy management script.
	`formatpolicy.sh (or .bat)`	Format policy management script.
	`launch_engine.sh (or .bat)`	Module launch script.
	`status_engine.sh (or .bat)`	Module status script.
	`shutdown_engine.sh (or .bat)`	Module stop script.
	`start_engine.sh (or .bat)`	Module startup script.
	`encrypt.sh (or .bat)`	Script to encrypt passwords. The password encryption is detailed in the Arcsys Installation Manual.
	`generate-key.sh (or .bat)`	Script to generate symmetric key used to encrypt passwords (with encrypt script). The key generation is detailed in the Arcsys Installation Manual.
	`reload_engine.sh (or .bat)`	Script to reload Arcsys Engine parameters when hot reloading is disabled.
	`workflowpolicy.sh (or .bat)`	Workflow policy management script.
	`journal.sh (or .bat)`	Journal management script.
	`installService_engine.cmd (non systematic)`	Module Windows service installation script.
	`removeService_engine.cmd (non systematic)`	Module Windows service uninstallation script.

	ant	Directory containing a complete version of Ant, which is useful for installation and updates.
	ArcsysDatabase	Directory containing the deployment scripts of the Arcsys Database.
	attestations	Folder defined as the default folder for attestations generation and containing the default attestation policies.
	etc	Directory containing the module configuration files.
	ADMINCOMMAND_params.properties	This file contains the parameters of the Arcsys Engine administration interface.
	arcsysJAAS.conf	This file contains the parameters of the Java authorization and authentication service.
	ENGINE.properties	Main settings file of the module.
	ERROR_message_fr_FR.properties	
	log4j2.xml	log4j configuration file.
	policy	This file contains the parameters of the module's Java class permissions.
	REFERENTIAL_JDBC.properties	
	TEMPLATE_MAIL_Alert_End_ Request_fr_FR.properties	
	TEMPLATE_MAIL_Alert_Lot_fr_ FR.properties	
	TEMPLATE_MAIL_CheckZones_fr_ FR.properties	
	THIRD_PARTY_SEARCH_ENGINE.properties	This file contains the settings of the third party search engine.
	formats	Folder containing the definition of formats for the native envelopes (AFP, ZIP, etc.)

		as well as the default format policies for standard files.
	jar	Directory containing the module Arcsys Java libraries (JAR).
	libext	Directory containing the different external Java libraries required in the module.
	log	Directory containing a list of all the log files. By default, these files are automatically rotated each day.
	mail	Mail directory.
	trace	Directory containing a list of all the trace files. By default, these files are automatically rotated each day.

6.3. etc/log4j2.xml file

6.3.1. Definition

This file contains the settings for the log file. The logs are generated with the log4J Apache library.

6.3.2. Parameters

The parameters of this file depend on the information desired in the logs. Hence please consult the standard documentation of Log4J for any changes.

See: https://logging.apache.org/log4j/2.x/

6.4. etc/REFERENTIAL_JDBC.properties File

6.4.1. Definition

This file contains the parameters for accessing the Arcsys Database in JDBC mode. See its description in page 123, « Configuring JDBC Access to the Relational Database ».

6.5. etc/ADMINCOMMAND_params.properties File

6.5.1. Definition

This file contains the necessary parameters for running the (**status, start, stop**, etc.) commands on the Arcsys Engine through the Arcsys Engine API.

6.5.2. Parameters

ENGINE_API_PORT_NUMBER: Port number on which the Arcsys Engine API is deployed (it must be the same as *COMPONENT_API_PORT_NUMBER* of the `ENGINE.properties` file).

ENGINE_API_SERVICE_NAME: This parameter must contain the API RMI service name of the Arcsys Engine, which is the Arcsys Engine code (therefore it must be the same as *COMPONENT_CODE* of the `ENGINE.properties` file).

ENGINE_API_USERNAME: LDAP user account used to start, stop, or query the engine status. This user must belong to a group with `GRANT_ALL` or `ADMIN` rights.

ENGINE_API_USERPASSWORD: Encrypted User password.

Securing RMI access:

For security parameters *ENGINE_API_RMI_SECURITY_PROTOCOL*, *TRUSTSTORE_FILE*, *TRUSTSTORE_PASSWORD*, *KEYSTORE_FILE* and *KEYSTORE_PASSWORD*, see Configuring Arcsys to Manage SSL on LDAP or RMI.

6.6. ENGINE.properties File

This file contains the parameters for the Arcsys Engine.

6.6.1. Component code and host name parameters

To make the system open, the application agents, engines, transfer servers and transfer services have a unique code that is used to identify them in the configuration files or in the Arcsys interface. This way, the host of a module can be changed without it impacting the other modules. This code is determined by the administrator, who will choose a code with meaning for the context.

Parameter name in `ENGINE.properties`	**COMPONENT_CODE**
Parameter type	Characters string
Description	The code (name) of this component, to avoid duplication. This name must be unique across all sites. The code must identify the module in the installation.

	This code is used as the RMI service name under which the engine is deployed. It is recommended to use a logical name distinct from a physical host name for example. It must not include the following characters: ; , / , \ , " , ' and space.
Shipped value	`ArcsysEnginePrimary`
Parameter name after installation	`COMPONENT_CODE`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `ENGINE.properties`	**COMPONENT_HOST**
Parameter type	Characters string
Description	Optional component host (to force the address of the component). If not empty, this host overrides the host chosen automatically. Here you can set an IP, a host name... This address will be the one used by the other modules. Module host. This parameter is optional. This is used to "force" the name under which the module will be registered in the Arcsys Database, overwriting the default name chosen by Arcsys. Use, for example, a localhost, an IP, a host name, etc.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**COMPONENT_AUTO_REFERENCE_MODE**
Parameter type	State variable. Possible values: `IP`, `HOSTNAME`
Description	Optional mode of referencing: IP or HOSTNAME (Default: HOSTNAME). This parameter is not used if the COMPONENT_HOST is not empty.
Shipped value	`HOSTNAME`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.2. Component API

Parameters for the Arcsys Engine API

Parameter name in `ENGINE.properties`	**COMPONENT_API_PORT_NUMBER**
Parameter type	Positive integer
Description	The port number on which the Engine API is deployed.
Shipped value	`25020`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**COMPONENT_API_RMI_SECURITY_PROTOCOL**
Parameter type	State variable. Possible values: `NONE`, `SSL`
Description	API RMI Protocol (NONE | SSL)
Shipped value	`NONE`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**COMPONENT_API_RMI_NEED_CLIENT_AUTH**
Parameter type	Boolean. Possible values: `true`, `false`
Description	The RMI server accepts only the connections of a client whose certificate is signed by a trust authority.
Shipped value	`false`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.3. Site

Parameters related to the site

Parameter name in `ENGINE.properties`	**SITE_CODE**
Parameter type	Characters string
Description	Code of the site in which this component is installed Site code to which the module must belong. This site must be created in the Web beforehand (by default, the installer creates systematically a `PRIMARY_SITE` site code).

Shipped value	`PRIMARY_SITE`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**ALL_SITES_REQUIRED**
Parameter type	Boolean. Possible values: `true`, `false`
Description	If true, an archiving request will end successfully only if it succeeded in all the target sites. Specifies whether, for an archiving request targeting the zones that belong to several sites, the archiving request is considered successful only if it succeeds on at least the mandatory zones of each site. In this case, no recovery plan will be possible. **Important** **This parameter must have the same value on all the engines.**
Shipped value	`true`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

6.6.4. Paths

Parameters related to paths

Parameter name in `ENGINE.properties`	**ENGINE_INSTALL_FOLDER**
Parameter type	Directory path
Description	Home Installation Folder of the Engine
Shipped value	`${TARGET_FOLDER}/ArcsysEngine`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**PROPERTIES_FILES_PATH**

Parameter type	Directory path
Description	Directory where the configuration files of the Engine are located (absolute or relative)
Shipped value	`${TARGET_FOLDER}/ArcsysEngine/etc`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**FORMAT_FILES_PATH**
Parameter type	Directory path
Description	Directory containing the format handler files used when archiving the native envelopes
Shipped value	`${TARGET_FOLDER}/ArcsysEngine/formats`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.5. Cluster management parameters

The following parameters enable the Engine to operate in cluster mode.

`ArcHP`: You must have this option to use these parameters.

Parameter name in `ENGINE.properties`	**CLUSTER_ENABLED**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Enables the clustering of engines (multicast is used when no clustering configuration file is set) Specifies whether you authorize operation of the engine in cluster mode. This mode means you can have several engines on the same site that share a particular specialization (archiving, retrieval, migration, etc.) in the same time slots. The clustering mechanism is based on the JGroups library (see http://jgroups.org) which is, for instance, used in the JBoss application server. **Important** **All engines of a given site that have `CLUSTER_ENABLED` set to true are in the same cluster. The engines of another site will be in another cluster: a cluster cannot be distributed over several sites. Synchronization is not part of the cluster specializations. Synchronization is a process involving several sites.**

Shipped value	`false`
Option(s)	`ArcHP`: You must have this option to use this parameter.
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**CLUSTER_MCAST_PORT**
Parameter type	Positive integer
Description	Port used for multicast when no clustering configuration file is set This parameter specifies a port that will be used for multicast communication among Engines of the same site. This value must therefore be the same for all Engines of a cluster. This parameter allows you to isolate cluster messages between different clusters, avoiding "discarded message from different cluster" messages in the log files.
Shipped value	`7600`
Parameter taken into account only if:	`CLUSTER_ENABLED` = `true`
Option(s)	`ArcHP`: You must have this option to use this parameter.
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**CLUSTER_CONFIG_FILE**
Parameter type	File path
Description	Optional clustering configuration file (cf. JGroups documentation) Path of the cluster configuration file in JGroups format. The use of a configuration file allows you to use for instance TCP instead of UDP when IP multicast poses a problem (e.g. during WAN use): see an example on http://www.jgroups.org/manual/index.html#_tcp1.
Shipped value	**No shipped value.**
Parameter taken into account only if:	`CLUSTER_ENABLED` = `true`
Option(s)	`ArcHP`: You must have this option to use this parameter.
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.6. Engine internal parameters

Internal parameters for the engine

Parameter name in `ENGINE.properties`	**CTRLPROC_LOOPSLEEPTIME**
Parameter type	Number of milliseconds
Description	Delay between each iteration of the general control process
Shipped value	`15000`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LOGPROC_LOOPSLEEPTIME**
Parameter type	Number of milliseconds
Description	Delay between each iteration of the thread reading data coming from the Transfer Server
Shipped value	`1000`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

6.6.7. Requests

Parameters related to processing of requests

Parameter name in `ENGINE.properties`	**ARCHIVING_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Process archiving requests of the current site? CAUTION: If the ArcHP option is not enabled, one engine per site with this specialization is authorized. If the ArcHP option is enabled, you can have as many engines per site as you want with this specialization. Specifies whether this engine will process archiving requests
Shipped value	**No shipped value.**

Option(s)	`ArcHP`: If the ArcHP option is not enabled, one engine per site with this specialization is authorized. If the ArcHP Option is enabled, you can have as many engines per site as you want with this specialization.
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**RETRIEVAL_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether this engine will process retrieval requests. CAUTION: If the ArcHP option is not enabled, one engine per site with this specialization is authorized. If the ArcHP option is enabled, you can have as many engines per site as you want with this specialization.
Shipped value	**No shipped value.**
Option(s)	`ArcHP`: If the ArcHP option is not enabled, one engine per site with this specialization is authorized. If the ArcHP Option is enabled, you can have as many engines per site as you want with this specialization.
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**RESTITUTION_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether this engine will process archive restitution requests. CAUTION: If the ArcHP option is not enabled, one engine per site with this specialization is authorized. If the ArcHP option is enabled, you can have as many engines per site as you want with this specialization.
Shipped value	**No shipped value.**
Option(s)	`ArcHP`: If the ArcHP option is not enabled, one engine per site with this specialization is authorized. If the ArcHP Option is enabled, you can have as many engines per site as you want with this specialization.
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MIGRATION_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether this engine will manage the migration process (migration of records from one storage zone to another storage zone). CAUTION: This involves both creating and processing these requests on the storage zones belonging to the site of this engine. There must be an engine managing these requests per site where the storage zones are defined. If ArcHP option is not enabled, it is possible to have several engines per site managing these requests, but the time slots must not overlap. If ArcHP option is enabled, it is possible to have several engines per site managing this processing, even with shared time slots.
Shipped value	**No shipped value.**
Option(s)	`ArcHP`: If ArcHP Option is not enabled, it is possible to have several engines per site managing these requests, but the time slots must not overlap. If ArcHP Option is enabled, it is possible to have several engines per site managing this processing, even with shared time slots.
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**CHECKZONES_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether the engine will process zone checking (random checking of envelope integrity). CAUTION: This involves both creating and processing these requests on the storage zones belonging to the site of this engine. There must be an engine managing these requests per site where the storage zones are defined. If ArcHP option is not enabled, it is possible to have several engines per site managing these requests, but the time slots must not overlap. If ArcHP option is enabled, it is possible to have several engines per site managing this processing, even with shared time slots.
Shipped value	**No shipped value.**
Option(s)	`ArcHP`: If ArcHP Option is not enabled, it is possible to have several engines per site managing these requests, but the time slots must not overlap. If ArcHP Option is enabled, it is possible to have several engines per site managing this processing, even with shared time slots.
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**RECOVERY_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether this engine will process recovery (re-establishment of copy showing an error). CAUTION: This involves both creating and processing these requests on the storage zones belonging to the site of this engine. There must be an engine managing these requests per site where the storage zones are defined. If ArcHP option is not enabled, it is possible to have several engines per site managing these requests, but the time slots must not overlap. If ArcHP option is enabled, it is possible to have several engines per site managing this processing, even with shared time slots.
Shipped value	**No shipped value.**
Option(s)	`ArcHP`: If ArcHP Option is not enabled, it is possible to have several engines per site managing these requests, but the time slots must not overlap. If ArcHP Option is enabled, it is possible to have several engines per site managing this processing, even with shared time slots.
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**POSTPONE_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether this engine will process postponed requests (requests created for potentially long operations in the web interface). CAUTION: This means both creating and processing these requests on all storage zones. It is possible to have several engines per site managing these requests but the time slots must not overlap.Even if ArcHP option is enabled, it is forbidden to have several engines managing these requests on shared time slots. **Important** **Even if ArcHP Option is enabled, it is forbidden to have several engines managing these requests on shared time slots**
Shipped value	**No shipped value.**
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).

Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**COPY_REQUESTS_PROCESSING**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether this engine will process the copy requests created by CopyRequestsManager from one zone in a zone list to a zone in another zone list of a storage profile. CAUTION: This involves processing these requests on the storage zones belonging to the site of this engine. There must be an engine managing these requests per site where the storage zones are defined. If ArcHP option is not enabled, it is possible to have several engines per site managing these requests, but the time slots must not overlap. If ArcHP option is enabled, it is possible to have several engines per site managing this processing, even with shared time slots.
Shipped value	`true`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SYNC_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether this engine will process manifest synchronization. CAUTION: This means both creating and processing these requests on all storage zones. It is possible to have several engines per site managing these requests but the time slots must not overlap.Even if ArcHP option is enabled, it is forbidden to have several engines managing these requests on shared time slots. **Important** **Even if ArcHP Option is enabled, it is forbidden to have several engines managing these requests on shared time slots**
Shipped value	**No shipped value.**
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**DIRECT_STREAM_ARCHIVING_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Process direct stream archiving requests of the current site? This parameter is only useful with ArcHSM.
Shipped value	`true`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**DIRECT_STREAM_RETRIEVAL_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Process direct stream retrieval requests of the current site? This parameter is only useful with ArcHSM.
Shipped value	`true`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MEDIA_MIGRATION_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether this engine will process media expiry. CAUTION: This involves both creating and processing these requests on the storage zones belonging to the site of this engine. There must be an engine managing these requests per site where the storage zones are defined. If ArcHP option is not enabled, it is possible to have several engines per site managing these requests, but the time slots must not overlap. If ArcHP option is enabled, it is possible to have several engines per site managing this processing, even with shared time slots.
Shipped value	**No shipped value.**
Option(s)	`ArcMover Tape`: You must have this option to use this parameter. `ArcHP`: If ArcHP Option is not enabled, it is possible to have several engines per site managing these requests, but the time slots must not overlap. If ArcHP Option is enabled, it is possible to have several engines per site managing this processing, even with shared time slots.
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching

	`reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**RECYCLING_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether this engine will process tape recycling. CAUTION: This involves both creating and processing these requests on the storage zones belonging to the site of this engine. There must be an engine managing these requests per site where the storage zones are defined. If ArcHP option is not enabled, it is possible to have several engines per site managing these requests, but the time slots must not overlap. If ArcHP option is enabled, it is possible to have several engines per site managing this processing, even with shared time slots.
Shipped value	**No shipped value.**
Option(s)	`ArcMover Tape`: You must have this option to use this parameter. `ArcHP`: If ArcHP Option is not enabled, it is possible to have several engines per site managing these requests, but the time slots must not overlap. If ArcHP Option is enabled, it is possible to have several engines per site managing this processing, even with shared time slots.
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**DISPOSAL_ENGINE**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether this engine will process disposal by retention schedule. CAUTION: This means both creating and processing these requests on all storage zones. It is possible to have several engines per site managing these requests but the time slots must not overlap.Even if ArcHP option is enabled, it is forbidden to have several engines managing these requests on shared time slots.
Shipped value	**No shipped value.**
Option(s)	`ArcHP`: Even if ArcHP Option is enabled, it is forbidden to have several engines managing these requests on shared time slots
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

6.6.8. Requests processing

Parameters related to requests processing

Parameter name in `ENGINE.properties`	**NB_REQUESTS_CREATED**
Parameter type	Positive integer
Description	`NB_REQUESTS_CREATED` requests (all types together) will be created each `REQUESTS_CREATE_LOOPSLEEPTIME`
Shipped value	`100`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**NB_REQUESTS_LOADED**
Parameter type	Positive integer
Description	`NB_REQUESTS_LOADED requests` (all types together) will be loaded each `REQUESTS_LOAD_LOOPSLEEPTIME` Number of requests processed simultaneously by the Engine.
Shipped value	`20`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**REQUESTS_LOAD_LOOPSLEEPTIME**
Parameter type	Number of milliseconds
Description	Requests load loop sleep time Time between each launching of the requests load procedure. A SQL `SELECT` determining the requests to be processed by this Engine then takes place in the relational database. If this time is increased, requests will be loaded less often by the Engine, and the database will be called on less. If this time is reduced, the Engine will react more quickly when processing a new request.
Shipped value	`500`
Parameter change assessment	After module reboot.

Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**REQUESTS_CREATE_LOOPSLEEPTIME**
Parameter type	Number of milliseconds
Description	Interval between each launch of the automatic request creation process. A shorter interval will result in more frequent automatic requests.
Shipped value	`60000`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEC_RESTPROCEEDED_TIMEOUT**
Parameter type	Number of seconds
Description	Timeout for the status PROCEEDED of retrieval Maximum time allowed for a retrieval or archive restitution request in PROCEEDED status (pending processing by the application agent). After this time lapse, if the request remains in PROCEEDED status, the Engine generates an error code 1003.
Shipped value	`1200`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEC_DIRECTSTRRESTPROCEEDED_TIMEOUT**
Parameter type	Number of seconds
Description	Timeout for the status PROCEEDED of direct stream retrieval Maximum time allowed for a direct stream retrieval request in PROCEEDED status (pending processing by the application agent). After this time lapse, if the request remains in PROCEEDED status, the Engine generates an error code 1003.
Shipped value	`1200`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEC_RESTVALIDATED_TIMEOUT**
Parameter type	Number of seconds
Description	Timeout for the status VALIDATED of retrieval Maximum time allowed for a retrieval or archive restitution request in VALIDATED status (pending processing by the Transfer Server). After this time lapse, if the request remains in PROCEEDED status, the Engine generates an error code 1003.
Shipped value	`600`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEC_BETWEEN_TIMEOUT_QUERIES**
Parameter type	Number of seconds
Description	Time between timeout queries Interval between two timeout confirmation requests for an Engine request to the Transfer Server.
Shipped value	`180`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEC_ENGINE_AVAILABLE_TIMEOUT**
Parameter type	Number of seconds
Description	Timeout duration after which the engine is considered unavailable Interval between two recordings in the relational database of the Engine.
Shipped value	`120`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**ENFORCE_DELETE_DATA**
Parameter type	Boolean. Possible values: `true`, `false`

Description	If "true", when a deletion occurs for an archive deletion (end of life), the success of the deletion is necessary for the request to complete. If "false", the deletion order is sent asynchronously and the success of the deletion is not required for the request to complete. When a deletion error is encountered on an automatic request, a new deletion request is automatically recreated. The deletion might for example fail due to the fact that the media is locked (in Arcsys or physically), or that the communication with the media manager encountered an error. The default value is set to false when updating from a version before 2024.1.LTS, to have the same behaviour as before this parameter was introduced. **Note** In the case where `ENFORCE_DELETE_DATA=false` on ArcMover Disk, ArcsysFsComparator File systems comparator can be used to detect envelopes that should have been deleted and delete them automatically. But on other media, there is no tool available to detect these envelopes. That is why setting `ENFORCE_DELETE_DATA` to `true` is significantly useful on media such as Cloud media (S3 for example).
Shipped value	`true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEC_TIME_INTERVAL**
Parameter type	Number of seconds
Description	The time to wait before executing the next request on a given pool. For example, an automatic request in ERROR won't be re-created until after this delay. A recovery request won't be created until after this delay following an archiving request. If a pool is set to "0 day", a migration request won't be created until after this delay.
Shipped value	`86400`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**REQUESTS_PROCESSING_STOP**

Parameter type	Boolean. Possible values: `true`, `false`
Description	If true is set -> cancels the requests being processed at the end of the time interval. This parameter stops automatic requests that are in progress once the defined creation and processing intervals have elapsed.
Shipped value	`false`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.9. Hash

Parameters related to hash

Parameter name in `ENGINE.properties`	**MANIFEST_TAR_HASH_FUNCTION**
Parameter type	State variable. Possible values: `MD5`, `SHA1`, `SHA256`, `SHA512`, `NONE`
Description	Hash Function for the Manifest and the TAR files Defines the default encoding algorithm for generating the hash value responsible for the integrity of TAR files and the XML manifest. You can opt for no integrity management by selecting the `NONE` value, but this is not recommended.
Shipped value	`SHA512`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.10. Archive

Parameters related to archive

Section of parameters taken into account only if: `ARCHIVING_ENGINE` = `true`

Parameter name in `ENGINE.properties`	**ALL_ZONES_REQUIRED**
Parameter type	Boolean. Possible values: `true`, `false`
Description	If true, an archiving request will end successfully only if it succeeded in all the targets on site. Specifies whether, for an archiving request targeting the zones that belong to several zone lists, the archiving request is only considered as successful if

	it is successful on at least the mandatory zones of each site. In this case, no recovery plan will be possible. **Important** **This parameter must have the same value on all the engines. If false, the notion of pool mandatory is unused**
Shipped value	`true`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEC_ARCHVALIDATED_TIMEOUT**
Parameter type	Number of seconds
Description	Timeout for the status VALIDATED of archiving Maximum time allowed for an archiving request in VALIDATED status (pending processing by the application agent). After this time lapse, if the request remains in VALIDATED status, the Engine generates an error code 1003.
Shipped value	`1200`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEC_ARCHPROCEEDED_TIMEOUT**
Parameter type	Number of seconds
Description	Timeout for the status PROCEEDED of archiving Maximum time allowed for an archiving request in the PROCEEDED status (pending processing by the Arcsys Transfer Server). After this time lapse, if the request remains in PROCEEDED status, the Engine requests the Transfer Server for the status. When the Transfer Server does not handle the request or does not answer, the Engine generates an error code 1004.
Shipped value	`600`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEC_DIRECTSTRARCHPROCEEDED_TIMEOUT**
Parameter type	Number of seconds
Description	Timeout for the status PROCEEDED of direct stream archiving
Shipped value	`600`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**ARCHIVING_FAILURE_BEHAVIOUR**
Parameter type	State variable. Possible values: `KEEP_CORRECT_ENVELOPES`, `DELETE_ALL_ENVELOPES`
Description	Algorithm used when the archiving fails on a pool This parameter configures the behavior of the Engine in the event of incorrect archiving in a zone but that is successful overall (event possible if `ALL_ZONES_REQUIRED` = `false` or `ALL_SITES_REQUIRED` = `false`). According to the behavior chosen, when archiving is successful but contains an error for the zone, the Engine will request the deletion of all envelopes in the zone in error or the deletion of the envelopes with error in the zone in error (envelopes correctly archived in a zone in error will be kept). For recovery in a zone in error, if `KEEP_CORRECT_ENVELOPES` is chosen, recovery will only affect the envelopes with error in archiving. Otherwise, if `DELETE_ALL_ENVELOPES` is chosen, all envelopes of the lot will be taken in recovery.
Shipped value	`DELETE_ALL_ENVELOPES`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.11. Migration

Parameters related to migration

Section of parameters taken into account only if: `MIGRATION_ENGINE` = `true`

Parameter name in `ENGINE.properties`	**SEC_MIGRATION_TIMEOUT**
Parameter type	Number of seconds
Description	Timeout for migration Maximum time allowed for a migration request to complete.

	If a migration request does not change its status within this time frame, it is considered to be in a hang state.
Shipped value	`600`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MIGRATE_CHECK_INTEGRITY**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Check the fixity? Defines whether the integrity of the records should be checked during this operation.
Shipped value	`true`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MIGRATE_DELETE_DATA**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Delete data from initial zone? Defines whether the data should be moved (`true`) or simply copied (`false`) during migration from one storage media to another.
Shipped value	`true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MIGRATION_LAUNCH_INTERVAL**
Parameter type	List of string separated by commas
Description	List of Cron intervals to create migration requests. Format: (<cron_interval>;<cron_interval>) List of intervals in CRON format for launching the creation (except for copy requests) and treatment process, each interval being separated by the ";" character. The paragraph "Managing CRON Format" gives more details about this format.

	Note The first value of the interval corresponds to the seconds and must be set to `*`. If anything else than `*` (for example `0`) is specified for this first value, an issue can occur, as a check is made that the current time matches the CRON expression specified. In the example, if `0` is specified, it would only match when the number of seconds of the current time is equal to 0.
Shipped value	`* * 00-07,20-23 ? * MON-FRI;* * * ? * SAT,SUN`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MAX_MIGRATION_REQUESTS_PER_DAY**
Parameter type	Positive integer
Description	This parameter sets the maximum number of automatic requests of this type that can be created by the Engine per day. The counter is reset to 0 when the Engine starts.
Shipped value	`300`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**CODE_ZONES_SOURCE_LIST_FOR_MIGRATION**
Parameter type	List of string separated by commas
Description	List of source zones code from which migration will be restricted. Migration from other zones will not occur, even for end of life migrations. Optional list of zone codes from which to limit the migration. This is used to (manually) implement a migration plan, for example. Note that even disposals will take place from these zones only.
Shipped value	**No shipped value.**
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

6.6.12. Postpone

Parameters related to postpone

Section of parameters taken into account only if: `POSTPONE_ENGINE` = `true`

Parameter name in `ENGINE.properties`	**SEC_POSTPONE_TIMEOUT**
Parameter type	Number of seconds
Description	Timeout for postponed entity modification/deletion Maximum time allowed for a postponed request to complete. If a postponed request does not change its status within this time frame, it is considered to be in a hang state.
Shipped value	`600`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**POSTPONE_LAUNCH_INTERVAL**
Parameter type	List of string separated by commas
Description	List of Cron intervals to create postponed entity modification/deletion requests. Format: (<cron_interval>;<cron_interval>) List of intervals in CRON format for launching the creation (except for copy requests) and treatment process, each interval being separated by the ";" character. The paragraph "Managing CRON Format" gives more details about this format. **Note** The first value of the interval corresponds to the seconds and must be set to `*`. If anything else than `*` (for example `0`) is specified for this first value, an issue can occur, as a check is made that the current time matches the CRON expression specified. In the example, if `0` is specified, it would only match when the number of seconds of the current time is equal to 0.
Shipped value	`0 0/1 * * * ?`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).

Required	False, this parameter is optional.

6.6.13. Recovery

Parameters related to recovery

Section of parameters taken into account only if: `RECOVERY_ENGINE` = `true`

Parameter name in `ENGINE.properties`	**RECOVERY_CHECK_INTEGRITY**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Check the fixity? Defines whether the integrity of the records should be checked during this operation.
Shipped value	`true`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**RECOVERY_LAUNCH_INTERVAL**
Parameter type	List of string separated by commas
Description	List of Cron intervals to create recovery requests. Format: (<cron_interval>;<cron_interval>) List of intervals in CRON format for launching the creation (except for copy requests) and treatment process, each interval being separated by the ";" character. The paragraph "Managing CRON Format" gives more details about this format. **Note** The first value of the interval corresponds to the seconds and must be set to `*`. If anything else than `*` (for example `0`) is specified for this first value, an issue can occur, as a check is made that the current time matches the CRON expression specified. In the example, if `0` is specified, it would only match when the number of seconds of the current time is equal to 0.
Shipped value	`* * 00-07,20-23 ? * MON-FRI;* * * ? * SAT,SUN`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching

	`reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MAX_RECOVERY_REQUESTS_PER_DAY**
Parameter type	Positive integer
Description	This parameter sets the maximum number of automatic requests of this type that can be created by the Engine per day. The counter is reset to 0 when the Engine starts.
Shipped value	`300`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

6.6.14. Copy

Parameters related to copy

Section of parameters taken into account only if: `COPY_REQUESTS_PROCESSING` = `true`

Parameter name in `ENGINE.properties`	**COPY_CHECK_INTEGRITY**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Check the fixity? Defines whether the integrity of the records should be checked during this operation.
Shipped value	`true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**COPY_LAUNCH_INTERVAL**
Parameter type	List of string separated by commas
Description	List of Cron intervals used to schedule copy request loading. Format: (<cron_interval>;<cron_interval>)

	List of intervals in CRON format for launching the creation (except for copy requests) and treatment process, each interval being separated by the ";" character. The paragraph "Managing CRON Format" gives more details about this format. **Note** The first value of the interval corresponds to the seconds and must be set to `*`. If anything else than `*` (for example `0`) is specified for this first value, an issue can occur, as a check is made that the current time matches the CRON expression specified. In the example, if `0` is specified, it would only match when the number of seconds of the current time is equal to 0.
Shipped value	`* * 00-07,20-23 ? * MON-FRI;* * * ? * SAT,SUN`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

6.6.15. Synchronisation

Parameters related to synchronisation

Section of parameters taken into account only if: `SYNC_ENGINE` = `true`

Parameter name in `ENGINE.properties`	**SYNC_LAUNCH_INTERVAL**
Parameter type	List of string separated by commas
Description	List of Cron intervals to create synchronization requests. Format: (<cron_interval>;<cron_interval>) List of intervals in CRON format for launching the creation (except for copy requests) and treatment process, each interval being separated by the ";" character. The paragraph "Managing CRON Format" gives more details about this format. **Note** The first value of the interval corresponds to the seconds and must be set to `*`. If anything else than `*` (for example `0`) is specified for this first value, an issue can occur, as a check is made that the current time matches the CRON expression specified. In the example, if `0` is specified, it would only match when the number of seconds of the current time is equal to 0.

Shipped value	`* * 00-07,20-23 ? * MON-FRI;* * * ? * SAT,SUN`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MAX_SYNC_REQUESTS_PER_DAY**
Parameter type	Positive integer
Description	This parameter sets the maximum number of automatic requests of this type that can be created by the Engine per day. The counter is reset to 0 when the Engine starts.
Shipped value	`300`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEC_SYNC_TIMEOUT**
Parameter type	Number of seconds
Description	Timeout for synchronization Maximum time allowed for a synchronization request to complete. If a synchronization request does not change its status within this time frame, it is considered to be in a hang state.
Shipped value	`600`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.16. Media Migration

Parameters related to media migration

Section of parameters taken into account only if: `MEDIA_MIGRATION_ENGINE` = `true`

Parameter name in `ENGINE.properties`	**MEDIA_MIGRATION_LAUNCH_INTERVAL**

Parameter type	List of string separated by commas
Description	List of Cron intervals to create postponed media migration requests. Format: (<cron_interval>;<cron_interval>) List of intervals in CRON format for launching the creation (except for copy requests) and treatment process, each interval being separated by the ";" character. The paragraph "Managing CRON Format" gives more details about this format. **Note** The first value of the interval corresponds to the seconds and must be set to `*`. If anything else than `*` (for example `0`) is specified for this first value, an issue can occur, as a check is made that the current time matches the CRON expression specified. In the example, if `0` is specified, it would only match when the number of seconds of the current time is equal to 0.
Shipped value	`* * 00-07,20-23 ? * MON-FRI;* * * ? * SAT,SUN`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MAX_MEDIA_MIGRATION_REQUESTS_PER_DAY**
Parameter type	Positive integer
Description	This parameter sets the maximum number of automatic requests of this type that can be created by the Engine per day. The counter is reset to 0 when the Engine starts.
Shipped value	`300`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.17. Recycling

Parameters related to recycling

Section of parameters taken into account only if: `RECYCLING_ENGINE` = `true`

Parameter name in `ENGINE.properties`	**RECYCLING_LAUNCH_INTERVAL**

Parameter type	List of string separated by commas
Description	List of Cron intervals to create recycling tape requests. Format: (<cron_interval>;<cron_interval>) List of intervals in CRON format for launching the creation (except for copy requests) and treatment process, each interval being separated by the ";" character. The paragraph "Managing CRON Format" gives more details about this format. **Note** The first value of the interval corresponds to the seconds and must be set to `*`. If anything else than `*` (for example `0`) is specified for this first value, an issue can occur, as a check is made that the current time matches the CRON expression specified. In the example, if `0` is specified, it would only match when the number of seconds of the current time is equal to 0.
Shipped value	`* * 00-07,20-23 ? * MON-FRI;* * * ? * SAT,SUN`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MAX_RECYCLING_REQUESTS_PER_DAY**
Parameter type	Positive integer
Description	This parameter sets the maximum number of automatic requests of this type that can be created by the Engine per day. The counter is reset to 0 when the Engine starts.
Shipped value	`300`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.18. Disposal

Parameters related to disposal

Section of parameters taken into account only if: `DISPOSAL_ENGINE` = `true`

Parameter name in `ENGINE.properties`	**DISPOSAL_LAUNCH_INTERVAL**

Parameter type	List of string separated by commas
Description	List of Cron intervals to create disposal requests. Format: (<cron_interval>;<cron_interval>) List of intervals in CRON format for launching the creation (except for copy requests) and treatment process, each interval being separated by the ";" character. The paragraph "Managing CRON Format" gives more details about this format. **Note** The first value of the interval corresponds to the seconds and must be set to `*`. If anything else than `*` (for example `0`) is specified for this first value, an issue can occur, as a check is made that the current time matches the CRON expression specified. In the example, if `0` is specified, it would only match when the number of seconds of the current time is equal to 0.
Shipped value	`* * 00-07,20-23 ? * MON-FRI;* * * ? * SAT,SUN`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MAX_DISPOSAL_REQUESTS_PER_DAY**
Parameter type	Positive integer
Description	This parameter sets the maximum number of automatic requests of this type that can be created by the Engine per day. The counter is reset to 0 when the Engine starts.
Shipped value	`300`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.19. Retention dates computation

Parameters related to retention dates computation

Parameter name in `ENGINE.properties`	**RETENTION_DATE_MAX_PROCESSED_LOTS**
Parameter type	Positive integer

Description	Maximum number of Lots processed by retention dates computation process For disposal by retention schedule requests, a daily engine process recalculates end of life dates when mass modifications take place (retention schedule modification, class modification, etc.). This parameter sets the number of lots processed every day.
Shipped value	`300000`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**RETENTION_DATE_SCHEDULED_HOUR**
Parameter type	Positive integer
Description	Hour in day to schedule the retention dates computation process When mass modifications occur on a retention schedule (e.g. change of retention start or end trigger, duration update, class linkage update), the end-of-life dates of all linked Lots are not recalculated immediately. Instead, a daily engine process performs a batch recomputation of retention end dates. This parameter defines the hour of day (0–23) at which this recomputation is triggered. Disposal requests can only be created after this recomputation has updated the Lots' end-of-life dates in the database. The recomputation process also runs at Engine startup. **Note** Direct modifications performed on an individual Lot trigger an immediate recalculation and are not affected by this scheduling parameter.
Shipped value	`5`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.20. Check Zones Process Parameters

Parameters related to check zones process

Section of parameters taken into account only if: `CHECKZONES_ENGINE = true`

Parameter name in `ENGINE.properties`	**CHECKZONE_TIME_START**

Parameter type	Positive integer
Description	Start of the check zones process. Format: HHMM (HH: hours in 24h format, MM: minutes). It should be different from the MIGRATION_TIME_START parameter to avoid potential conflicts.
Shipped value	`2030`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**CHECKZONE_MAX_LOTS**
Parameter type	Positive integer
Description	Maximum number of lots to check for a zone
Shipped value	`10`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**CHECKZONE_CHECK_INTEGRITY**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Check fixity of the lot after it has been retrieved? Defines whether the integrity of the records should be checked during this operation.
Shipped value	`true`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEC_CHECK_TIMEOUT**
Parameter type	Number of seconds
Description	Timeout for check Maximum time allowed for a check request to complete.

	If a check request does not change its status within this time frame, it is considered to be in a hang state.
Shipped value	`600`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

6.6.20.1. Check Zone Mail parameters

Parameters related to check zone mail

Parameter name in `ENGINE.properties`	**CHECKZONE_SEND_MAIL**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Send the zone check report by mail Specifies whether you want to send a status summary mail of the zones after checking the zones.
Shipped value	`false`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**CHECKZONE_SEND_MAIL_TO**
Parameter type	Characters string
Description	Recipient of the report sent by mail
Shipped value	**No shipped value.**
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**CHECKZONE_SEND_MAIL_LEVEL**
Parameter type	State variable. Possible values: `OK`, `FAILED_TO_CHECK`, `ERROR`

Description	Level from which a mail will be sent Specifies from which zone status level an alert mail should be sent. If the `OK` value is set, a mail will be systematically sent on zone check, If the `FAILED_TO_CHECK` value is set, a mail will be sent where the zone status is at least `FAILED_TO_CHECK` or `ERROR`, If the `ERROR` value is set, a mail will be sent where the zone status is `ERROR`.
Shipped value	`ERROR`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

6.6.21. Arcsys Transfer Servers

Parameters related to Arcsys Transfer Servers

Parameter name in `ENGINE.properties`	**TRANSFER_SERVERS**
Parameter type	List of string separated by commas
Description	The list of transfer server codes. This is the list of the transfer server codes (as set in the `transferServer.conf` property file) separated by "," with which this engine is authorized to communicate. This enables a failover mechanism. List of the transfer server codes with which the engine can communicate for all requests (archiving, retrieval, migration, etc.). If the engine is specialized for an operation, the associated transfer servers can also be specialized for these operations. If a code is not specified, in this case, all transfer servers of the site will be taken into consideration.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**DISTANT_TRANSFER_SERVERS**
Parameter type	List of string separated by commas
Description	The list of distant transfer server codes. This is the list of the transfer server codes (as set in the `transferServer.conf` property file) separated by "," with which this engine is authorized to communicate. This enables a failover mechanism.

	This parameter is used optionally to filter the remote transfer servers targeted (for example, when archiving with a remote copy, to choose which remote transfer servers will perform the archiving). It contains a list of remote transfer server codes. By default, any transfer server can be used.
Shipped value	**No shipped value.**
Option(s)	`ArcHP`: You must have this option to use this parameter.
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**HEALTH_CHECK_TRANSFER_SERVERS_PROC_LOOPSLEEPTIME**
Parameter type	Number of milliseconds
Description	Interval between the checks of the health of the transfer servers Control interval for checking the presence of transfer servers, for failover or load balancing management.
Shipped value	`30000`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**TRANSFER_SERVER_STRATEGY**
Parameter type	State variable. Possible values: `RANDOM`, `FIRST_AVAILABLE`
Description	The strategy for communicating with transfer servers. If `RANDOM`, a random shuffle is performed on the list before each call. If `FIRST_AVAILABLE`, the first available transfer server of the list is systematically called. `FIRST_AVAILABLE` means that the request is always sent to the first transfer server, then to the second if the request fails, and so on. `RANDOM` operates in the same way, but on each request, a transfer server is chosen randomly from the list of transfer servers available.
Shipped value	`RANDOM`
Option(s)	`ArcHP`: If the ArcHP option is enabled, you should set this parameter to RANDOM.
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**TRANSFER_SERVER_SECURITY_PROTOCOL**

Parameter type	State variable. Possible values: `NONE`, `INTERNAL`, `SSL`
Description	Security protocol for communicating with transfer servers Indicates if communication to and from the transfer server is encrypted or not, and the type of encryption (symmetrical for `INTERNAL`: Blowfish algorithm is used; or asymmetrical for `SSL`). Disabling encryption increases performance but may be forbidden depending on the required security level.
Shipped value	`NONE`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.21.1. Transfer server SSL

Parameters related to SSL protocol

Section of parameters taken into account only if: `TRANSFER_SERVER_SECURITY_PROTOCOL` = `SSL`

Parameter name in `ENGINE.properties`	**TRANSFER_SERVER_KEYSTORE_PATH**
Parameter type	File path
Description	Location of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**TRANSFER_SERVER_KEYSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Password of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**TRANSFER_SERVER_KEYSTORE_TYPE**
Parameter type	State variable. Possible values: `JKS`, `PKCS12`
Description	Type of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	`JKS`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**TRANSFER_SERVER_KEYSTORE_KEY_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Password of the key in the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**TRANSFER_SERVER_KEYSTORE_ALIAS**
Parameter type	Characters string
Description	Alias of the key in the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**TRANSFER_SERVER_TRUSTSTORE_PATH**
Parameter type	File path
Description	Location of the truststore used for SSL/TLS exchanges.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.

Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**TRANSFER_SERVER_TRUSTSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Password for the truststore used for SSL/TLS exchanges.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**TRANSFER_SERVER_TRUSTSTORE_TYPE**
Parameter type	State variable. Possible values: `JKS`, `PKCS12`
Description	Type of the truststore used for SSL/TLS exchanges.
Shipped value	`JKS`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.22. Authentication and authorization (LDAP)

Parameters related to authentication and authorization (LDAP)

Parameter name in `ENGINE.properties`	**USERNAME**
Parameter type	Characters string
Description	Identification username LDAP user used to launch the engine. This user must belong to a group with a valid `arcsysRight` attribute but does not require any specific rights. This parameter is not required in SSO mode.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**USERPASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Identification user password LDAP user password used to start the engine. This parameter is not required in SSO mode.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_PROPERTIES_FILE**
Parameter type	File path
Description	Optional external file where the LDAP_** properties will be read. If this parameter is not empty, the LDAP_** properties will be read in this file. Note: Then, the TRUSTSTORE/KEYSTORE_FILE/PASSWORD will also be read in this file. If these properties are also present in the main configuration file, there will not be overridden.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_TYPE**
Parameter type	State variable. Possible values: `OPEN_DJ`, `OPEN_DS`, `OPEN_LDAP`, `ACTIVE_DIRECTORY`, `TIVOLI_DIRECTORY_SERVER`, `SUN_ONE_DIRECTORY_SERVER`
Description	The LDAP server type.
Shipped value	`OPEN_LDAP`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_SERVERS**

Parameter type	List of string separated by commas
Description	List of LDAP servers
Shipped value	`SERVER1`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SERVER1.HOST**
Parameter type	Characters string
Description	Host of the LDAP server
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SERVER1.PORT**
Parameter type	Positive integer
Description	Port of the LDAP server
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_SECURITY_PROTOCOL**
Parameter type	State variable. Possible values: `NONE`, `SSL`, `STARTTLS`
Description	Security protocol to use (NONE, SSL, STARTTLS)
Shipped value	`NONE`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_ARCSYS_ATTRIBUT_RIGHT_NAME**

Parameter type	Characters string
Description	The Arcsys right attribute name in the LDAP
Shipped value	`arcsysRight`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_GLOBAL_ATTRIBUT_MAIL_NAME**
Parameter type	Characters string
Description	The mail attribute name in the LDAP
Shipped value	`mail`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_MODE**
Parameter type	State variable. Possible values: `STANDARD`, `GENERIC`, `FIRST_GROUP`
Description	LDAP mode (STANDARD, GENERIC, FIRST_GROUP)
Shipped value	`GENERIC`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.22.1. LDAP STANDARD mode parameters

These parameters are to be defined only when STANDARD mode is selected

Section of parameters taken into account only if: `LDAP_MODE` = `STANDARD`

Parameter name in `ENGINE.properties`	**LDAP_INITIAL_BIND_LOGIN**
Parameter type	Characters string
Description	LDAP initial bind login
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.

Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_INITIAL_BIND_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	LDAP initial bind password
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_INITIAL_BIND_PATH**
Parameter type	Characters string
Description	LDAP initial bind path
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_ADMIN_LOGIN**
Parameter type	Characters string
Description	LDAP administrator login
Shipped value	`anonymous`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_ADMIN_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).

Description	LDAP administrator password
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_ADMIN_PATH**
Parameter type	Characters string
Description	LDAP administrator path
Shipped value	`dc=infotel,dc=com`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_USER_PATH**
Parameter type	Characters string
Description	LDAP user path (all the Arcsys users must be at this LDAP path)
Shipped value	`ou=users,dc=infotel,dc=com`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_ARCSYS_GROUP_PATH**
Parameter type	Characters string
Description	LDAP group path (all the Arcsys groups must be at this LDAP path)
Shipped value	`ou=arcsysgroup,ou=groups,dc=infotel,dc=com`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_USER_TYPE_LIST**
Parameter type	List of string separated by commas
Description	User identifier attribute name in the LDAP (it must not be changed and be unique for each user because it is saved in the database - it is possible to

	use a list but it is not recommended, in this case, the parameters are tried in the order they are written)
Shipped value	`cn,uid`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_GROUP_TYPE**
Parameter type	Characters string
Description	Group identifier attribute name in the LDAP (it must not be changed and be unique for each group because it is saved in the database)
Shipped value	`cn`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_DISPLAY_NAME_ATTRIBUTE_NAME**
Parameter type	Characters string
Description	Attribute name which contains the value to display in the LDAP
Shipped value	`sn`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_REFERRAL_FOLLOW**
Parameter type	State variable. Possible values: `follow`, `ignore`
Description	Enables to "follow" or "ignore" referrals
Shipped value	`follow`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.22.2. LDAP GENERIC mode parameters

These parameters are to be defined only when GENERIC mode is selected

Section of parameters taken into account only if: `LDAP_MODE` = `GENERIC`

Parameter name in `ENGINE.properties`	**LDAP_AUTHENT_DN**
Parameter type	Characters string
Description	The LDAP user DN for authentication (initial bind to create the connection). It is the DN/login of a user with sufficient rights to consult the entire LDAP and, in the case of the Web Agent, to change a password (for example, for Active Directory, this user must be part of the Administrators group). If anonymous is chosen, an anonymous connection is used (under the premise that this is authorized by the specified directory). Such a connection cannot be used to write in the LDAP; therefore an anonymous login cannot be used if you want to enable the password change option
Shipped value	`anonymous`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_AUTHENT_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	The LDAP user password for authentication (initial bind to create the connection).
Shipped value	**No shipped value.**
Parameter taken into account only if:	`LDAP_AUTHENT_DN` is different from `anonymous`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_USER_LOGIN_ATTRIBUTE_NAME**
Parameter type	Characters string
Description	The attribute name of an LDAP user entry containing the user login. This attribute must not be changed and be unique for each user because it is saved in the database.
Shipped value	`cn`
Parameter change assessment	After module reboot.

Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_GROUP_ID_ATTRIBUTE_NAME**
Parameter type	Characters string
Description	The attribute name of an LDAP group entry containing the group name. This attribute must not be changed and be unique for each user because it is saved in the database.
Shipped value	`cn`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_USER_DISPLAY_NAME_ATTRIBUTE_NAME**
Parameter type	Characters string
Description	The attribute name of an LDAP user entry containing the name to display for a user.
Shipped value	`sn`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_GROUP_DISPLAY_NAME_ATTRIBUTE_NAME**
Parameter type	Characters string
Description	The attribute name of an LDAP group entry containing the name to display for a group.
Shipped value	`sn`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_GROUP_MEMBERSHIP_ATTRIBUTE_NAME**
Parameter type	Characters string
Description	Group membership attribute name in the LDAP ("member" or "uniqueMember")

Shipped value	`member`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

LDAP user search

The users are searched in the LDAP from the search base DN according to the search scope and the search filter.

Parameter name in `ENGINE.properties`	**LDAP_USER_SEARCH_BASE_DN**
Parameter type	Characters string
Description	The base DN of the LDAP server for user searches.
Shipped value	`ou=users,dc=infotel,dc=com`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_USER_SEARCH_SCOPE**
Parameter type	State variable. Possible values: `BASE`, `ONE`, `SUB`, `SUBORDINATE_SUBTREE`
Description	The scope of the user search in the LDAP server. * BASE: only the entry defined by the base DN is considered for the search * ONE: only the entries immediately below the entry defined by the base DN are considered for the search, the entry defined by the base DN is excluded * SUB: all the entries below the entry defined by the base DN and the entry defined by the base DN are considered for the search * SUBORDINATE_SUBTREE: all the entries immediately below the base DN are considered for the search, the entry defined by the base DN is excluded
Shipped value	`SUB`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_USER_SEARCH_FILTER**
Parameter type	Characters string

Description	The user search filter in the LDAP server.
Shipped value	`(objectClass=inetOrgPerson)`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

LDAP group search

The groups are searched in the LDAP from the search base DN according to the search scope and the search filter.

Parameter name in `ENGINE.properties`	**LDAP_GROUP_SEARCH_BASE_DN**
Parameter type	Characters string
Description	The base DN of the LDAP server for group searches.
Shipped value	`ou=arcsysgroup,ou=groups,dc=infotel,dc=com`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_GROUP_SEARCH_SCOPE**
Parameter type	State variable. Possible values: `BASE`, `ONE`, `SUB`, `SUBORDINATE_SUBTREE`
Description	The scope of the group search in the LDAP server. * BASE: only the entry defined by the base DN is considered for the search * ONE: only the entries immediately below the entry defined by the base DN are considered for the search, the entry defined by the base DN is excluded * SUB: all the entries below the entry defined by the base DN and the entry defined by the base DN are considered for the search * SUBORDINATE_SUBTREE: all the entries immediately below the base DN are considered for the search, the entry defined by the base DN is excluded
Shipped value	`SUB`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_GROUP_SEARCH_FILTER**

Parameter type	Characters string
Description	The search filter for groups in the LDAP server.
Shipped value	`(objectClass=groupOfNames)`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

LDAP - Other parameters

Other parameters for the LDAP

Parameter name in `ENGINE.properties`	**LDAP_FOLLOW_REFERRALS**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Flag to indicate if the referral must be followed (true) or not (false)
Shipped value	`false`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_GROUP_SEARCH_SCOPE**
Parameter type	State variable. Possible values: `BASE`, `ONE`, `SUB`, `SUBORDINATE_SUBTREE`
Description	The scope of the group search in the LDAP server. * BASE: only the entry defined by the base DN is considered for the search * ONE: only the entries immediately below the entry defined by the base DN are considered for the search, the entry defined by the base DN is excluded * SUB: all the entries below the entry defined by the base DN and the entry defined by the base DN are considered for the search * SUBORDINATE_SUBTREE: all the entries immediately below the base DN are considered for the search, the entry defined by the base DN is excluded
Shipped value	`SUB`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**LDAP_ALIAS_DEREFERENCING_POLICY**

Parameter type	State variable. Possible values: `ALWAYS`, `FINDING`, `SEARCHING`, `NEVER`
Description	Alias dereferencing policy (ALWAYS, FINDING, SEARCHING, NEVER)
Shipped value	`NEVER`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.23. Common Truststore / Keystore

Common Truststore / Keystore parameters

Parameter name in `ENGINE.properties`	**TRUSTSTORE_FILE**
Parameter type	File path
Description	Location of the truststore file used for SSL/TLS exchanges. This file is used for securing both LDAP and RMI. You can inform it in the `etc` directory of the component, for example. Absolute file path. Use Unix like separators ('`/`'), even on Windows.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**TRUSTSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Encrypted password of the truststore file (passphrase)
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**KEYSTORE_FILE**
Parameter type	File path

Description	Location of the keystore file. This file is used for securing both LDAP and RMI. You can, for example, set it in the component's directory, `etc`.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**KEYSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Encrypted password of the keystore file (passphrase).
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.24. Language

The two parameters of this sections are required to load the localized files.

Localized files:

`ERROR_message_xx_XX.properties`

`TEMPLATE_MAIL_Alert_End_Request_xx_XX.properties`

`TEMPLATE_MAIL_Alert_Lot_xx_XX.properties`

`TEMPLATE_MAIL_CheckZones_xx_XX.properties`

Where xx is replaced by the language code, and XX is replaced by the country code.

Parameter name in `ENGINE.properties`	**LANGUAGE_CODE**
Parameter type	State variable. Possible values: `fr`, `en`
Description	The language code
Shipped value	`fr`
Parameter change assessment	After module reboot.

Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**COUNTRY_CODE**
Parameter type	State variable. Possible values: `FR`, `GB`
Description	The country code
Shipped value	`FR`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.25. Event management

Event management parameters

Parameter name in `ENGINE.properties`	**EVENT_BASE_ALERT_LOOPSLEEP_DAY**
Parameter type	Positive integer
Description	Interval in days between two processing loops of events for zone change and end of retention.
Shipped value	**No shipped value.**
Default value	`1`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	0

Parameter name in `ENGINE.properties`	**EVENT_ALERT_TIME_START_CHG_ZONE**
Parameter type	Characters string
Description	Event processing time for zone change and end of retention. Attended format: hhmm Example(s): `0601` for 6 hours and 1 minute
Shipped value	**No shipped value.**
Default value	`0600`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Minimum length	4
Maximum length	4

Parameter name in `ENGINE.properties`	**EVENT_ALERT_ADMIN_MAIL_LIST**
Parameter type	List of string separated by commas
Description	List of the administrators mails. The alerts will be systematically sent to the recipients provided in this parameter in the following cases: 1. When a user should receive an alert and cannot receive it because the mail is not associated in the LDAP directory (attribute 'mail'); 2. When the EVENT_ALERT_TO_ADMIN parameter contains the value 'true' and it is a migration or an end of retention alert.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**EVENT_ALERT_TO_ADMIN**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Systematically sends alerts for the end of retention and migration to any of the Arcsys alert administrators.
Shipped value	**No shipped value.**
Default value	`false`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.26. Mail management

Used by the Engine to send emails: either alert emails (end of archiving/retrieval request, zone changes, zone checks) or emails from the notification system. MAIL_DIRECTORY is not used for the notification system.

Parameter name in `ENGINE.properties`	**MAIL_SEND_PROC_LOOPSLEEP_TIME**

Parameter type	Number of seconds
Description	Time for the mail sending loop.
Shipped value	**No shipped value.**
Default value	`600`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	0

Parameter name in `ENGINE.properties`	**MAIL_SENDER**
Parameter type	Characters string
Description	The mails recipient
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MAIL_SMTP_SERVER**
Parameter type	Characters string
Description	Name of the SMTP server for sending the mails.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MAIL_SMTP_SERVER_PORT**
Parameter type	Positive integer
Description	Port of the SMTP Server.
Shipped value	**No shipped value.**
Default value	`25`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MAIL_SSL_PROTOCOLS**
Parameter type	List of string separated by commas
Description	Specifies the SSL protocols that will be enabled for SSL connections for mail sending. The property value is a comma separated list of protocols. Example: TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MAIL_DIRECTORY**
Parameter type	File path
Description	The directory in which mails are temporarily saved. This is the directory where the files encoded in MIME format of the alert mails generated will be stored before sending. These files are deleted when the mails are sent correctly. The specified directory must exist.
Shipped value	**No shipped value.**
Default value	`mail`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MAIL_USING_PASSWORD**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether mail sending requires a login/password.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.26.1. Login and password

Section of parameters taken into account only if: `MAIL_USING_PASSWORD` = `true`

Parameter name in `ENGINE.properties`	**MAIL_LOGIN**

Parameter type	Characters string
Description	User name where sending a mail that requires a login/password.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**MAIL_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Encrypted password of the user where mail sending requires a login/password.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.27. Check configuration file rights

Rights of the configuration file

Parameter name in `ENGINE.properties`	**CURRENT_FILE_RIGHT_MUST_BE**
Parameter type	Positive integer
Description	If you want the module to check that this properties file has the specified Unix rights, set an octal value here.If you set it to 0, no check will occur (default). We recommend you to set it to 600 (read only for user).
Shipped value	`0`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Maximum length	3

6.6.28. Rename mask

Rename mask parameters

Parameter name in `ENGINE.properties`	**DEFAULT_AFP_RENAME_MASK**
Parameter type	Characters string
Description	Naming mask of retrieved files used in partial retrieval without fusion of the results files. The mask is defined by a character string where the following substrings will be automatically replaced: `$METADATA{<Keyword code>}` for a metadata value; `$DATE` for the date YYYYMMdd (e.g. 20240426); `$TIMESTAMP` for a timestamp YYYYMMddhhmmSS (e.g. 20240426112313); `$REQIDENT` for the request identifier; `$LOTIDENT` for the lot identifier; `$OBJIDENT` for the object identifier; Example(s): `$REQIDENT_$METADATA{NUMERO_CLIENT}_$DATE_$METADATA{TYPE_DOCUMENT}.afp` **Note** It is important to choose a naming mask that ensures the uniqueness of the results files. You need to get a file per retrieved object, so you need to use this metadata, which ensures their uniqueness, or use the `$OBJIDENT` variable, otherwise the AFP files will overwrite each other and can become corrupt.
Shipped value	`result$REQIDENT_$OBJIDENT.afp`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.29. Database

Database parameters

Parameter name in `ENGINE.properties`	**RETRY_MAX_CHECK_DB_CONNECTION**
Parameter type	Positive integer
Description	Maximum number of database connections trials When the database is detected offline by the Control process of the Engine (which checks each `CTRLPROC_LOOPSLEEPTIME` the state of the

	database), the Engine is stopped after `RETRY_MAX_CHECK_DB_CONNECTION` consecutive offline detections and remains stopped even if the database is online again. If the database is identified as online before reaching `RETRY_MAX_CHECK_DB_CONNECTION` offline detections, the attempt counter is reset to zero.
Shipped value	`5`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	0

Parameter name in `ENGINE.properties`	**ALERT_PROPERTIES_FILE**
Parameter type	File path
Description	The SNMP configuration file contains the configuration of the alerts. See 'notify.conf File' part in the Arcsys administration manual
Shipped value	`${TARGET_FOLDER}/ArcsysEngine/etc/notify.conf`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.30. Parameters linked with REST callbacks

The parameters about the request destination URL and the way to transmit Arcsys requests information

Parameter name in `ENGINE.properties`	**SEND_REST_REQUESTS**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether the engine should call a REST method in end of request.
Shipped value	`false`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

6.6.30.1. REST callback when request end

Section of parameters taken into account only if: `SEND_REST_REQUESTS` = `true`

Parameter name in `ENGINE.properties`	**SEND_REST_REQUESTS_TYPE**
Parameter type	Characters string
Description	List without separator of the request types to be considered in the call to the REST method. Possible values: ARMCSPDYXLE. A (archiving), R (retrieval), U (archive restitution), M (migration), C (check), S (delete), P(recovery), D (copy), Y (synchronization), X (postponed), L (recycling), E (media expiry). The list gives the set of desired types concatenated
Shipped value	`AR`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEND_REST_REQUESTS_URL**
Parameter type	Characters string
Description	URL to send the REST requests. The URL can include the following variables, which are automatically populated during the call: - `%REQID%` (request identifier), - `%REQSTATUS%` (request status: 1 for ENDED, 6 for ERROR, 4 for ABORTED), - `%REQTYPE%` (request type : 'A' for archiving, 'R' for retrieval, 'U' for archive restitution, 'M' for migration, 'P' for recovery, 'C' for check, 'S' for deletion, 'D' for duplicate, 'Y' for synchronization, 'X' for postpone, 'E' for media expiration, 'L' for recycle), - `%APPLICATION_AGENT_HOST%` (application agent host involved), - `%APPLICATION_AGENT_CODE%` (code of the application agent involved), - `%LOT_ID%` (identifier of the lot involved), - `%LOT_CODE%` (lot code), - `%LOT_CREATOR%` (lot creator), - `%BASE_ID%` (repository identifier), - `%BASE_CODE%` (repository code) - `%COLLECTION_ID%` (collection identifier), - `%COLLECTION_CODE%` (collection code)

	Note When the REST callback message cannot be sent, the error message systematically includes the string `ENGINE_REST_CALLBACK_FAILURE_LOG`, along with the request ID and the request URL in the following format: `"The error [ENGINE_REST_CALLBACK_FAILURE_LOG] occurred on request REQID[...] and request URL [...]"`. Clients can use this specific string to handle such errors programmatically.
Shipped value	`http://localhost:8080/servlet/api/cb/%REQID%?status=%REQSTATUS%&apphost=%APPLICATION_AGENT_HOST%&appcode=%APPLICATION_AGENT_CODE%&lotId=%LOT_ID%`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

REST request keystore

Used for SSL/TLS exchanges.

Parameter name in `ENGINE.properties`	**SEND_REST_REQUESTS_KEYSTORE_TYPE**
Parameter type	Characters string
Description	Type of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	`jks`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEND_REST_REQUESTS_KEYSTORE_FILE**
Parameter type	File path
Description	Location of the keystore containing the certificate used for SSL/TLS exchanges. Absolute file path. Use Unix like separators ('/'), even on Windows.
Shipped value	**No shipped value.**

Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEND_REST_REQUESTS_KEYSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Password of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	**No shipped value.**
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

REST request truststore

Used for SSL/TLS exchanges.

Parameter name in `ENGINE.properties`	**SEND_REST_REQUESTS_TRUSTSTORE_TYPE**
Parameter type	Characters string
Description	Type of the truststore used for SSL/TLS exchanges.
Shipped value	`jks`
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEND_REST_REQUESTS_TRUSTSTORE_FILE**
Parameter type	File path
Description	Location of the truststore used for SSL/TLS exchanges. Absolute file path. Use Unix like separators ('/'), even on Windows.

Shipped value	**No shipped value.**
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEND_REST_REQUESTS_TRUSTSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Password for the truststore used for SSL/TLS exchanges.
Shipped value	**No shipped value.**
Parameter change assessment	Hot-reload property. Hot-reload properties can be edited without module reboot (if `ENABLE_HOT_RELOAD_ENGINE_PARAMETERS=true`) or when launching `reload_engine` script.(See the section reload_engine script.sh (or .bat) Script).
Required	False, this parameter is optional.

6.6.31. Parameters loading

Parameters loading mechanism

Parameter name in `ENGINE.properties`	**ENABLE_HOT_RELOAD_ENGINE_PARAMETERS**
Parameter type	Boolean. Possible values: `true`, `false`
Description	Specifies whether changes to settings in hot mode (without engine startup) is authorized or not. If it is authorized, only the settings that are explicitly specified as modifiable in hot reload mode will be taken into account without restart. Not all parameters are enabled for a hot reload.
Shipped value	`false`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.32. Cleaning of the incoherent entities in database

Cleaning of the incoherent entities in database parameters

Parameter name in `ENGINE.properties`	**CLEAN_ENTITIES_SCHEDULED_HOUR**
Parameter type	Positive integer
Description	Time (hour) at which the task of cleaning inconsistent entities (synchronous retrieval requests, mover medias, lot pools) in the database is triggered each day. This task is also executed when the engine starts.
Shipped value	`4`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	0
Maximum value	23

Parameter name in `ENGINE.properties`	**CLEAN_ONLINE_RESTORE_REQUESTS**
Parameter type	Boolean. Possible values: `true`, `false`
Description	This parameter is used to specify if this engine must clean in the Arcsys Database the synchronous retrieval requests dated over than `SEC_ONLINE_RESTORE_REQUEST_TIMEOUT` seconds. The cleaning process is launched on engine startup, and then every day at the hour specified in `CLEAN_ENTITIES_SCHEDULED_HOUR`. **Important** **Even if ArcHP Option is enabled, it is forbidden to have several engines with this parameter set to true.**
Shipped value	`true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**SEC_ONLINE_RESTORE_REQUEST_TIMEOUT**
Parameter type	Number of seconds
Description	The time period at the end of which a synchronous retrieval request is timed out when it is not finished.
Shipped value	`86400`
Parameter taken into account only if:	`CLEAN_ONLINE_RESTORE_REQUESTS` = `true`
Parameter change assessment	After module reboot.

Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**CLEAN_MOVER_MEDIAS**
Parameter type	Boolean. Possible values: `true`, `false`
Description	This parameter is used to specify if this engine must clean the inconsistent mover medias in the Arcsys Database. The cleaning process is launched on engine startup and then every day at the hour specified in CLEAN_ENTITIES_SCHEDULED_HOUR.
Shipped value	`true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `ENGINE.properties`	**CLEAN_LOT_POOLS**
Parameter type	Boolean. Possible values: `true`, `false`
Description	This parameter is used to specify if this engine must clean the inconsistent lot pools in the Arcsys Database. The cleaning process is launched on engine startup and then every day at the hour specified in CLEAN_ENTITIES_SCHEDULED_HOUR.
Shipped value	`true`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

6.6.33. Generation of attestations

Generation of attestations parameters

Parameter name in `ENGINE.properties`	**ATTESTATION_DIRECTORY**
Parameter type	Directory path
Description	Path of the directory where the attestations will be generated The path must correspond to an existing directory, with adequate access rights (read, write and execute). Absolute file path. Use Unix like separators ('/'), even on Windows.
Shipped value	`attestations`
Parameter change assessment	After module reboot.

Required	False, this parameter is optional.

6.7. etc/ERROR_message_xx_XX.properties File

6.7.1. Definition

These files contain all the labels of the error messages.

Syntax:

xx: Defines the language. XX: Defines the country.

Arcsys comes with two files by default: `ERROR_message_fr_FR.properties` and `ERROR_message_en_GB.properties`

Caution

It is strongly advised you do not make any changes in these texts, as this may lead to the exact cause for the error being unknown.

6.8. etc/TEMPLATE_MAIL_Alert_End_Request_fr_FR.properties File

6.8.1. Definition

This file is the mail template sent at the end of an archiving or retrieval request.

Syntax:

xx: Defines the language. XX: Defines the country.

Arcsys comes with two files by default: `TEMPLATE_MAIL_Alert_End_Request_fr_FR.properties` and `TEMPLATE_MAIL_Alert_End_Request_en_GB.properties`.

6.8.2. General Parameters For Mail Templates

The parameters are detailed in the section Customizing the Mail Template Option.

6.8.3. Specific Parameters

`ARCHIVE_IDENTIFICATION`: Record identification

`ARCHIVING_DESCRIPTION_TITLE`: Description of the list of archived files

RESTITUTION_DESCRIPTION_TILE: Description of the list of retrieved files

ARCHIVING_OBJECT_DESC_xx: Description of the different configuration of the record objects (name, identifier, checksum function, checksum value, size, etc.).

RESTITUTION_OBJECT_DESC_xx: Description of the different configuration of the retrieved objects (name, identifier, checksum function, checksum value, size, etc.).

6.8.4. Specific Parameters in Mails Sent to the Administrator

MAIL_LINE_HEADER_ADMIN_PROBLEM_xx: Mail header lines for messages sent to administrators in the event of a missing e-mail address for normal recipients.

ADMIN_ALERT_RECIPIENTS_TITLE: Line indicating to the administrator the users to whom the alert was originally destined.

ADMIN_ALERT_NO_MAIL_TITLE: Line indicating to the administrator the users to whom the alert was originally destined who do not have e-mail (and who therefore could not be sent the alert).

6.9. etc/TEMPLATE_MAIL_ AlertLot_xx_XX.properties File

6.9.1. Definition

This file is the mail template sent when changing a lot zone (or when it reaches its disposal).

Syntax:

xx: Defines the language. XX: Defines the country.

Arcsys comes with two files by default: `TEMPLATE_MAIL_Alert_Lot_fr_FR.properties` and `TEMPLATE_MAIL_Alert_Lot_en_GB.properties`

6.9.2. General Parameters For Mail Templates

The parameters are detailed in the section Customizing the Mail Template Option.

6.9.3. Specific Parameters

ARCHIVE_END_LIFE_TITLE/ARCHIVE_MIGRATE_TITLE: Title of the section containing the records that are at disposal / that will be migrated.

ARCHIVE_IDENTIFICATION: Record identification

ARCHIVE_END_LIFE_DATE/ARCHIVE_MIGRATION_DATE: Line indicating the time when the record reached its scheduled end of retention date or was migrated.

ARCHIVE_END_LIFE_ZONE/ARCHIVE_MIGRATION_ZONES: Line indicating the current zone of the record (and the next zone, for migration).

ARCHIVE_DESC_xx: Description of the different configuration of the record (repository code, collection, lot label, lot size).

METADATA_DESC: Description of a metadata item.

ARCHIVE_OBJECTS_TITLE: Title of the section giving the objects in the record.

OBJECT_DESC_xx: Description of the different configuration of the record objects (name, identifier, type, size).

DATE_FORMAT: Format of all dates associated.

6.9.4. Specific Parameters in Mails Sent to the Administrator

MAIL_LINE_HEADER_ADMIN_COPY_xx: Mail header lines for messages sent systematically with copy to the administrators.

MAIL_LINE_HEADER_ADMIN_PROBLEM_xx: Mail header lines for messages sent to administrators in the event of a missing email address for normal recipients.

ADMIN_ALERT_RECIPIENTS_TITLE: Line indicating to the administrator the users to whom the alert was originally destined.

ADMIN_ALERT_NO_MAIL_TITLE: Line indicating to the administrator the users to whom the alert was originally destined who do not have mail (and who therefore could not be sent the alert).

6.10. etc/TEMPLATE_MAIL_ CheckZones_xx_XX.properties File

6.10.1. Definition

This file is the mail template sent for a zone check. Syntax:

xx: Defines the language. XX: Defines the country.

Arcsys comes with two files by default: `TEMPLATE_MAIL_CheckZones_fr_FR.properties` and `TEMPLATE_MAIL_CheckZones_en_GB.properties`

6.10.2. General Parameters For Mail Templates

The parameters are detailed in the section Customizing the Mail Template Option.

6.10.3. Specific Parameters

- *ZONE_CHECKED*: Message displayed when a zone has been checked. You can use the variables %IDENT_ZONE% (zone identification) and %STATUS_ZONE% (zone status) in this parameter;
- *ZONE_NOT_CHECKED*: Message attached when a zone has not been checked (in this case, its last status is given). You can use the variables %IDENT_ZONE % (zone identification), %LAST_STATUS_DATE% (date of the last check) and %LAST_STATUS% (status of the last check) in this parameter.

6.11. etc/THIRD_PARTY_SEARCH_ENGINE.properties File

6.11.1. Definition

This file contains the parameters for accessing the Third Party Search Engine for indexing or search. See its description in page 139, « Configuring ArcRFT Option ».

6.12. etc/arcsysJAAS.conf File

This file contains the parameters of the Java authorization and authentication service. See the description in JAAS configuration file.

6.13. launch_engine.sh (or .bat) Script

This script starts up the Arcsys Engine. The Arcsys Engine is in 'launched' status but does not process the requests. It will process the requests on Arcsys Engine start.

6.14. shutdown_engine.sh (or .bat) Script

This script stops the Arcsys Engine completely. Any requests in progress are stopped in a stable status.

6.15. start_engine.sh (or .bat) Script

This script gives the Arcsys Engine the order to process the requests. Before running this script, the Arcsys Engine must first be in 'launched' status.

6.16. stop_engine.sh (or .bat) Script

This script gives the Arcsys Engine the order to stop processing the requests. Before running this script, the Arcsys Engine must first be in 'launched' status.

6.17. stop_engine_gracefully.sh (or .bat) Script

This script gives the Arcsys Engine the order to stop processing the requests. Those requests in progress are normally completed. Before running this script, the Arcsys Engine must first be in 'launched' status.

6.18. status_engine.sh (or .bat) Script

This script sends a return code matching the current status of the Arcsys Engine (see the same page).

Return code value	Arcsys Engine status	Description
31	STARTING	The Arcsys Engine prepares to process requests (`start_engine` script launched).
32	STOPPING	The Arcsys Engine prepares to stop processing requests (`start_engine` script launched).
33	STARTED_NOT_WORKING	The Arcsys Engine is started but not processing any requests. (`launch_engine` or `stop_engine` script launched).
34	STARTED_AND_WORKING	The Arcsys Engine is started and processing requests.
35	SHUTTING_DOWN	The Arcsys Engine prepares to stop (`shutdown_engine` script launched)
36	SHUTDOWN	The Arcsys Engine is stopped.
39	GET_STATUS_ERROR	Error in retrieving the status.

6.19. encrypt.sh (or .bat) Script

This script allows to encrypt passwords. The password encryption is detailed in the Arcsys Installation Manual.

6.20. generate-key.sh (or .bat) Script

This script allows to generate a symmetric key that can be used to encrypt passwords. The key generation is detailed in the Arcsys Installation Manual.

6.21. reload_engine.sh (or .bat) Script

This script is used to reload Arcsys Engine parameters when *ENABLE_HOT_RELOAD_ENGINE_PARAMETERS* is set to `false`.

Note

Only the settings that are explicitly specified as modifiable in hot reload mode will be taken into account without restart.

For more information about *ENABLE_HOT_RELOAD_ENGINE_PARAMETERS* parameter, see page 284, « ENGINE.properties File »).

6.22. Managing CRON Format

6.22.1. Description

CRON format is the format used by the CRON program that allows Unix system users to automatically run scripts, commands or software at a date and time specified in advance, or according to a cycle defined in advance.

In Arcsys, the CRON format is used to define a range in which automatic requests can be processed.

6.22.2. Format

A CRON expression is made up of 6 or 7 fields separated by spaces. These fields can contain certain authorized values as well as a combination of these values. The fields as well as their authorized values are informed in the table below (taking the simplified example of the table found at http://www.quartz-scheduler.org/documentation/quartz-2.3.0/tutorials/crontrigger.html[1]):

	Mandatory?	Authorized values	Special characters allowed
Seconds	YES	0-59	, - * /
Minutes	YES	0-59	, - * /
Hours	YES	0-23	, - * /
Day of the month	YES	1-31	, - * ? / L W
Month	YES	1-12 or JAN-DEC	, - * /
Day of the week	YES	1-7 or SUN-SAT	, - * ? / L #
Year	NO	empty, 1970-2099	, - * /

[1] http://www.quartz-scheduler.org/documentation/quartz-2.3.0/tutorials/crontrigger.html

6.22.3. Example of CRON range

From 20:00 to 08:00. from Monday to Friday and all day Saturday and Sunday (make sure the last hour is counted -> to stop at 8:00 am., specify 7 in the parameter as this is the seventh full hour): `* * 01-07,20-23 ? * MON-FRI;* * * ? * SAT,SUN`

range of one minute (trigger at 01:30 for 60 seconds): `* 30 01 ? * *`

from 01:30 to 02:30 every day (note that the last minute is counted):

`* 30-59 01 ? * *; * 0-29 02 ? * *`

6.23. formatpolicy.sh (or .bat) Script

6.23.1. Definition

The `formatpolicy.sh` script is used to perform import, export, modification and delete functions on the format policies in Arcsys.

6.23.2. Importing a Format Policy

To import a format policy in Arcsys, run the following command:

```
formatpolicy.sh(.bat) -a import -b <baseId> -f <filePath>
```

where:

- `baseId` represents the identifier of the Arcsys repository in which the format policy will be created.
- `filePath` represents the path to the XML file defining the format policy.

Important

You must import the relevant policy in each repository where you wish to use it.

6.23.3. Exporting a Format Policy

To export an existing format policy, run the following command:

```
formatpolicy.sh(.bat) -a export -d <dirPath> [-i <policyId> | [-b <baseId> -c
 <policyCode>]]
```

where:

- `dirPath` represents the folder in which the format policy will be exported in XML file format.
- The format policy to be exported can be identified via the `policyId` format policy parameter in Arcsys, or via the `baseId` and `policyCode` parameters that represent the identifier of the Arcsys repository in which the format policy is located as well as the code of the latter respectively.

6.23.4. Modifying a Format Policy

To modify an existing format policy, run the following command:

```
formatpolicy.sh(.bat) -a modify -f <filePath> [-i <policyId> | [-b <baseId> -c
 <policyCode>]]
```

where:

- `filePath` represents the XML file defining the new content of the format policy to modify.
- The format policy to be modified can be identified via the `policyId` format policy parameter in Arcsys, or via the `baseId` and `policyCode` parameters that represent the identifier of the Arcsys repository in which the format policy is located as well as the code of the latter respectively.

6.23.5. Deleting a Format Policy

To delete an existing format policy, run the following command:

```
formatpolicy.sh(.bat) -a delete [-i <policyId> | [-b <baseId> -c <policyCode>]]
```

where:

- The format policy to be deleted can be identified via the `policyId` format policy parameter in Arcsys, or via the `baseId` and `policyCode` parameters that represent the identifier of the Arcsys repository in which the format policy is located as well as the code of the latter respectively.

6.24. attestationpolicy.sh (or .bat) Script

6.24.1. Definition

The `attestationpolicy.sh` script is used to perform import, export, modification and delete functions on the attestation policies in Arcsys.

6.24.2. Importing an Attestation Policy

To import an attestation policy in Arcsys, run the following command:

```
attestationpolicy.sh(.bat) -a import -b <repositoryId> -f <filePath>
```

where:

- `repositoryId` represents the identifier of the Arcsys repository in which the attestation policy will be created.
- `filePath` represents the path to the XML file defining the attestation policy.

Important

You must import the relevant policy in each repository where you wish to use it.

6.24.3. Exporting an Attestation Policy

To export an existing attestation policy, run the following command:

```
attestationpolicy.sh(.bat) -a export -d <dirPath> [-i <policyId> | [-b <repositoryId> -c
 <policyCode>]]
```

where:

- `dirPath` represents the folder in which the attestation policy will be exported in XML file format.
- The attestation policy to be exported can be identified via the `policyId` attestation policy parameter in Arcsys, or via the `repositoryId` and `policyCode` parameters that represent the identifier of the Arcsys repository in which the attestation policy is located as well as the code of the latter respectively.

6.24.4. Modifying an Attestation Policy

To modify an existing attestation policy, run the following command:

```
attestationpolicy.sh(.bat) -a modify -f <filePath> [-i <policyId> | [-b <repositoryId> -c
 <policyCode>]]
```

where:

- `filePath` represents the XML file defining the new content of the attestation policy to modify.

- The attestation policy to be modified can be identified via the `policyId` attestation policy parameter in Arcsys, or via the `repositoryId` and `policyCode` parameters that represent the identifier of the Arcsys repository in which the attestation policy is located as well as the code of the latter respectively.

6.24.5. Deleting an Attestation Policy

To delete an existing attestation policy, run the following command:

```
attestationpolicy.sh(.bat) -a delete [-i <policyId> | [-b <repositoryId> -c <policyCode>]]
```

where:

- The attestation policy to be deleted can be identified via the `policyId` attestation policy parameter in Arcsys, or via the `repositoryId` and `policyCode` parameters that represent the identifier of the Arcsys repository in which the attestation policy is located as well as the code of the latter respectively.

6.25. workflowpolicy.sh (or .bat) Script

The `workflowpolicy.sh` script is used to perform import, export, modification and delete functions on the workflow policies in Arcsys.

6.25.1. Importing a Workflow Policy

To import a workflow policy in Arcsys, run the following command:

```
workflowpolicy.sh(.bat) -a import -b <repositoryId> -f <filePath>
```

where:

- `repositoryId` represents the identifier of the Arcsys repository in which the workflow policy will be created.
- `filePath` represents the path to the XML file defining the workflow policy.

Important

You must import the relevant policy in each repository where you wish to use it.

6.25.2. Exporting a Workflow Policy

To export an existing workflow policy, run the following command:

```
workflowpolicy.sh(.bat) -a export -d <dirPath> [-i <policyId> | [-b <repositoryId> -c
 <policyCode>]]
```

where:

- `dirPath` represents the folder in which the workflow policy will be exported in XML file format.
- The workflow policy to be exported can be identified via the `policyId` workflow policy parameter in Arcsys, or via the `repositoryId` and `policyCode` parameters that represent the identifier of the Arcsys repository in which the workflow policy is located as well as the code of the latter respectively.

6.25.3. Modifying a Workflow Policy

To modify an existing workflow policy, run the following command:

```
workflowpolicy.sh(.bat) -a modify -f <filePath> [-i <policyId> | [-b <repositoryId> -c
 <policyCode>]]
```

where:

- `filePath` represents the XML file defining the new content of the workflow policy to modify.
- The workflow policy to be modified can be identified via the `policyId` workflow policy parameter in Arcsys, or via the `repositoryId` and `policyCode` parameters that represent the identifier of the Arcsys repository in which the workflow policy is located as well as the code of the latter respectively.

6.25.4. Deleting a Workflow Policy

To delete an existing workflow policy, run the following command:

```
workflowpolicy.sh(.bat) -a delete [-i <policyId> | [-b <repositoryId> -c <policyCode>]]
```

where:

- The workflow policy to be deleted can be identified via the `policyId` workflow policy parameter in Arcsys, or via the `repositoryId` and `policyCode` parameters that represent the identifier of the Arcsys repository in which the workflow policy is located as well as the code of the latter respectively.

6.26. journal.sh (or .bat) Script

The `journal.sh (or .bat)` script is used to:

- generate event journals for a given repository;
- delete in the Arcsys Database the events that have been written in a journal for a given repository;
- count the remaining events to be written in a journal.

6.26.1. Generate a journal

The journals generated for an Arcsys repository are XML files with the following name model: `journal_<repositoryCode>_<sequenceNumber>.xml`. These journals are generated in the `journal` folder of the Arcsys Engine.

Note

The sequence number is a number linked to a repository, which is incremented each time a new journal is generated for that repository. Thus, the first journal generated for a given repository will be named `journal_<repositoryCode>_1.xml`. This number can be used to track the chaining of generated journals for a given repository, delete events linked to a given journal and to generate journals that have already been generated in the past (see below).

To generate a journal in Arcsys, run the following command:

```
journal.sh(.bat) generate -r <repositoryCode> [-s <sequenceNumber> | -l <limit>]
```

where:

- `repositoryCode` represents the code of the Arcsys repository for which the journal will be generated.
- `sequenceNumber` represents the sequence number of the journal to generate. This parameter must be used to generate a journal that has already been generated in the past and whose events have not been deleted from the Arcsys Database.
- `limit` represents the maximum number of events to be written in the journal. If this parameter is not specified, it will default to 10000.

6.26.2. Delete journal events

For database size considerations, it is advisable to regularly delete events that have been written to a journal that has itself been archived in Arcsys.

To delete the events for a repository, run the following command:

```
journal.sh(.bat) delete-events -r <repositoryCode> -s <sequenceNumber>
```

where:

- `repositoryCode` represents the code of the Arcsys repository for which the events should be deleted.
- `sequenceNumber` represents the sequence number of the journal from which events can be deleted.

6.26.3. Count remaining journal events

It is possible to obtain the number of remaining events to be written for a given repository. This information will be provided both in the console output and in the script's logs.

To obtain the number of remaining events to be written for a repository, run the following command:

```
journal.sh(.bat) count-events -r <repositoryCode>
```

where:

- `repositoryCode` represents the code of the Arcsys repository for which you want to obtain the number of remaining events to be written.

6.27. installService_engine.cmd Script

This script deploys the Arcsys Engine as a Windows service.

6.28. removeService_engine.cmd Script

This script uninstalls the application service as a Windows service.

7. Arcsys Application Agent

7.1. Overview

The Arcsys Application Agent is installed in a directory called `ArcsysApplicationAgent`.

7.2. Directory Structure

Type	Name	Description
	`launch_applicationAgent.sh (or .bat)`	Module launch script.
	`status_applicationAgent.sh (or .bat)`	Module status script.
	`shutdown_applicationAgent.sh (or .bat)`	Module stop script.
	`checksignature.sh (or .bat)`	Check signature script.
	`installService_applicationAgent.cmd (non systematic)`	Module Windows service installation script.
	`removeService_applicationAgent.cmd (non systematic)`	Module Windows service uninstallation script.
	`etc`	Directory containing the module configuration files.
	`APPAGENT.properties`	Main configuration file of the module.
	`log4j2.xml`	log4j configuration file.
	`policy`	This file contains the parameters of the module's Java class permissions.
	`THIRD_PARTY_SEARCH_ENGINE.properties`	This file contains the settings of the third party search engine.
	`jar`	Directory containing the module Arcsys Java libraries (JAR).
	`lib`	Directory containing the different external Java libraries used by the JNI required in the module.
	`libext`	Directory containing the different external Java libraries required in the module.

	`log`	Directory containing a list of all the log files. By default, these files are automatically rotated each day.
	`xml`	Directory containing the last version of the XML schema for describing the manifests and the object handlers.

7.3. etc/log4j2.xml file

7.3.1. Definition

This file contains the settings for the log file. The logs are generated with the log4J Apache library.

7.3.2. Parameters

The parameters of this file depend on the information desired in the logs. Hence please consult the standard documentation of Log4J for any changes.

See: https://logging.apache.org/log4j/2.x/

7.4. etc/APPAGENT.properties File

7.4.1. Definition

This file contains the parameters for the Arcsys Application Agent.

7.4.2. Component Code and Host Name Parameters

- For the *COMPONENT_CODE*, *COMPONENT_HOST* and *PARAMETER_AUTO_REFERENCE_MODE* parameters, see Codes for Modules, IP and host names.

7.4.3. Component API parameters

Parameter name	**COMPONENT_API_PORT_NUMBER**
Parameter type	Port number
Parameter change assessment	After module reboot.
Description	Port number on which the application agent API is deployed.
Shipped value	`25030`

Parameter name	**COMPONENT_API_SERVICE_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Name of the application agent API service.
Shipped value	`ArcsysApplicationAgentPrimary`

7.4.4. Directories and Paths

- *`APPLICATION_AGENT_INSTALL_FOLDER`*: Arcsys Application Agent installation directory;
- *`DIR_SCHEMAS`*: Directory name from the installation directory root of the Arcsys Application Agent where the different XML schema versions for the manifests and object handlers are stored. Default value: `xml`.

7.4.5. Archiving and retrieval

Parameter name	**ARCH_REPOSITORY_PATH**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Access path to the directory used to store the files to be archived. This path corresponds to the *`output_stage_dir`* directory of the transfer service. **Note** This directory is shared with Arcsys Transfer Service or Arcsys Transfer Server. Both modules must have access to it. For more details, please refer to the Arcsys Administration Manual, chapter "Setting rights on directories shared between Arcsys modules".
Shipped value	`${TARGET_FOLDER} (special variable that automatically takes the value of the parameter of the -DTARGET_FOLDER command line)/ArcsysTransferServer/stage/cli/out`

Parameter name	**REST_REPOSITORY_PATH**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Access path to the directory used to store the files to be retrieved. This path corresponds to the *input_stage_dir* directory of the transfer service. **Note** This directory is shared with Arcsys Transfer Service or Arcsys Transfer Server. Both modules must have access to it. For more details, please refer to the Arcsys Administration Manual, chapter "Setting rights on directories shared between Arcsys modules".
Shipped value	`${TARGET_FOLDER} (special variable that automatically takes the value of the parameter of the -DTARGET_FOLDER command line)/ArcsysTransferServer/stage/cli/in`

Parameter name	**WINDOWS_TO_UNIX_DEFAULT_RETRIEVAL_FOLDER**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Default retrieval folder on Unix for the files archived on Windows. When retrieving on Unix a file archived on Windows, the drive letter is removed. If the retrieval folder is not specified in the retrieval request, the file is retrieved in the folder specified by this parameter.
Shipped value	`/`

7.4.6. Access modes to files being archived and retrieved

Parameter name	**LOCAL_TRANSFER_SERVICES**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	List of the codes of the local transfer services (NFS mount included) for the application agent. The transfer servers are also transfer services: they can be included in this list. If a local code is not given, the *ARCH_REPOSITORY_PATH* and *REST_REPOSITORY_PATH* directories must contain the directories of a transfer service from which data will be transferred to a transfer server managing the storage zone. If a local code is given, the data will simply be moved from the *ARCH_REPOSITORY_PATH* and *REST_REPOSITORY_PATH* directories to the cache directories of the transfer servers.
Shipped value	`ArcsysTransferServerPrimary`

7.4.7. Communication with the engine

Parameter name	**ENGINE_API_LIST**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	List of addresses used for contacting the engine API. Each address has the following format `rmi://Engine host:Engine port/Engine code`. The parameter `Engine host` must contain an address from the application agent machine. It does not necessarily correspond to the value of the engine's physical host registered in the relational database and visible in the engine interface. The value `localhost` is accepted, for instance. The engine code can be found in the *`COMPONENT_CODE`* parameter of the engine configuration file; the engine port is specified in the *`COMPONENT_API_PORT_NUMBER`* parameter of the engine configuration file. Example: `rmi://localhost:25020/ArcsysPrimaryEngine`. If an engine in the list cannot be contacted, the agent will not start. An engine is considered unavailable if it does not exist in the repository or if it has not been updated for more than two minutes. This time can be set with the *`SEC_ENGINE_AVAILABLE_TIMEOUT`* parameter of the engine.
Shipped value	`rmi://localhost:25020/ArcsysEnginePrimary/`

Parameter name	**MAX_TRY_CONTACT_ENGINE**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Number of engine connection attempts on startup.
Shipped value	`5`

Parameter name	**DELAY_RETRY_CONTACT_ENGINE**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	Timeout in milliseconds for a new initialisation attempt to the engine after a failure.
Shipped value	`3000`

Parameter name	**PRIORITY_SITE_LIST**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	List of site codes. The order of the list is important: the engines of the first site in the list have priority for processing the archiving, retrieval or archive restitution requests for this agent. All engines on the same site have the same priority. If a site is not defined in the list, the request will not be processed by an engine of this site.
Shipped value	`PRIMARY_SITE`

Parameter name	**PROCESS_ONLY_IN_ENGINES**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	List of Arcsys Engines codes, corresponding to the Arcsys Engines authorized to process the requests of this Arcsys Application Agent. The Arcsys Engines must be part of the sites chosen in the *PRIORITY_SITE_LIST* parameter, otherwise they will be ignored. If no code is specified, all the Arcsys Engines of the sites selected in *PRIORITY_SITE_LIST* parameter will be considered. The engine code can be found in the *COMPONENT_CODE* parameter of the Arcsys Engine configuration file.
Shipped value	Empty (all the Arcsys Engines of the sites selected in *PRIORITY_SITE_LIST* parameter)

7.4.8. Securing RMI Access

For security parameters *COMPONENT_API_RMI_SECURITY_PROTOCOL*, *TRUSTSTORE_FILE*, *TRUSTSTORE_PASSWORD*, *KEYSTORE_FILE* and *KEYSTORE_PASSWORD*, see Configuring Arcsys to Manage SSL on LDAP or RMI.

7.4.9. Integrity parameters

Parameter name	**DEFAULT_HASH_FUNCTION**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the default encoding algorithm for generating the hash value responsible for the integrity of files. The algorithm is not necessarily identical to that chosen in the *MANIFEST_TAR_HASH_FUNCTION* parameter of the archiving engine. This default algorithm is only taken into account if the "Hash Function" field of the object does not have a value. No integrity management can be determined by selecting the value `NONE` but this is highly discouraged. Possible values: `MD5` / `SHA1` / `SHA256` / `SHA512` / `NONE` **Note** For end-user archiving, the hash type chosen in *FILE_UPLOADER_HASH_ALG* will be used.
Shipped value	`SHA512`

Parameter name	**VERIFY_REFERENTIAL_HASH_VALUE**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Indicates whether you want the application agent to perform a hash value check calculation when it has a value at object level in the relational database.
Shipped value	`true`

7.4.10. Parallelization Parameters

- *NB_MAX_SIMULTANEOUS_REQUESTS*: Maximum number of requests processed simultaneously by the Arcsys Application Agent. Default value: `10`;
- *NB_MAX_THREADS_BY_REQUEST*: Maximum number of threads for a given request. The thread is launched for each TAR file produced. Default value: `10`.

Note

The maximum number of threads that can run on the Arcsys Application Agent is thus:

```
NB_MAX_SIMULTANEOUS_REQUESTS*NB_MAX_THREADS_BY_REQUEST
```

7.4.11. Manifest

- *RESTITUTE_MANIFEST*: Retrieves the manifest during lot retrieval or archive restitution. Default value: FALSE. Possible values: TRUE / FALSE;

- *MANIFEST_DEFAULT_DIRECTORY*: Full retrieval path of the manifests. If this parameter is not given a value and *RESTITUTE_MANIFEST* is set to TRUE, the agent will be stopped during startup.

7.4.12. Object Handler

- *RESTITUTE_OBJECT_DESCRIPTOR*: Retrieves the object handlers during lot retrieval or archive restitution. Default value: FALSE. Possible values: TRUE / FALSE;

- *OBJECT_DESCRIPTOR_DEFAULT_DIRECTORY*: Full retrieval path of the object handlers. If this parameter is not given a value and *RESTITUTE_OBJECT_DESCRIPTOR* is set to TRUE, the agent will be stopped during start-up.

7.4.13. Managing UNIX/LDAP Rights

- *USE_POSIX_USER*: Indicates for UNIX if you prefer to not use the UNIX user that started up the application agent as the file read or write user, rather to use the user that created the archiving, retrieval or archival restitution request.

  ### Note

  Example of use: The Arcsys Application Agent is launched in root to access all directories of a file system, but users can only archive the data on the files that they have read rights for!

  The table below summarizes the behavior of Arcsys according to the value of the USE_POSIX_USER parameter:

  - for the file

USE_POSIX_USER=true	*USE_POSIX_USER*=false
■ Creation of the retrieved file with the UID / GID of the current Arcsys user ■ Creation of the missing directories with the UID / GID of the current Arcsys user ■ Permissions set to the permissions of the file when it was archived ■ atime (last access) set to the atime of the file when it was archived ■ mtime (last modification) set to the atime of the file when it was archived	■ Creation of the retrieved file with the UID / GID of the file when it was archived ■ Creation of the missing directories with the UID / GID of the application agent ■ Permissions set to the permissions of the file when it was archived ■ atime (last access) set to the atime of the file when it was archived ■ mtime (last modification) set to the atime of the file when it was archived

- for the manifest

USE_POSIX_USER=true	*USE_POSIX_USER*=false
■ Creation of the retrieved file with the UID / GID of the current Arcsys user ■ Creation of the missing directories with the UID / GID of the current Arcsys user ■ Default permissions when creating a file / directory on the OS (see umask for Unix) ■ atime (last access) set to the creation time of the manifest ■ mtime (last modification) set to the creation time of manifest	■ Creation of the retrieved file with the UID / GID of the application agent ■ Creation of the missing directories with the UID / GID of the application agent ■ Default permissions when creating a file / directory on the OS (see umask for Unix) ■ atime (last access) set to the creation time of the manifest ■ mtime (last modification) set to the creation time of manifest

- for the link

USE_POSIX_USER=true	*USE_POSIX_USER*=false
■ Creation of the retrieved link with the UID / GID of the application agent ■ Creation of the missing directories with the UID / GID of the current Arcsys user ■ Permissions of the link set 777 ■ atime (last access) set to the creation time of retrieved link ■ mtime (last modification) set to the creation time of retrieved link	■ Creation of the retrieved link with the UID / GID of the application agent ■ Creation of the missing directories with the UID / GID of the application agent ■ Permissions of the link set 777 ■ atime (last access) set to the creation time of retrieved link ■ mtime (last modification) set to the creation time of retrieved link

- for the ZIP file

USE_POSIX_USER =true/false
▪ Creation of the retrieved file with the UID / GID of the application agent ▪ Creation of the missing directories with the UID / GID of the application agent ▪ Default permissions when creating a file / directory on the OS (see umask for Unix) ▪ atime (last access) set to the creation time of the retrieved file ▪ mtime (last modification) set to the creation time of the retrieved file

- for the AFP file (complete retrieval)

USE_POSIX_USER=true	*USE_POSIX_USER*=false
▪ Creation of the retrieved file with the UID / GID of the current Arcsys user ▪ Creation of the missing directories with the UID / GID of the current Arcsys user ▪ Permissions set to the permissions of the file when it was archived ▪ atime (last access) set to the atime of the file when it was archived ▪ mtime (last modification) set to the atime of the file when it was archived	▪ Creation of the retrieved file with the UID / GID of the file when it was archived ▪ Creation of the missing directories with the UID / GID of the application agent ▪ Permissions set to the permissions of the file when it was archived ▪ atime (last access) set to the atime of the file when it was archived ▪ mtime (last modification) set to the atime of the file when it was archived

- for the AFP file (partial retrieval)

USE_POSIX_USER=true	*USE_POSIX_USER*=false
▪ Creation of the retrieved file with the UID / GID of the current Arcsys user ▪ Default permissions when creating a file / directory on the OS (see umask for Unix) ▪ atime (last access) set to the creation time of the retrieved file ▪ mtime (last modification) set to the creation time of retrieved file	▪ Creation of the retrieved file with the UID / GID of the file when it was archived ▪ Default permissions when creating a file / directory on the OS (see umask for Unix) ▪ atime (last access) set to the creation time of the retrieved file ▪ mtime (last modification) set to the creation time of retrieved file

7.4.14. Read Buffer Size

- *BUFFER_READ_SIZE*: Size (in bytes) of the read buffer of the Arcsys Application Agent data to be archived. This size notably affects indexing performance. Hence, it is wise to indicate a significant value for full text indexing of text documents. Default value: 32768.

7.4.15. Signing Agent (ArcSign Option)

- *SIGNING_AGENT*: Indicates whether an application agent signature is used or not. Default value: `FALSE`.
- *SIGNING_AGENT_CLASS*: Indicates the name of a class implementing the `SignatureClient` interface provided in the `arcsys_security.jar` of the Arcsys Application Agent. This class takes an Arcsys object to sign as a parameter and returns a signature in PKCS7 format. It then implements the following method:

```
public byte[] getObjectSignature(ArcsysObject arcsysObject) throws
 ArcsysSecurityException;
```

  where the *arcsysObject* parameter contains the hash of the object to sign, which can be accessed by the `getHash` method. The return of the method is a range of bytes representing the signature.

  Note

  This signature by external class cannot be used with the common signature file (*USE_COMMON_SIGNATURE_FILE* described below).

- *SIGNING_KEYSTORE_FILE*: Location of the keystore file containing the private key to be used for signing. You can inform it in the etc directory of the component, for example. This parameter is optional.
- *SIGNING_KEYSTORE_PASSWORD*: Password of the keystore file (this is the passphrase). This password must be encrypted. This parameter is optional.
- *SIGNING_PRIVATE_KEY_ALIAS*: Name of the alias matching the private key used for signing. This private key must be found in the keystore (given a value by the SIGNING_KEYSTORE_FILE parameter). This parameter is optional.

  Note

  Only one alias can be used at a time.

- *SIGNING_PRIVATE_KEY_PASSWORD*: Password of the private key used. This password must be encrypted. This parameter is optional.
- *USE_COMMON_SIGNATURE_FILE*: Activates the creation of the common signature file for signed records. This function is designed to reduce the number of timestamping tokens and to simplify signature implementation.
- *COMMON_SIGNATURE_FILE_FOLDER*: Indicates the folder path where the common signature files are retrieved where no destination folder is defined for a retrieval request.

7.4.16. Timestamping (ArcSign Connector)

- *TSA_CLASS* Indicates the name of the Java class to be used for communicating with the TSA server used for timestamping. Default value: `TSAClientBouncyCastle` (works for the majority of TSAs in accordance with standard RF3161).
- *TSA_URL*: URL of the TSA server used for timestamping.
- *TSA_CONNEXION_LOGIN*: Identifier of the connection to the TSA server, only if required. This parameter is optional.
- *TSA_CONNEXION_PASSWORD*: Password of the connection to the TSA server, only if required. This parameter is optional.
- *TSA_CONNEXION_KEYSTORE_TYPE*: Keystore type to be used for connecting to the TSA server (JKS, PKCS12, etc.) This parameter is optional.
- *TSA_CONNEXION_KEYSTORE*: Path of the keystore file for connecting to the TSA server. This parameter is optional.
- *TSA_CONNEXION_KEYSTORE_PASSWORD*: Keystore password for connecting to the TSA server. This parameter is optional.
- *TSA_CONNEXION_TRUSTSTORE_TYPE*: Truststore type to be used for connecting to the TSA server (JKS, PKCS12, etc.) This parameter is optional.
- *TSA_CONNEXION_TRUSTSTORE*: Path of the truststore file for connecting to the TSA server. This parameter is optional.
- *TSA_CONNEXION_TRUSTSTORE_PASSWORD*: Truststore password for connecting to the TSA server. This parameter is optional.
- *TSA_MAX_REQUEST_PER_DAY*: Maximum number of requests allowed to the TSA server per day. This parameter is optional. Default value: `0` (this is equivalent to no request limit).
- *TSA_MILLISECONDS_BETWEEN_REQUESTS*: Number of milliseconds between sending two requests to the TSA. Default value: `0` (this is equivalent to no wait between sending two requests).
- *TIMESTAMP_SOURCE_TYPE*: Source of the timestamped data for Arcsys. This property can have `DOCUMENT` as a value where the hash of the archived files is timestamped, or `DOCUMENT_SIGNATURE` where the hash of the file signatures is timestamped, if they exist; otherwise the behaviour is identical. The recommended value is `DOCUMENT_SIGNATURE` because it allows to timestamp the fact that a signature has been added at a given time. Default value: `DOCUMENT`.

7.4.17. Native Envelope

- *NATIVE_DEFAULT_DIRECTORY*: Default retrieval directory for native ZIP or LZMA envelopes, used if no retrieval directory is specified.
- *USE_LINK_FOR_NATIVE_MOVER*: Allows the use of symbolic links on systems that support them (Unix-like) in order to reduce archiving time of ZIP native envelopes.
- *VERIFY_NATIVE_REFERENTIAL_HASH_VALUE*: Reduces the archiving time of native envelopes by not checking the hash values of the native envelopes at the Arcsys Application Agent level if there are any in the relational database (this task is performed in all cases by the Arcsys Transfer Server).

7.4.18. Format control

Parameter name	**ARCFF_SERVER_URL**
Parameter taken into account only if:	A lot processed by this application agent has a class with a defined format policy
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Access URL of the server hosting the ArcFF format control module. The syntax for conventional HTTP exchanges is as follows: `http://host:port`. The syntax for SSL exchanges is as follows: `https://host:port`.
Shipped value	`http://localhost:8085`

Parameter name	**ARCFF_CLIENT_ID**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The identifier of the client application for OAuth2 authentication on the ArcFF format control module. This value must be identical to the value of the *SECURITY_CLIENT_ID* parameter in ArcFF format control module.
Shipped value	`arcff`

Parameter name	**ARCFF_CLIENT_SECRET**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	The secret shared by identifier of the client application for OAuth2 authentication on the ArcFF format control module. See the parameter *SECURITY_CLIENT_SECRET*.
Shipped value	No shipped value.

Parameter name	**ARCFF_SAME_HOST**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Flag to indicate if the ArcFF format control module is on the same server as the Arcsys Application Agent. If this is the case, the files transmitted between these two modules will not be uploaded (the file path will be used by the ArcFF format control module). **Important** **If the files processed by the ArcFF format control module may exceed 2 GB, it is required to install the ArcFF format control module on the same server as the Arcsys Application Agent and to set this flag to `true`. In this configuration, the local file path is used, avoiding file transfer size limitations.**
Shipped value	`false`

Parameter name	**ARCFF_SECURITY_PROTOCOL**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The protocol used for exchanges with the ArcFF format control module. Possible values are: • `NONE`: Interactions with the ArcFF format control module are conventional HTTP exchanges. • `SSL`: Exchanges with the ArcFF format control module are secured via SSL.
Shipped value	`NONE`

Parameter name	**ARCFF_TRUSTSTORE_TYPE**
Parameter taken into account only if:	*`ARCFF_SECURITY_PROTOCOL`*`= SSL`
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	For the exchanges with ArcFF format control module: Type of the truststore used for SSL/TLS exchanges.
Shipped value	`jks`

Parameter name	**ARCFF_TRUSTSTORE_FILE**
Parameter taken into account only if:	*ARCFF_SECURITY_PROTOCOL*= SSL
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	For the exchanges with ArcFF format control module : Location of the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**ARCFF_TRUSTSTORE_PASSWORD**
Parameter taken into account only if:	*ARCFF_SECURITY_PROTOCOL*= SSL
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	For the exchanges with ArcFF format control module: Password for the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

7.5. etc/THIRD_PARTY_SEARCH_ENGINE.properties File

7.5.1. Definition

This file contains the parameters for accessing the Third Party Search Engine for indexing or search. See its description in page 139, « Configuring ArcRFT Option ».

7.6. launch_applicationAgent.sh (or .bat) Script

This script starts up the Arcsys Application Agent. The Arcsys Application Agent will then connect to the Arcsys Engine to signal that it is active.

7.7. shutdown_applicationAgent.sh (or .bat) Script

This script stops the Arcsys Application Agent.

7.8. status_applicationAgent.sh (or .bat) Script

This script retrieves a return code matching the current status of the Arcsys Application Agent.

Return code value	Application Agent status	Description
43	STARTED_NOT_WORKING	The agent is started but not handling any requests. (`launch_applicationAgent` script launched).
44	STARTED_AND_WORKING	The agent is started and handling requests (`launch_applicationAgent` script launched).
46	SHUTDOWN	The agent is stopped.
49	GET_STATUS_ERROR	Error in retrieving the status.

7.9. installService_applicationAgent.cmd Script

This script deploys the Arcsys Application Agent as a Windows service.

7.10. removeService_applicationAgent.cmd Script

This script uninstalls the application service as a Windows service.

7.11. checksignature.sh (or .bat) Script

This script checks the status of internal signature (performed by the Arcsys Application Agent) for all the documents of a given archive.

The script is in the Arcsys Application Agent because it can be called after having first performed a retrieval of the archive, and its manifest (by using *`RESTITUTE_MANIFEST`* option of the Arcsys Application Agent) in a folder.

The script is used as follows:

```
checksignature.sh <pathOfXMLManifest> [<restoreFolder>] [<partOfThePathToIgnore>]
```

where:

- `<PathOfXMLManifest>` is the path leading to the XML Manifest of the archive.
- `<restoreFolder>` is the target folder in which the retrieval request was performed. This parameter is optional. If it is specified, the checksums of the files of the archive are re-computed and compared with the values mentioned in the manifest.
- `<partOfThePathToIgnore>` is the "path substitution" value if the archive was restored with a "path substitution" option (which is possible when retrieving an archive through the GUI or the various APIs, and allows to "remove" a part of the path when extracting the files). This parameter is optional.

Given these parameters, it reads the content of the manifest, and, for each object that is mentioned in the manifest (including the common signature file):

- If the restore folder is specified, the script *compares* the given checksum of the object with the real checksum of this object (that must be present inside the restore folder with the path given in the manifest, optionally reduced by the "Part of the path to ignore" parameter); if it is equals, there is a message "The computed checksum for object ID[...] is the same as the stored checksum", otherwise "!!! The computed checksum for object ID[...] is not the stored checksum ! "
- It *verifies* the digital signature of this object. If it is correct, there is a message "--> The signature in the manifest for object ID[...] is correct." otherwise "!!! The signature in the manifest for object ID[...] is not correct. "

Here is an example of command to call `checksignature.sh` :

```
checksignature.sh arcsys42super_ZQW_manifest_lot_43.arcsys.xml /home/arcsys/
ArcsysApplicationAgent/restoreFolder
```

8. Arcsys Transfer Service

The Arcsys Transfer Service is installed in a directory called `ArcsysTransferService`.

8.1. Directory Structure

Type	Name	Description
	`launch_transferService.sh (or .bat)`	Module launch script.
	`status_transferService.sh (or .bat)`	Module status script.
	`shutdown_transferService.sh (or .bat)`	Module stop script.
	`reload_transferService.sh` (or .bat)	This script is used to reload specific parts of the Arcsys Transfer Service configuration. It can be used to reload the log configuration, to reload the `notify.conf` settings or to rotate the Arcsys Transfer Service log file.
	`rotate_logs_transferService.sh` (or .bat)	This script triggers the rotation of the Arcsys Transfer Service logs.
	`installService_transferService.cmd (non systematic)`	This script installs the Arcsys Transfer Service as a Windows service. This script is specific to the Windows platform.
	`removeService_transferService.cmd (non systematic)`	This script uninstalls the Arcsys Transfer Service as a Windows service. This script is specific to the Windows platform.
	`bin`	Contains the executables for the installation platform.
	`etc`	Directory containing the module configuration files.
	`LogParam.cfg`	This file contains the parameters of the logs.
	`transferService.conf`	Main settings file of the module.
	`lib`	Contains the shared and dynamic libraries used by the module.

	`log`	Directory containing a list of all the log files. By default, these files are automatically rotated each day.
	`stage`	Directory tree where the files sent and received are stored.

8.2. transferService.conf File

This file contains the parameters for the Arcsys Transfer Service.

8.2.1. Component parameters

To make the system open, the application agents, engines, transfer servers and transfer services have a unique code that is used to identify them in the configuration files or in the Arcsys interface. This way, the host of a module can be changed without it impacting the other modules. This code is determined by the administrator, who will choose a code with meaning for the context.

Parameter name in `transferService.conf`	**component_code**
Parameter type	Characters string
Description	Name of this component. To avoid duplication, this name must be unique (on all sites). It is recommended to use a "logical" name distinct from a physical host name for example. Unique component code. The code must identify the module in the Arcsys installation.
Shipped value	`ArcsysTransferServicePrimary`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferService.conf`	**component_port_number**
Parameter type	Characters string
Description	Transfer Service port number Port number on which the transfer service is deployed. For compatibility, this parameter also accepts a service name defined in /etc/services.
Shipped value	`25010`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferService.conf`	**component_host**
Parameter type	Characters string
Description	Optional component host (to force the address of the component). If not empty, this host overrides the host chosen automatically. You can set there an IP, a host name...this address will be the one used from the other modules.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferService.conf`	**external_component_host**
Parameter type	Characters string
Description	Optional external component host. If set, this host will be used by components that are not in the same site instead of the component_host.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferService.conf`	**component_auto_reference_mode**
Parameter type	State variable. Possible values: `HOSTNAME`, `IP`
Description	Automatic mode of referencing. This parameter is used only if the parameter `component_host` is not filled.
Shipped value	`HOSTNAME`
Parameter taken into account only if:	`component_host` =
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

8.2.2. Stage directories

These are the working spaces

Parameter name in `transferService.conf`	**input_stage_dir**
Parameter type	Characters string
Description	File reception directory. Usually in /stage/cli/in. Absolute file path. Use Unix like separators ('/'), even on Windows. This directory is shared with Arcsys Application Agent. Both modules must have access to it.
Shipped value	`${TARGET_FOLDER}/ArcsysTransferService/stage/cli/in`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferService.conf`	**output_stage_dir**
Parameter type	Characters string
Description	File sending directory. Usually in /stage/cli/out. Absolute file path. Use Unix like separators ('/'), even on Windows. This directory is shared with Arcsys Application Agent. Both modules must have access to it.
Shipped value	`${TARGET_FOLDER}/ArcsysTransferService/stage/cli/out`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

8.2.3. General parameters

Parameter name in `transferService.conf`	**transfer_server_list**
Parameter type	List of string separated by commas
Description	List of addresses of Transfer Servers: the first addresses being the first to be contacted, and being given the most priority for registering the Transfer Service. The pattern of a Transfer Server URL is : <Host or IP or localhost>:<Port> of the Transfer Server An example of Transfer Server URL is 192.168.43.222:25010 List of addresses on which to contact the transfer server on startup of the transfer service. The transfer service contacts a transfer server to register the transfer server in the relational database. This parameter can list all the transfer servers installed, irrespective of the site to which they are attached. The transfer servers will be contacted in their order of definition, until a connection leads to registering the service in the repository (addition or update).

	Each address has the following format "Transfer server host:Transfer server port". The parameter "Transfer server host" must contain an address known from the transfer service machine. It does not necessarily correspond to the value of the physical host of the transfer server registered in the relational database and visible in the transfer servers. The value "localhost" is accepted, for instance. The "Transfer server port" is specified in the `component_port_number` parameter of the configuration file of the transfer server. Example(s): 192.168.43.222:25010,192.168.43.212:25010
Shipped value	
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

8.2.4. TCP clients

Parameter name in `transferService.conf`	**max_tcp_cli**
Parameter type	Positive integer
Description	Defines the maximum number of simultaneous TCP clients managed by the Transfer Service Maximum number of TCP clients accepted simultaneously.
Shipped value	25
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

Parameter name in `transferService.conf`	**max_tcp_retry**
Parameter type	Positive integer
Description	Maximum number of retry before considering all clients TCP has failed Maximum number of attempts for communication with a remote Arcsys Transfer Server (for registering purpose, at launch of the Arcsys Transfer Service).
Shipped value	3

Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferService.conf`	**delay_tcp_retry**
Parameter type	Number of seconds
Description	Retry delay before new attempt connection to server daemon Delay between two contact attempts to a remote Arcsys Transfer Server (for registering purpose, at launch of the Arcsys Transfer Service).
Shipped value	`5`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.
Minimum value	1

8.2.5. Code set

The following parameters are only taken into account in Windows environment.

Section of parameters taken into account only if: `OS` = `windows`

Parameter name in `transferService.conf`	**local_charset**
Parameter type	Characters string
Description	Local system codeset Factor indicating the local codeset of the character strings of the platform containing the Arcsys Transfer Service. This factor is required to convert the character strings used by the Arcsys Transfer Service on Windows to the local codeset. To display the system locale used, type "chcp" in DOS.
Shipped value	`850`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferService.conf`	**daemon_charset**
Parameter type	Characters string
Description	Daemon char strings used codeset - usually UTF-8

	Factor indicating the codeset to which the character strings transmitted by the network should be converted.
Shipped value	`UTF-8`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

8.2.6. Communication protocol

Security Parameters

Important

The security parameters (protocol type, certificates) on the Arcsys Transfer Service must be coherent with those of the Arcsys Transfer Servers communicating with it.

Parameter name in `transferService.conf`	**security_protocol**
Parameter type	State variable. Possible values: `NONE`, `INTERNAL`, `SSL`
Description	Security protocol Indicates if communication to and from the transfer server is encrypted or not, and the type of encryption (symmetrical for INTERNAL: Blowfish algorithm is used; or asymmetrical for SSL). Disabling encryption increases performance but may be forbidden depending on the required security level. **Note** The value must be the same between all clients and servers exchanging data. It is not mandatorily the same on a global Arcsys installation.
Shipped value	`NONE`
Parameter change assessment	After module reboot.
Required	True, you have to fill this parameter.

Parameter name in `transferService.conf`	**private_key_filename**
Parameter type	File path
Description	Path of the file containing the private key.

	Absolute file path. Use Unix like separators ('/'), even on Windows. Name of the file containing the private key(s) for the INTERNAL encryption mode only.
Shipped value	**No shipped value.**
Parameter taken into account only if:	`security_protocol` = `INTERNAL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferService.conf`	**ssl_keystore_filename**
Parameter type	File path
Description	Path of the PKCS#12 keystore. Absolute file path. Use Unix like separators ('/'), even on Windows. Location of the keystore containing the certificate used for SSL/TLS exchanges. The file must mandatorily be in PKCS12 format.
Shipped value	**No shipped value.**
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferService.conf`	**ssl_keystore_password**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Encrypted password of the PKCS#12 keystore Password of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	**No shipped value.**
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferService.conf`	**ssl_truststore_filename**
Parameter type	File path
Description	Path of the PKCS#12 truststore. Absolute file path. Use Unix like separators ('/'), even on Windows. Location of the truststore used for SSL/TLS exchanges. The file must mandatorily be in PKCS12 format.
Shipped value	**No shipped value.**
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferService.conf`	**ssl_truststore_password**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility (`generate-key` and `encrypt` tools) provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Description	Encrypted password of the PKCS#12 truststore Password for the truststore used for SSL/TLS exchanges.
Shipped value	
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferService.conf`	**ssl_library_libssl**
Parameter type	File path
Description	Optional path of the not provided dynamic library libssl if not standard. By default: on Linux, the library will be loaded by the path `libssl.so`. On Windows, the library will be loaded first by the path `ssleay32.dll` (Microsoft library), then `libssl32.dll` and then `libssl.dll`. Absolute file path. Use Unix like separators ('/'), even on Windows. Optional path of the libssl library used for SSL communication. This path will not be taken into account for other features requiring libssl (ArcCrypt Option uses the system installation). The version of this libssl library must be between 1.1.0 and 1.1.2.

	By default, the loaded libssl library path is `libssl.so` on Linux, and on Windows `ssleay32.dll` is first loaded, then in case of failure, `libssl32.dll` followed by `libssl.dll`. It is recommended to use this parameter if there are several versions of libssl installed on the system. Example(s): /usr/lib64/libssl.so.3
Shipped value	**No shipped value.**
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferService.conf`	**ssl_library_libcrypto**
Parameter type	File path
Description	Optional path of the not provided dynamic library libcrypto if not standard. By default: on Linux, the library will be loaded by the path `libcrypto.so`. On Windows, the library will be loaded first by the path `libeay32.dll` (Microsoft library), then `libcrypto32.dll` and then `libcrypto.dll`. Absolute file path. Use Unix like separators ('`/`'), even on Windows. Optional path of the libcrypto library used for SSL communication. By default, the loaded libcrypto library path is `libcrypto.so` on Linux, and on Windows `libeay32.dll` first, then in case of failure, `libcrypto32.dll` followed by libcrypto.dll. Example(s): /usr/lib64/libcrypto.so.3
Shipped value	**No shipped value.**
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferService.conf`	**ssl_module_path**
Parameter type	Characters string
Description	Optional path to the directory containing legacy cryptographic modules (e.g., `legacy.dll`/`legacy.so`) used by the SSL module. This may be required if your system relies on older cryptographic algorithms or libraries for compatibility reasons.

	Absolute file path. Use Unix like separators ('/'), even on Windows.
Shipped value	
Parameter taken into account only if:	`security_protocol` = `SSL`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

Parameter name in `transferService.conf`	**ip_whitelist**
Parameter type	List of string separated by commas
Description	List of IPs authorized to contact the Transfer Service. If empty, all IPs are authorized. This list defines the workstations that can send requests to the Arcsys Transfer Service. This parameter is composed of a comma-separated list of addresses. If this parameter has a value, only the requests from the workstations whose IP appears in the list will be authorized. Other requests will not receive a response. The ip_whitelist parameter can be left blank, in which case all connection requests will be authorized. If this parameter is used, the IP of the site on which the service is launched must be added to run the status, reload and shutdown scripts. According to the site configuration, this IP will be the local IP (generally, 127.0.0.1) or the network IP (e.g. 192.168.40.120). Example(s): To restrict communication between an Arcsys Transfer Server on a site where the IP is 192.168.40.120 and a service on a site where the IP is 192.168.40.130, the parameters are: - for the server: ip_whitelist=192.168.40.120 - for the service: ip_whitelist=192.168.40.130 The Arcsys Transfer Server can contact the service on site 192.168.40.130. In any case, since the IP of the server is not on its list of authorized IPs, this service will not answer the request.
Shipped value	**No shipped value.**
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

8.2.7. Alerts

To monitor Arcsys operation

Parameter name in `transferService.conf`	**notify_conf**

Parameter type	Characters string
Description	Alert configuration file path. Absolute file path. Use Unix like separators ('/'), even on Windows. To monitor Arcsys operation, the various modules can be configured to issue alerts to a monitoring server (Centreon, Nagios, etc.) in the event of error or a change in status.
Shipped value	`${TARGET_FOLDER}/ArcsysTransferService/etc/notify.conf`
Parameter change assessment	After module reboot.
Required	False, this parameter is optional.

8.3. LogParam.cfg File for Arcsys Transfer Service

This section describes the configuration of the log files generated by the Arcsys Transfer Service. Arcsys uses a proprietary standard for these components, which is based on the Log4J standard. The `LogParam.cfg` file found in the `etc/` directory of the Arcsys Transfer Server and the Arcsys Transfer Service enables configuration of these logs. Any configuration error is indicated in the output file of the logs. The rows starting with # and empty rows are ignored. The definition of each variable is optional. Where a variable is not defined, a default value will be associated.

Parameter name in `LogParam.cfg`	**LogLevel**
Parameter type	State variable. Possible values: `DEBUG`, `INFO`, `NOTICE`, `WARN`, `ERROR`, `FATAL`
Description	Defines the level of the logs to write. If the level configured is `ERROR`, only `ERROR` or `FATAL` logs are taken into account.
Shipped value	`INFO`
Default value	`NOTICE`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `LogParam.cfg`	**ConsoleAppender**
Parameter type	State variable. Possible values: `yes`, `no`
Description	Output on stdout of the main process
Shipped value	`no`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `LogParam.cfg`	**FileAppender**
Parameter type	State variable. Possible values: `yes`, `no`
Description	Writes logs in a file
Shipped value	`yes`
Default value	`no`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `LogParam.cfg`	**RotateLogDaily**
Parameter type	State variable. Possible values: `yes`, `no`
Description	Indicates whether logs should be rotated each day at midnight
Shipped value	`yes`
Default value	`no`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `LogParam.cfg`	**File**
Parameter type	File path
Description	Defines the name and path of the file where the logs will be recorded. The path can be absolute or relative. If this variable is not defined or the file defined is incorrect, the default log file will be used.
Shipped value	`${TARGET_FOLDER}/ArcsysTransferService/log/arcsys-transfer-service.log`
Default value	`arcsys-transfer-service.log`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `LogParam.cfg`	**WinEventLogAppender**
Parameter type	State variable. Possible values: `yes`, `no`
Description	Writes logs in a Windows event log
Shipped value	`no`

Parameter taken into account only if:	`OS` = `Windows`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

Parameter name in `LogParam.cfg`	**WinEventLogSource**
Parameter type	Characters string
Description	Defines the source name of the logs to write in the event log.
Shipped value	`ELEMNISCATE`
Parameter taken into account only if:	`WinEventLogAppender` = `yes`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	True, you have to fill this parameter.

Parameter name in `LogParam.cfg`	**layout.ConversionPattern**
Parameter type	Characters string
Description	Defines the content of the format of the logs generated. The different formats allowed are as follows: %t: Call thread identifier; %i: Call process identifier (PID); %p: Priority level of the message to be logged; %m: Message to log; %n: Line skip character; %d: Date and time in ISO 8601 format (e.g. 2000-09-07 14:07:41); %d(format_date): Date and/or time in the format defined by format_date; Authorized expressions to set the date/time format: - yyyy: Year in 4 digits, - yy: Year in 2 digits, - MM: Month in 2 digits, - MMM: Month abbreviation, - MMMM: month full name,

	- dd: Day of the month in 2 digits (01-31), - D: Number of the day in the year (001-365), - HH: Time in 24 hours (00-23), - K: Time in 12 hours (0-11), - mm: Minutes (00-59), - ss: Seconds (00-59). %u: milliseconds; **Important** **Curly brackets cannot be used in the date format because this character finishes the date format.**
Shipped value	`%p %d{dd-MM-YYYY HH:mm:ss},%u [TID:%t PID:%i] %m`
Default value	`%i.%t.%d.%p %m`
Parameter change assessment	After module reboot or using the reload script of the component(See the section reload_transferServer.sh (or .bat) Script).
Required	False, this parameter is optional.

8.4. Scripts

The Arcsys Transfer Service is started up or stopped through scripts or in the case of Windows, through a Windows service. In the latter case, prior setup should be performed for these operations to function correctly.

8.4.1. launch_transferService.sh (or .bat) Script

This script starts up the Arcsys Transfer Service.

8.4.2. reload_transferService.sh (or .bat) Script

This script is used to reload specific parts of the Arcsys Transfer Service configuration while the Arcsys Transfer Service is running. It can be used to reload the log configuration, to reload the `notify.conf` settings or to rotate the Arcsys Transfer Service log file.

The script is used as follows:

```
reload_transferService.sh [<sectionName>]
```

The name of the section is optional. If specified, it is possible to indicate only one each time.

The possible sections are:

- `log`: reloads the log configuration given in the file `LogParam.cfg`. When `log` section is specified, there is no rotation of the Arcsys Transfer Service logs. If the rotation of the Arcsys Transfer Service logs is requested, `rotate` section must be used.
- `rotate`: reloads the log configuration given in the file `LogParam.cfg` and triggers the rotation of the Arcsys Transfer Service logs.
- `notify`: reloads the `notify.conf` settings (see Page 47).

When no section is specified, the script reloads the log configuration given in the file `LogParam.cfg` and the `notify.conf` configuration file. Otherwise, the script only performs the action corresponding to the specified section.

8.4.3. rotate_logs_transferService.sh (or .bat) Script

This script triggers the rotation of the Arcsys Transfer Service logs. This rotation can be done as a planned task or a cron task. Remember that the parameter:

```
RotateLogDaily=yes
```

(default value: `no`) at `LogParam.cfg` level enables automatic rotation each day at midnight.

8.4.4. shutdown_transferService.sh (or .bat) Script

This script stops the Arcsys Transfer Service.

8.4.5. status_transferService.sh (or .bat) Script

This script sends a return code matching the current status of the Arcsys Transfer Service.

Return code value	Arcsys Engine status	Description
23	STARTED_NOT_WORKING	The service is started but not handling any requests. (`launch_transferService` script launched).
24	STARTED_AND_WORKING	The service is started and handling requests.
25	SHUTTING_DOWN	The service prepares to stop (`shutdown_transferService` script launched)
26	SHUTDOWN	The service is stopped.

8.4.6. Windows Systems

In Windows, the Arcsys Transfer Service can be run in two ways:

- Windows application
- Windows service

If you wish to start the Arcsys Transfer Service in "Windows Service" mode, you must install the Windows service by starting the `installService_transferService.cmd` script.

8.4.6.1. installService_transferService.cmd Script

This script installs the Arcsys Transfer Service as a Windows service. This script is specific to the Windows platform.

For more details, please refer to the Arcsys Installation Manual.

8.4.6.2. removeService_transferService.cmd Script

This script uninstalls the Arcsys Transfer Service as a Windows service. This script is specific to the Windows platform.

For more details, please refer to the Arcsys Installation Manual.

9. Arcsys Web Agent

9.1. Overview

The Arcsys Web Agent is installed in a directory called `ArcsysWebAgent`. The structure of this module depends on the application server used.

9.2. Directory Structure

Type	Name	Description
	`ant` (non systematic)	Directory containing a complete version of Ant, which is useful for installation and updates.
	`war`	Directory containing the war to deploy in the web server
	`etc`	Directory containing the module configuration files.
	`arcsysJAAS.conf`	This file contains the parameters of the Java authorization and authentication service.
	`log4j2.xml`	log4j configuration file.
	`REFERENTIAL_JDBC.properties`	This file contains the parameters for accessing the Arcsys Database in JDBC mode.
	`TEMPLATE_MAIL_ContactManager.properties`	This file is the mail template sent by the End User Interface in an administrator contact request.
	`TEMPLATE_MAIL_CreateAccount.properties`	This file is the mail template sent by the End User Interface in a request to the administrator to open an account.
	`THIRD_PARTY_SEARCH_ENGINE.properties`	This file contains the settings of the third party search engine.
	`WEBAGENT_AccountCreation_<Language Prefix>.properties`	This file configures the selection screen for requesting the administrator to open an Arcsys account.
	`WEBAGENT_AnomalyCodes_<Language Prefix>.properties`	This file contains the error – Arcsys error message

		equivalences in English (en_GB) or French (fr_FR).
	WEBAGENT_configuration.properties	Main configuration file of the module.
	WEBAGENT_guipreferences_example.xml	Sample file of preferences for information display. This file can be used as a model to create global and user preferences, specific user preferences, and specific group preferences.
	WEBAGENT_JobActionCodes_<Language Prefix>.properties	This file contains the code – Arcsys error message equivalences in English (en_GB) or French (fr_FR)
	WEBAGENT_Label_<Language Prefix>.properties	These files contain all the labels displayed on the web in English (en_GB) or French (fr_FR).
	libext	Directory containing the different external Java libraries required in the module.
	log	Directory containing a list of all the log files. By default, these files are automatically rotated each day.
	trace	Directory containing a list of all the trace files. By default, these files are automatically rotated each day.
	xml	Directory containing the XML validation schemas used to save searches.

9.3. etc/log4j2.xml file

9.3.1. Definition

This file contains the settings for the log file. The logs are generated with the log4J Apache library.

9.3.2. Parameters

The parameters of this file depend on the information desired in the logs. Hence please consult the standard documentation of Log4J for any changes.

See: https://logging.apache.org/log4j/2.x/

9.4. etc/REFERENTIAL_JDBC.properties File

This file contains the parameters for accessing the Arcsys Database in JDBC mode. See the description in page 123, « Configuring JDBC Access to the Relational Database ».

9.5. etc/THIRD_PARTY_SEARCH_ENGINE.properties File

9.5.1. Definition

This file contains the parameters for accessing the Third Party Search Engine for indexing or search. See its description in page 139, « Configuring ArcRFT Option ».

9.6. etc/WEBAGENT_configuration.properties File

This file contains the parameters for the Arcsys Web Agent.

9.6.1. Global parameters

Parameter name	**SITE_CODE**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Site code to which the module must belong to. This site must be created in the Arcsys Web beforehand (by default, the installer creates systematically a `PRIMARY_SITE` code site).
Shipped value	`PRIMARY_SITE` (primary site created by default during the installation step).

- *`LABEL_FILE_DIR`* : Access directory to label files for the web. The label files are files called `WEBAGENT_label_xx_XX.properties` ;
- *`LANGUAGE_ALLOWED_LANGUAGE_CODE_x`* : Language code of the label to generate. When the user clicks on the corresponding flag icon (see below), this is the language that will be used. Example: fr.
- *`LANGUAGE_ALLOWED_COUNTRY_CODE_x`* : Country code of the label to generate. When the user clicks on the corresponding flag icon (see below), this is the country that will be used. Example: FR.

Note

In the example given for language code fr and country code FR, the label file loaded in the *`LABEL_FILE_DIR`* directory will be `WEBAGENT_label_fr_FR.properties` .

- *LANGUAGE_ALLOWED_IMG_PATH_x* : Path of the corresponding flag icon. A list of flags is available by default in the `img/flags` directory. Example: `img/flags/FR.png` .

- *MILLISEC_TRANSFER_SERVER_COMMUNICATION_TIMEOUT* : Defines the default value for the contact timeout of the transfer servers. If no response has been received from the transfer server after this time, the connection is considered inactive. This parameter must be adjusted in certain cases (transfer server on a remote network, etc.) Default value: `1000` .

9.6.2. General parameters for communication with the Arcsys Transfer Server

Parameter name	**TRANSFER_SERVERS**
Parameter type	Characters string list, separated by a comma (",").
Parameter change assessment	After module reboot.
Description	List of the transfer server codes with which the module must communicate for synchronous retrieval and creating/deleting file systems. For synchronous retrieval, the request is tried successively on the transfer servers of the list (in the order they are listed), until it succeeds. For creating/deleting file systems, one of the reachable transfer servers is contacted. This parameter is not mandatory. If left blank, all transfer servers of the site will be taken into consideration (without specific order).
Shipped value	No shipped value.

Parameter name	**MILLISEC_TRANSFER_SERVER_COMMUNICATION_TIMEOUT**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	Timeout after which communication without a reply to the transfer server triggers an error.
Shipped value	`1000`

9.6.3. Available Flags

Figure 9.1. Flags

LR.png
MR.png
NU.png
RE.png
SO.png
TV.png
YE.png
LS.png
MS.png
NZ.png
RO.png
SR.png
TW.png
YT.png
LT.png
MT.png
OM.png
RR.png
ST.png
TZ.png
YU.png
LU.png
MU.png
OS.png
RS.png
SV.png
UA.png
ZA.png
LV.png
MV.png
PA.png
RU.png
SY.png
UG.png
ZH.png
LY.png
MW.png
PE.png
RW.png
SZ.png
UM.png
ZM.png
MA.png
MX.png
PF.png
SA.png
TC.png
US.png
ZW.png
MC.png
MY.png
PG.png
SB.png
TD.png
UY.png
MD.png
MZ.png
PH.png
SC.png
TF.png
UZ.png
ME.png
NA.png
PK.png
SD.png
TG.png
VA.png
MG.png
NC.png
PL.png
SE.png
TH.png
VC.png
MH.png
NE.png
PM.png
SG.png
TJ.png
VE.png
MK.png
NF.png
PN.png
SH.png
TK.png
VG.png
ML.png
NG.png
PR.png
SI.png
TL.png
VI.png
MM.png
NI.png
PS.png
SJ.png
TM.png
VN.png
MN.png
NL.png
PT.png
SK.png
TN.png
VU.png
MO.png
NO.png
PW.png
SL.png
TO.png
WF.png
MP.png
NP.png
PY.png
SM.png
TR.png
WS.png
MQ.png
NR.png
QA.png
SN.png
TT.png
XY.png

9.6.4. Parameters for synchronous retrieval

Parameter name	**ONLINE_CHECK_INTEGRITY**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Attribute defining whether an integrity check of the **record** should be done during synchronous retrieval.
Shipped value	`true`

Parameter name	**ONLINE_CHECK_FILE_INTEGRITY**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Attribute defining whether an integrity check of the **document** should be done during synchronous retrieval.
Shipped value	`true`

Parameter name	**FILENAME_PATTERNS_FOR_CONTENT_TYPE**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Attribute used to "force" HTTP "ContentType" for files complying with the given patterns (for example, you want to force all the *-SAP-* files to be interpreted as `text/plain type`). The parameter is a <File name pattern, ContentType> sequence separated by semi-colons. Characters that can be used for the pattern: `?` (any character), `*` (0 or more than one character), `[abc]` for any character from the set {a,b,c} `[a-z]` for any character from the range a-z, etc. **Important** **If a document enters in the selection criteria of this parameter, the MIME type set takes precedence over the MIME type of the document set by the ArcFF format control module or that obtained by the parameter *`INFER_CONTENT_TYPE_FROM_FILENAME`*.**
Shipped value	No shipped value.

Parameter name	**INFER_CONTENT_TYPE_FROM_FILENAME**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Attribute defining whether, during synchronous retrieval, the HTTP "ContentType" should be explicitly set according to the file extension. In this case, approximately 70 different extensions (txt, htm, gif, jpg, png, pdf, rtf, doc, xls, ppt, odt, wav, avi, etc.) are managed by default. The content type is usually automatically inferred by the browser, but in certain cases this may be necessary. **Important** **If this parameter is set to `true` and a document enters in the selection criteria of this parameter, the MIME type obtained by *`FILENAME_PATTERNS_FOR_CONTENT_TYPE`* first takes precedence, and then the MIME type of the document set by the ArcFF format control module.**
Shipped value	`false`

Parameter name	**ARCHIVE_DOWNLOAD_MAX_OBJECT**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Determines the maximum number of objects that can be downloaded from the web interface.
Shipped value	`100`

9.6.5. Parameters for creating batch requests

Parameter name	**POSTPONE_ALL_LONG_OPERATIONS**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Defines whether a potentially long operation (deletion of a collection, a lot, a zone or a pool; modification of a mask) must trigger a postponed request or not. When set to `false`, the operation is run synchronously in the Arcsys Web Agent. When set to `true`, the postponed request is processed asynchronously in the Arcsys Engine. See Postponed request for more information.
Shipped value	`false`

9.6.6. Parameters for logon

Parameter name	**AUTHENTICATION_TYPE**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the type of authentication used. Possible values are: • `LDAP` • `HTTP_HEADER_AUTHENTICATION` • `SAML2` For LDAP parameters (preset by LDAP_), see Description of Different LDAP Parameters.
Shipped value	LDAP

Parameter name	**MAPPING_GROUP_URL**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the LDAP/URL group associations. This field must have a value respecting the following format: group1,idUrl1|group2,idUrl2. This association determines to which page a user is directed after being authenticated. URLS are defined through *`REDIRECT.URL.<idUrl>`* properties.
Shipped value	No shipped value.

Parameter name	**REDIRECT.URL.idUrl**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the URL associated to the LDAP groups according to their identifiers (format REDIRECT.URL.<idUrl>).
Shipped value	No shipped value.

Parameter name	**USE_LOGIN_AUTOCOMPLETE**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Defines if the "Identifier" field of the login page uses auto-complete. If yes, the browser displays a scroll list of values already selected for this field, and the password will be given by default. As a security measure, it is recommended that you do not use auto-complete.
Shipped value	`false`

9.6.7. Parameters for external authentication

9.6.7.1. HTTP header authentication

This authentication mode is used when the *AUTHENTICATION_TYPE* parameter has the HTTP_HEADER_AUTHENTICATION value. In this case, the authentication is done through an HTTP request. This HTTP request must target the TrtIdentification.jspa (http://<host>:<port>/webarcsys/TrtIdentification.jspa) page of the web agent. The connection page will not appear and the user identifier is specified in an HTTP header. The connection of this user is done through the administrator LDAP user. For this reason, the *LDAP_ADMIN_LOGIN* (without using the "anonymous" user) and *LDAP_ADMIN_PASSWORD* parameters must be set with this connection mode.

Parameter name	**HTTP_HEADER_AUTHENTICATION_NAME**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*= HTTP_HEADER_AUTHENTICATION
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the name of the HTTP header name used to retrieve the user name in external authentication.
Shipped value	SM_USER

9.6.7.2. SAML2 authentication

This authentication mode is used when the *AUTHENTICATION_TYPE* parameter has the SAML2 value. For security reasons, some Identity Providers need to know authentication and logout endpoints. Below are the endpoints used by Arcsys Web Agent:

- Authentication endpoint: http://<host>:<port>/webarcsys/TrtIdentification.jspa
- Logout endpoint: http://<host>:<port>/webarcsys/logout.jspa

Parameter name	**SAML2_PROTOCOL_BINDING**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the protocol binding for SAML2.
Shipped value	urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect

Parameter name	**SAML2_PROTOCOL_VERSION**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the protocol version for SAML2.
Shipped value	2.0

Parameter name	**SAML2_AUTHN_CONTEXT_CLASS_REF**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the authentication context (type) for SAML2 (password, kerberos, smartcard, token, specific authentication type, ...).
Shipped value	No shipped value.

Parameter name	**SAML2_ARCSYS_ISSUER**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the Arcsys issuer for SAML2.
Shipped value	No shipped value.

Parameter name	**SAML2_IDP_ISSUER**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the identification provider issuer for SAML2.
Shipped value	No shipped value.
Example	*SAML2_IDP_ISSUER*=http://adfs.labodev.local/adfs/services/trust

Parameter name	**SAML2_LOGIN_REQUEST_URL**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the request URL for the login in Arcsys for SAML2.
Shipped value	No shipped value.

Parameter name	**SAML2_LOGOUT_REQUEST_URL**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the request URL for the logout request from Arcsys for SAML2.
Shipped value	No shipped value.

Parameter name	**SAML2_LOGOUT_RESPONSE_URL**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the response URL to the logout request from the identification provider for SAML2.
Shipped value	No shipped value.

Parameter name	**SAML2_USER_ID_KEY**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the user id key to map the SAML2 request id in the application.
Shipped value	No shipped value.

Parameter name	**SAML2_CERT_SUBJECT_NAME**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the subject name of the certificate used for SAML2. This parameter is mandatory if SAML2_TRUSTSTORE_FILE is not empty.
Shipped value	No shipped value.

Parameter name	**SAML2_KEYSTORE_FILE**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Defines the keystore file to generate the signature in the SAML2 request.
Shipped value	No shipped value.

Parameter name	**SAML2_KEYSTORE_PASSWORD**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Parameter change assessment	After module reboot.
Description	Defines the keystore password to generate the signature in the SAML2 request.
Shipped value	No shipped value.

Parameter name	**SAML2_KEYSTORE_TYPE**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the keystore type to generate the signature in the SAML2 request.
Shipped value	No shipped value.

Parameter name	**SAML2_KEYSTORE_KEY_ALIAS**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the alias to use in the keystore to generate the signature in the SAML2 request.
Shipped value	No shipped value.

Parameter name	**SAML2_KEYSTORE_KEY_PASSWORD**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Defines the keystore key password to generate the signature in the SAML2 request.
Shipped value	No shipped value.

Parameter name	**SAML2_TRUSTSTORE_FILE**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Defines the truststore file to check the signature in the SAML2 response.
Shipped value	No shipped value.

Parameter name	**SAML2_TRUSTSTORE_PASSWORD**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Defines truststore password to check the signature in the SAML2 response.
Shipped value	No shipped value.

Parameter name	**SAML2_TRUSTSTORE_TYPE**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Defines the truststore type to check the signature in the SAML2 response.
Shipped value	No shipped value.

Parameter name	**SAML2_SIGNATURE_ALGORITHM**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The signature algorithm to sign SAML2 requests.
Possible values	MD5withRSA, SHA1withDSA, SHA256withDSA, SHA1withRSA, SHA224withRSA, SHA256withRSA , SHA384withRSA, SHA512withRSA, HmacSHA1, HmacSHA224, HmacSHA256, HmacSHA384, HmacSHA512
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SAML2_DEFLATER_ENABLED**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Indicates if the SAML2 request must be compressed.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SAML2_INFLATER_ENABLED**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Indicates if the SAML2 response must be decompressed.
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**SAML2_NAME_ID_FORMAT**
Parameter taken into account only if:	*AUTHENTICATION_TYPE*=SAML2
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The name id format. Controls the format of the subject of the SAML2 assertion. Select a value supported by your IDP.
Possible values	• urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified • urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress • urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName • urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName • urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos • urn:oasis:names:tc:SAML:2.0:nameid-format:entity • urn:oasis:names:tc:SAML:2.0:nameid-format:persistent • urn:oasis:names:tc:SAML:2.0:nameid-format:transient • urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted
Shipped value	No shipped value.
Parameter available since version (if applicable)	2024.1.LTS

9.6.8. Communication Protocol Parameters with the Arcsys Transfer Server

For parameters setting the communication protocol with the Arcsys Transfer Server (preset by *TRANSFER_SERVER_*), see Encryption Method with the Arcsys Transfer Server.

9.6.9. Parameters for display customization

Parameter name	**WEB_NOM_CLIENT**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	First part of the "title" of the web application, displayed in the title bar of the web browser.
Shipped value	Arcsys

Parameter name	**WEB_APPLI_NOM**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Second part of the "title" of the web application, displayed in the title bar of the web browser. This name is also displayed in the header of the website.
Shipped value	`Arcsys`

Parameter name	**WEB_APPLI_NOM_LONG**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Details of the application name. This name is also displayed in the header of the website.
Shipped value	`Archiving and Retrieval`

Parameter name	**FOLDERS_RANKING_SEPARATOR**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Separator between the range and the group name for the folder name associated to the customization on group level of `WEBAGENT_guipreferences.xml`. For more information on rankings, see `etc/WEBAGENT_guipreferences_example.xml` file.
Shipped value	_

Parameter name	**APPLICATION_HEADER_LOGO**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	This parameter allows to modify the header image logo. The image with the given absolute path will be loaded to replace the default Arcsys logo on the application header. The recommended size of the picture is an height of 36px and a width of 110px. A wider picture will be automatically cropped. The supported image formats are: JPEG, GIF, PNG and ICO.
Shipped value	No shipped value.

Parameter name	**KEEP_DEPRECATED_LOT_MODIFICATION_SCREEN**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	By default, the screen of lot modification in the administration interface of the Arcsys Web Agent is only available if the lot is not archived. *KEEP_DEPRECATED_LOT_MODIFICATION_SCREEN* parameter allows you to display this screen even when the lot is archived. Set this parameter to `true` when you want the modification of the lot to be always enabled in the administration interface of the Arcsys Web Agent. **Caution** When set to `true`, a user that has the permission to access the administration interface of the Arcsys Web Agent can change the start retention date of a lot.
Shipped value	`false`

Parameter name	**MAX_SELECT_ENREG_xxxx**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Maximum number of items selected in the relational database for the management page of the element xxxx (repository, collection, etc.). For example, if `500` is selected as *MAX_SELECT_ENREG_MASQUE*, only 500 masks will be loaded in memory from the relational database. The other masks will be ignored.
Shipped value	`500`

Parameter name	**MAX_AFFICH_ENREG_xxxx**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Maximum number of items displayed per page on the web for the management page of element xxxx (repository, lot, collection, etc.). For example, if `10` is selected as *MAX_AFFICH_ENREG_MASQUE*, only 10 masks will be displayed per page.
Shipped value	`10`

Parameter name	**MAX_ENREG_xxxx_BY_PAGE**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Maximum number of rows sent per display page of the element xxx. When this parameter is indicated, it replaces the two *MAX_SELECT* and *MAX_AFFICH* parameters.
Shipped value	`10` or `500` depending on the page

Due to the vast number of items in these lists, all elements in the relational database are not loaded in memory, rather only the elements to display per page. A request is made to the relational database for each new page.

9.6.10. Parameters for Sending Mails through the End User Interface

- *MAIL_ADMIN*: Defines the mail address of the administrator who sent the message;
- *MAIL_SENDER*: Indicates the name of the sender used;
- *MAIL_SMTP_SERVER*: Defines the host of the SMTP server;
- *MAIL_SMTP_SERVER_PORT*: Defines the port of the SMTP server;
- *MAIL_USING_PASSWORD*: Indicates that authentication must be done on the SMTP server;
- *MAIL_LOGIN* ; Login used for identification on the SMTP server;
- *MAIL_PASSWORD*: Password used for identification on the SMTP server;
- *MAIL_TEMPLATES_DIR*: Defines the directory where the template file to be used for the message is located;
- *MAIL_ADMIN_TEMPLATE*: Defines the file name of the template to be used for the message;
- *MAIL_SSL_PROTOCOLS*: SSL protocols that will be enabled for SSL connections for mail sending. The list of usable protocols depends on the Java environment. However, here is a list of protocols examples: `TLSv1.3`, `TLSv1.2`, `TLSv1.1`, `TLSv1`, `SSLv3` , `SSLv2Hello`. For further information, please refer to the Java documentation. The value of the property is a comma-separated list of protocols;
- *MAIL_TRUSTSTORE_TYPE*: Truststore type to be used for mail sending. Possible values: `JKS`, `PKCS12`, `JCEKS`, `Windows-MY`;
- *MAIL_TRUSTSTORE_FILE*: Path of the truststore file for mail sending;
- *MAIL_TRUSTSTORE_PASSWORD*: Truststore password for mail sending. This password must be encrypted. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.

9.6.11. Various Parameters in the End User Interface

- *DEFAULT_PRIORITY_REQUEST*: Defines the priority level used in creating requests through the End User Interface;

- *RIGHTS_POLICY_AFFECT_OWN_GROUPS*: Defines whether the repercussion of the rights on the groups when creating lots is enabled;
- *RIGHTS_POLICY_REPERCUT_UPPER*: Defines whether the upper repercussion of the rights when creating records is enabled;
- *ASSOCIATION_LECTEURS_AGENTS*: Defines the "network/agents drives" associations. This field must have a value respecting the following format: drive1,idAgent1|drive2,idAgent2. Several drives can be set for the same agent. On the other hand, several agents cannot be assigned to the same drive. To enter the drive paths in Windows, double the '\';
- *AFFICHE_ENTREE_<End User Menu>*: Allows you to specify whether an End User menu option is visible or not. Default value: `true`.

9.6.12. Work directories

Parameter name	**ONLINE_FILE_REP**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Storage directory for files resulting from queries (synchronous retrieval), obtained via communication with the transfer server. You must ensure sufficient storage space according to the desired use. With Unix, the directory must be accessible in write mode for the user running the application server. Operations must clean this directory.
Shipped value	`${TARGET_FOLDER} (special variable that automatically takes the value of the parameter of the -DTARGET_FOLDER command line)/ArcsysBuffer/onlineweb`

Parameter name	**STOCK_USERS_REP**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Root directory for saving user files (each user has a sub-folder). With Unix, the directory must be accessible in write mode for the user running the application server. This directory also contains: • Saved searches; • Exports from Arcsys entities (e.g. mask export, repository); • Uploaded files. **Note** This directory is shared with Arcsys Application Agent. Both modules must have access to it. See Setting rights on directories shared between Arcsys modules for more information.
Shipped value	`${TARGET_FOLDER} (special variable that automatically takes the value of the parameter of the -DTARGET_FOLDER command line)/ArcsysBuffer/users`

Parameter name	**STOCK_GROUPS_REP**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Directory where group preference folders are backed up. With Unix, the directory must be accessible in write mode to the user running the application server.
Shipped value	`${TARGET_FOLDER} (special variable that automatically takes the value of the parameter of the -DTARGET_FOLDER command line)/ArcsysBuffer/groups`

Parameter name	**CATEGORIES_DIR**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Directory where the record categories are located. These files are saved searches (with or without results).
Shipped value	`${TARGET_FOLDER} (special variable that automatically takes the value of the parameter of the -DTARGET_FOLDER command line)/ArcsysBuffer/categories`

Parameter name	**DIR_SCHEMAS**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Defines the directory where the schemas validating the XML file format are located (searches, result, export, etc.).
Shipped value	`${TARGET_FOLDER} (special variable that automatically takes the value of the parameter of the -DTARGET_FOLDER command line)/xml`

9.6.13. Parameters for Uploading Files

- *USE_FILE_UPLOADER*: Defines where a file upload widget is used.

Important

No other value than `true` is supported for this parameter.

- *FILE_UPLOADER_MAX_SIZE*: Maximum file size (in bytes) for each file uploaded. Default value: `104857600 (100 Mib)`. Max value: `4294967296 (4 Gib)`;

- *FILE_UPLOADER_NAME_PATTERN*: Defines using a regular expression the names of the files authorized for upload. Used to authorize certain file extensions. Default value: none;

- *FILE_UPLOADER_HASH_ALG*: Defines the checksum calculation algorithm used to check the fixity of the data uploaded. The hash value calculated with this algorithm will be stored in the object corresponding to the document in the relational database. The default hash type chosen in the application agent will therefore not be used in this case. Default value: `SHA-512`. Allowed values: MD5, SHA-1, SHA-256 or SHA-512;

- *FILE_UPLOADER_DO_CHECKSUM*: Defines in file uploading whether the web server calculates the hash value. Default value: `false`.

9.6.14. Retrieval Parameters

- *REST_PREFIX_PATH*: Defines the folder name where each lot will be retrieved, only in the event of retrieving several lots simultaneously (multiple action). This parameter can contain the variables "%ID_LOT%", "%LABEL_LOT%" and "%CODE_LOT%". These variables will be replaced by the lot identifier, the lot label and the lot code during retrieval.

9.6.15. Parameters for Managing Native Envelopes (ArcAFP Option)

- *ENABLE_NATIVE_MOVER*: Determines whether native envelope management is enabled or not in the web interface (values: `true`,/`false`).

9.7. etc/WEBAGENT_guipreferences_example.xml File

9.7.1. Definition

This file is a template for setting the display preferences.

The preferences common to all users are defined in the `etc/WEBAGENT_guipreferences.xml` file (optional).

The preferences can be loaded for each group in the `STOCK_GROUP_REP` folder. A user can belong to several groups, the name of the folder containing the `WEBAGENT_guipreferences.xml` file of the group must be of the form ranking + `FOLDERS_RANKING_SEPARATOR` + group name, with the ranking being an integer. The file used will be that where the ranking is the smallest.

They can also be loaded for each user by a file of the same name in the `STOCK_USERS_REP`<UserName> directory.

9.7.2. Parameters

The configuration of each page are stored in each page tag. To change a value, you need to pinpoint the field in question and change its value in the inherited file at user level.

9.7.3. Example of overwriting

If you want the record creation data and the record label to be displayed by default in the search results of the End User Interface, the `WEBAGENT_guipreferences.xml` should contain the following:

```
<?xml version="1.0" encoding="UTF-8"?>
 <GUI_PREFERENCES>
 <PAGE ID="311_END_USER">
 <COLUMN NAME="ARCHIVE_LABEL" SHOW="true" />
 <COLUMN NAME="ARCHIVE_CREATION_DATE" SHOW="true" />
 </PAGE>
 </GUI_PREFERENCES>
```

9.7.4. Customizing display

Overriding `WEBAGENT_guipreferences.xml` enables you to customize the following pages:

- End user creation/modification of a record
- End user search

9.7.4.1. Global end user customization

If `BASE_COLLECTION` parameter is set to false, the base and collection fields will be hidden, on condition that there are only one base and one collection visible to the connected user. Otherwise, the base and collection fields will not be hidden.

```
<?xml version="1.0" encoding="UTF-8"?>
 <GUI_PREFERENCES>
 <PAGE ID="GLOBAL">
    <FIELD NAME="BASE_COLLECTION" SHOW="false"/>
 </PAGE>
 </GUI_PREFERENCES>
```

9.7.4.2. Customizing creation/modification of a record

The following zones may be hidden by setting to `false` the appropriate variables in page `315`:

- Class of the record (`CLASS_CODE` field)
- Archive "Retention" tab (`ARCHIVE_RETENTION` field)
- Archive "Rights" tab (`ARCHIVE_RIGHTS` field)
- Display of the "Prepare archive" button (`PREPARE_ARCHIVE_BUTTON` field)

9.7.4.3. Customizing search page criteria

The following zones may be hidden by setting to `false` the appropriate variables in page `310`:

- Entity type (`ENTITY_TYPE` field)
- Archive number (`ARCHIVE_NUMBER` field)
- Name (`ARCHIVE_NAME` field)
- Author (`ARCHIVE_AUTHOR` field)

- Retention (`RETENTION` field)
- Class (`CLASS` field)
- File path (`FILE_PATH` field)

9.8. etc/WEBAGENT_label_xx_XX.properties File

9.8.1. Definition

These files contain all the labels displayed on the web.

9.8.2. Syntax

- xx: Defines the language;
- XX: Defines the country.

9.8.3. Languages Supplied

Arcsys comes with two files by default:

- WEBAGENT_label_fr_FR.properties: French (France);
- WEBAGENT_label_en_GB.properties: English (UK).

9.8.4. Parameters

All values in these files are used to display fields on the web. All texts can be changed but the application needs to be restarted for the changes to take effect.

9.9. etc/TEMPLATE_MAIL_ContactManager.properties File

9.9.1. Definition

This file is the mail template sent by the End User Interface in an administrator contact request.

9.9.2. Parameters

General parameters for mail templates:

The parameters are detailed in the section Customizing the Mail Template Option.

Specific parameters:

MAIL_MSG_OBJECT: Object of the message sent;

MAIL_MSG_CONTENT: Header of the content for the message sent;

MAIL_MSG_LOGIN: Login of the user sending the message;

MAIL_MSG_MAIL: Mail of the user sending the message;

MAIL_MSG_DATE: Title of the section giving the data and time of sending.

9.10. etc/TEMPLATE_MAIL_CreateAccount.properties File

9.10.1. Definition

This file is the mail template sent by the End User Interface in a request to the administrator to open an account.

9.10.2. Parameters

The parameters are detailed in the section Customizing the Mail Template Option.

9.11. etc/WEBAGENT_AccountCreation_xx_XX .properties File

9.11.1. Definition

This file configures the selection screen for requesting the administrator to open an Arcsys account.

9.11.2. Parameters

ACCOUNT_CREATION_TEXT_1: Text zone in the form;

ACCOUNT_CREATION_NAME: Title - Name (mandatory field);

ACCOUNT_CREATION_ADDRESS: Title - Address;

ACCOUNT_CREATION_LOGIN: Title - Identifier (mandatory field);

ACCOUNT_CREATION_CGU: Title - Usage conditions;

ACCOUNT_CREATION_CGU_CONTENT: Contents of the usage conditions.

Once all the mandatory fields have been completed, a summary mail will be sent to the administrator.

9.11.3. Account Opening Request Screen

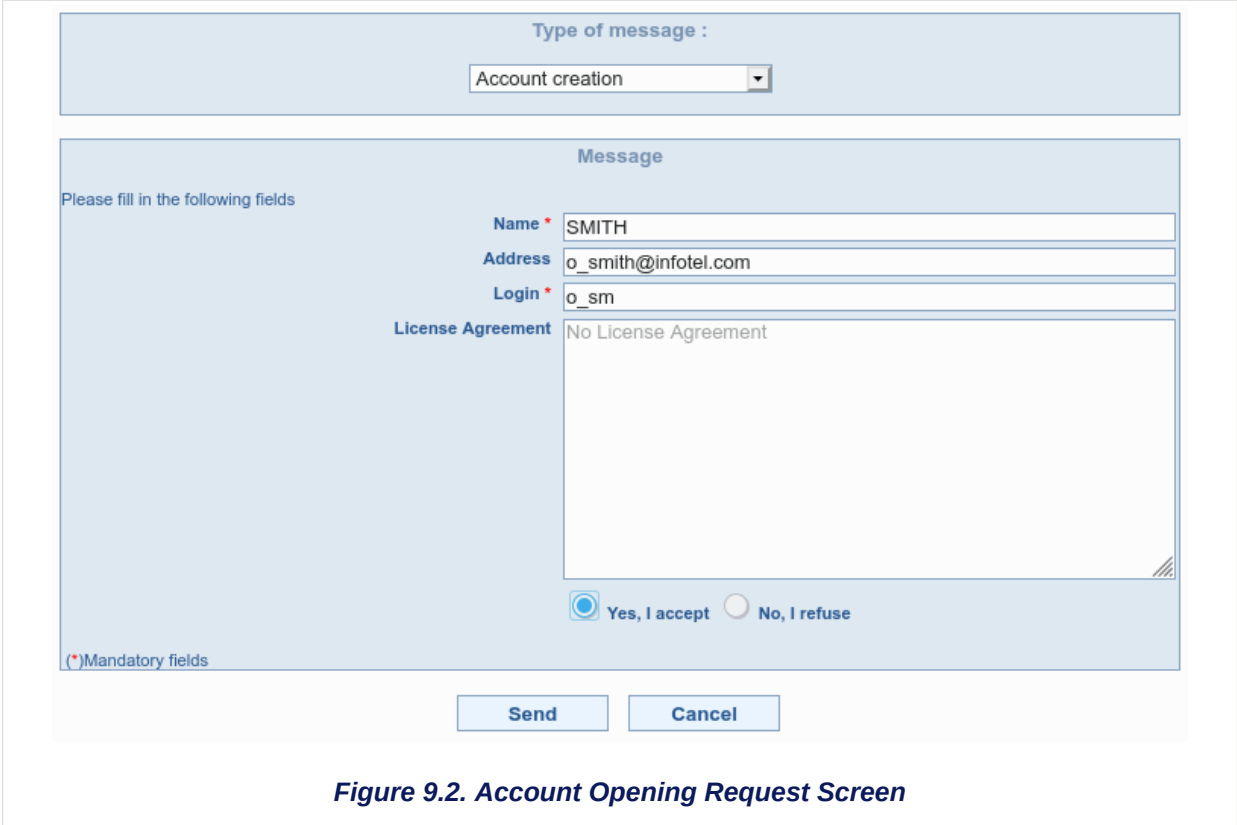


Figure 9.2. Account Opening Request Screen

9.11.4. etc/WEBAGENT_JobActionCodes_xx_XX.properties File

This file contains the code – Arcsys error message equivalences. Do not edit.

9.11.5. etc/WEBAGENT_AnomalyCodes_xx_XX.properties File

This file contains the error – Arcsys error message equivalences. Do not edit.

9.11.6. etc/arcsysJAAS.conf File

This file contains the parameters of the Java authorization and authentication service. See the description in JAAS configuration file.

10. Arcsys Auto-Archive Agent

10.1. Overview

The Arcsys Auto-Archive Agent is a module that enables automatic archiving of files. Although it operates with all file types, its behavior is more suitable for Arcsys log files.

It is installed in a directory called `ArcsysAutoArchiveAgent`. This module must be installed on each machine containing one or more Arcsys components. This way, Arcsys can auto-archive its own logs.

This module also can archive functional traces by collections.

10.2. List of Return Codes Sent

The Arcsys Auto-Archive Agent sends a return code following the transmission of the processes to be performed. See the table for the meaning of these codes.

In case of an error (return code other than 0), always consult the log files for more information.

Return code	Meaning
0	OK.
1	A parameter is incorrect.
2	The API is not available.
3	The Arcsys Application Agent specified cannot be found.
4	An error occurred when making the call to an API method.
10	Technical error.

10.3. Directory Structure

Type	Name	Description
	`launch_autoArchive.sh (or .bat)`	Log archiving module launch script.
	`launch_autoArchiveTrace.sh (or .bat)`	Trace archiving module launch script.
	`etc`	Directory containing the module configuration files.
	`AutoArchive.properties`	Main settings file of the module.
	`log4j2.xml`	log4j configuration file.

	`jar`	Directory containing the module Arcsys Java libraries (JAR).
	`libext`	Directory containing the different external Java libraries required in the module.
	`log`	Directory containing a list of all the log files. By default, these files are automatically rotated each day.

10.4. etc/log4j2.xml file

10.4.1. Definition

This file contains the settings for the log file. The logs are generated with the log4J Apache library.

10.4.2. Parameters

The parameters of this file depend on the information desired in the logs. Hence please consult the standard documentation of Log4J for any changes.

See: https://logging.apache.org/log4j/2.x/

10.5. etc/AutoArchive.properties File

10.5.1. Parameters for accessing the RMI API

Parameter name	**RMI_API_HOST_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Arcsys RMI API host name. If the client is on the same host as the RMI API, the `localhost` value is authorized.
Shipped value	`localhost`

Parameter name	**RMI_API_SERVICE_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Name of the Arcsys RMI API service. (it should be the same as COMPONENT_API_SERVICE_NAME in the RMI API properties file).
Shipped value	`ArcsysRmiAPI`

Parameter name	**RMI_API_PORT_NUMBER**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Arcsys RMI API port number.
Shipped value	`25040`

Parameter name	**RMI_API_USERNAME**
Parameter type	LDAP user. The user must exist in the used LDAP directory.
Parameter change assessment	After module reboot.
Description	Name of the RMI API LDAP user.
Shipped value	No shipped value.

Parameter name	**RMI_API_USERPASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the RMI API LDAP user.
Shipped value	No shipped value.

Parameter name	**KEYSTORE_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**KEYSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRUSTSTORE_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRUSTSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password for the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

10.5.2. Parameters for accessing the Application Agent

Parameter name	**APP_AGENT_HOSTNAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Host name of the application agent as referenced in the agent host field in the relational database, as shown in the agent list section of the website.
Shipped value	`localhost`

Parameter name	**APP_AGENT_PORT**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Application agent RMI port.
Shipped value	`25030`

Parameter name	**APP_AGENT_SERVICE_NAME**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Publication name of the RMI access interface to the application agent.
Shipped value	`ArcsysAgentAPI`

10.5.3. Archiving Media Parameters

- *`REFERENTIAL_USED_COLLECTION`*: Identifier of the collection where the lots are to be stored. Default value: `LogDefault`.

- *REFERENTIAL_USED_BASE*: Identifier of the repository of the collection involved. Default value: `AdminBase`.
- *LOT_CODE_PREFIX*: Lot code prefix. Default value: `AutoLog`.
- *ARCHIVE_ONLY_CLOSED_FILES*: Indicates if you want to archive only "closed" files, for which a renaming attempt finishes with success. This prevents problems resulting from the deletion of origin files that are still open (for logs dating several days that are still active in a module). Default value: `false`;
- *X_ARCHIVE_FILES*: Path of the files to archive. It may contain wildcard characters.

Note

On Windows, relative paths are not supported; a full path must be specified.

- *X_ARCHIVE_LASTMODIFIED*: Minimum number of days between the last change date of the file and the current date on which it should be archived.
- *X_ARCHIVE_SERVICE_NAME*: Provides a keyword for the archived lot.
- *X_ARCHIVE_COMPONENT_TYPE*: Provides a keyword that describes the Arcsys component where the logs are to be archived. Possible values: List of the controlled values for the "ArcysComponentType" keyword. By default, those installed are `Engine`, `TransferServer`, `WebAgent`, `ApplicationAgent`, `TransferService`, `Referential`, `SAPConnector`, `MailConnector`, `AutoArchiveAgent`, `APIs`, `Web Services`.

To specify the values of the parameters of the lot to archive, add the "X" prefix to the parameter name where X is a number. The first four should be given the number 1 and thus successively for each new series of parameters associated with a lot.

By default, automatic archiving enables the archiving of local Arcsys Application Agent, Arcsys Transfer Server, Arcsys Engine, Arcsys Web Agent and Arcsys Auto-Archive Agent.

10.5.4. Default Configuration

```
1_ARCHIVE_FILES=log/*.log.*
1_ARCHIVE_LASTMODIFIED=1
1_ARCHIVE_SERVICE_NAME=AutoArchiveAgentService
1_ARCHIVE_COMPONENT_TYPE=AutoArchiveAgent

# Transfer Server logs
2_ARCHIVE_FILES=../ArcsysTransferServer/log/*.log.*
2_ARCHIVE_LASTMODIFIED=1
2_ARCHIVE_SERVICE_NAME=TransferServerService
2_ARCHIVE_COMPONENT_TYPE=TransferServer

# Engine logs
```

```
3_ARCHIVE_FILES=../ArcsysEngine/log/*.log.*
3_ARCHIVE_LASTMODIFIED=1
3_ARCHIVE_SERVICE_NAME=EngineService
3_ARCHIVE_COMPONENT_TYPE=Engine

# Application Agent logs
4_ARCHIVE_FILES=../ArcsysApplicationAgent/log/*.log.*
4_ARCHIVE_LASTMODIFIED=1
4_ARCHIVE_SERVICE_NAME=ApplicationAgentService
4_ARCHIVE_COMPONENT_TYPE=ApplicationAgent

# Web Agent logs
5_ARCHIVE_FILES=../ArcsysWebAgent/log/*.log.*
5_ARCHIVE_LASTMODIFIED=1
5_ARCHIVE_SERVICE_NAME=WebAgentService
5_ARCHIVE_COMPONENT_TYPE=WebAgent
```

Important

The *REFERENTIAL_USED_COLLECTION* parameter designates the collection used by the Arcsys auto archiving module.

- **The storage policy of this collection must be valid (see Arcsys Functional Description Manual).**
- **The mask of this collection contains a number of mandatory keywords. If you wish to use a collection other than the default collection (`LogDefault`), the associated mask must contain the following keywords:**
  - **StartDate: Date-type keyword**
  - **EndDate: Date-type keyword**
  - **ArchiveDate: Date-type keyword**
  - **HostName: String-type keyword**
  - **ServiceName: String-type keyword**

    For more details on the different keywords, see the Module Operation paragraph.

LOT_CODE_PREFIX_FOR_TRACE: Lot code prefix. Default value: `AutoTrace`.

X_TRACE_DIRECTORY: Path of the trace file directory. The trace files that will be archived follow the pattern: collection_<idCollection>.log.*

10.6. launch_autoArchive.sh (or .bat) Script

This script starts up the automatic log archiving module. The Arcsys Engine, application agent and Arcsys APIs must be launched for it to run.

10.7. Module Operation

10.7.1. Introduction

This script automatically archives files and more specifically, log files.

10.7.2. Starting the Module / CRON

The operating system’s mechanisms can be used to launch the archiving task regularly (CRON in Unix or Task Manager in Windows).

10.7.3. Keywords Associated with the Lot

Keywords are associated to each lot created automatically by the auto archiving module. They can be viewed using the Arcsys Web Interface.

10.7.4. End Date

Date of the last change in all files referenced in the lot.

10.7.5. StartDate

Creation date of the oldest file in all files referenced in the lot.

10.7.6. ArchiveDate

Creation date of the archiving request.

10.7.7. HostName

Name of the host on which the module is installed and the files extracted.

10.7.8. ServiceName

Name of the application from where the files are extracted (optional).

10.7.9. Arcsys ComponentType

Name of the Arcsys component from which the logs are extracted.

10.8. launch_autoArchiveTrace.sh (or .bat) Script

This script starts up the trace automatic archiving module. The Arcsys Engine, application agent and Arcsys APIs must be launched for it to run.

10.9. Module Operation

10.9.1. Introduction

This module automatically archives functional trace files by collection.

10.9.2. Starting the Module / CRON

The operating system's mechanisms can be used to launch the auto-archiving task regularly (CRON in Unix or Task Manager in Windows).

11. CopyRequestManager

In order to copy existing archives, this tool must be used when adding a new copy to the storage profile. This tool is used to copy the archived lots of a given collection to a given pool. This pool is determined by the association `repositoryCode`/`collectionCode`/`zoneCode`.

Important

Because no automated mechanism will recreate the copy requests, any copy request that ends with an `ERROR` status (regardless of the cause of the error) must be re-created manually with CopyRequestManager once the error has been resolved.

11.1. Directory Structure

Type	Name	Description
	`copy-requests-manager.sh (or .cmd)`	Module launch script.
	`etc`	Directory containing the module configuration files.
	`log4j2.xml`	log4j configuration file.
	`REFERENTIAL_JDBC.properties`	File containing the parameters for accessing the Arcsys Database in JDBC mode.
	`jar`	Directory containing the module-specific Java libraries (jar).
	`libext`	Directory containing the third-party Java libraries (jar) of the module.
	`log`	Directory containing a list of all the log files.

11.2. etc/log4j2.xml file

11.2.1. Definition

This file contains the settings for the log file. The logs are generated with the log4J Apache library.

11.2.2. Parameters

The parameters of this file depend on the information desired in the logs. Hence please consult the standard documentation of Log4J for any changes.

See: https://logging.apache.org/log4j/2.x/

11.3. etc/REFERENTIAL_JDBC.properties File

This file contains the parameters for accessing the Arcsys Database in JDBC mode. See the description in page 123, « Configuring JDBC Access to the Relational Database ».

11.4. copy-requests-manager.sh (or .cmd) Script

The script is used as follows:

```
./copy-requests-manager.sh (or cmd) <mode> [options] [--help]
```

Where the available modes are:

- `create-copy-requests`
- `compute-statistics`

The help option displays the possible parameters and their description for each mode.

For each mode of the application, the exit codes are the following:

Return code value	Description
0	The process was successful
1	The script usage was wrong
2	An internal process error occurred

The JVM option *`arccop.max.rows.limit`* can be specified to indicate the maximal number of rows per block of SQL select to be stored in memory and processed (10000 if not specified).

11.4.1. `create-copy-requests` mode

The `create-copy-requests` mode is used to create the copy requests for all the archived lots of the collection that are not already archived on the target pool.

Important

Before executing this mode, be sure that:

- **there is not another instance of the script running on the same collection,**
- **there is at least one engine running with the parameter *`COPY_REQUESTS_PROCESSING`* set to `true` to process the created copy requests.**

If the traces are enabled on the target collection, the traces of the copy requests creation will be generated in the `trace` folder of the application. If necessary, you can archive them with the Arcsys Auto-Archive Agent module (see page 422, « Arcsys Auto-Archive Agent »).

A `last-handled-lot` file is created and regularly updated in the root folder of the application. This file contains the identifier of the last lot processed by the script (for a block of lots). This allows a possible recovery of the copy process via the *`lotIdMin`* option.

The `create-copy-requests` mode usage is:

```
./copy-requests-manager.sh (or cmd) create-copy-requests
--repositoryCode <code of the repository> --collectionCode <code of the collection>
--zoneCode <code of the zone>
[--lotIdMin <minimal lot Identifier, to be able to resume work if it is interrupted> |
 --lotIds <list of lot identifiers separated by comma without space>]
[--maxRequests <maximum number of requests to create, 1000 if not specified>]
```

11.4.2. `compute-statistics` mode

The `compute-statistics` mode is used to have an overview of the progress of the copies on a given pool.

The mode gives the following information at the end of its execution:

- The number of archived lots on the collection and the total size of these lots,
- The number of archived lots that should be copied on the specified target pool, the percentage of lots of the collection that it represents, the total size of the lots that should be copied and the percentage of the size of the lots of the collection that it represents.

The `compute-statistics mode` usage is:

```
./copy-requests-manager.sh (or cmd) compute-statistics
--repositoryCode <code of the repository> --collectionCode <code of the collection>
--zoneCode <code of the zone>
```

12. ArcFF format control module

The ArcFF format control module is installed in a directory called `ArcsysFileFormatServer`. This module identifies and validates the format of files to be archived in Arcsys.

12.1. Directory Structure

Type	Name	Description
	`launch_fileformat_server.sh (or .cmd)`	Module launch script.
	`shutdown_fileformat_server.sh (or .cmd)`	Module stop script.
	`status_fileformat_server.sh (or .cmd)`	Module status verification script.
	`installService_fileFormatServer.cmd (non systematic)`	Module Windows service installation script.
	`removeService_fileFormatServer.cmd (non systematic)`	Module Windows service uninstallation script.
	`etc`	Directory containing the module configuration files.
	arcff.properties	General configuration file for the module.
	`arcff_tool_assignments.xml`	Configuration file for assigning module tools.
	`arcff_tool_registry.xml`	File listing the tools used by the module.
	`log4j2.xml`	log4j configuration file.
	`droid`	Directory containing the configuration files necessary for DROID tool operations.
	`jhove`	Directory containing the configuration files necessary for JHOVE tool operations.
	`internal-signature-validation`	Directory containing the configuration files necessary for internal signature validation tool operations.
	`jar`	Directory containing the module-specific Java libraries (jar).

	`libext`	Directory containing the third-party Java libraries (jar) of the module.
	`log`	Directory containing a list of all the log files. By default, the files are automatically rotated daily.
	`xsd`	Directory containing the different XML schemas of the configuration files used by the application.

12.2. General Settings

12.2.1. etc/arcff.properties File

12.2.1.1. Definition

This file contains the general settings for the module.

12.2.1.2. Parameters

Parameter name	**JETTY_SERVER_PORT**
Parameter taken into account only if:	*SECURITY_PROTOCOL*= NONE
Parameter type	Port number
Parameter change assessment	After module reboot.
Description	Port for the REST web services of ArcFF format control module.
Shipped value	8085

Parameter name	**STOP_JETTY_TOKEN**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Stop token of the embedded Jetty server. This token is represented by any character string but must be unique with respect to embedded Jetty tokens present on the machine.
Shipped value	9lFVfB9ES31=

Parameter name	**TOOL_ASSIGNMENTS_EXACT_MATCH**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.

Description	This property concerns the `arcff_tool_assignments.xml` file, which determines which tool must be used for a MIME type and a given task. If the value of this property is `true`, the ArcFF format control module will search for a given MIME, an assignment with exactly this MIME type specified. If no assignment is found, an error occurs. If the value of this property is `false`, the ArcFF format control module will search as a priority for a given MIME type, an assignment with exactly this MIME type specified. If no assignment is found, the ArcFF format control module will search in these assignments for one possible whose specified MIME type is a given root MIME. For example, imagine the following assignment in the file `arcff_tool_assignments.xml` :
	
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<ASSIGNMENTS>`
	`  <ASSIGNMENT>`
	`  <TYPE>MIME_TYPE</TYPE>`
	`  <MIME_TYPE>application/pdf</MIME_TYPE>`
	`  <PROCESSES>`
	`  <PROCESS>`
	`  <PROCESS_TYPE>VALIDATION</PROCESS_TYPE>`
	`  <TOOL_CODE>JHOVE</TOOL_CODE>`
	`  </PROCESS>`
	`  </PROCESSES>`
	`  </ASSIGNMENT>`
	`<ASSIGNMENTS>`
	
	For a validation task for a MIME-type file `application/pdf; version=1.7` :
	
	• Value *`TOOL_ASSIGNMENTS_EXACT_MATCH`* set to `true`: no tools will be found because there is no exact MIME-type match.
	
	• Value *`TOOL_ASSIGNMENTS_EXACT_MATCH`* set to `false` : tool `JHOVE` will be used because "application/pdf; version=1.7" is an "application/pdf" specialization.

Shipped value	`false`

Parameter name	**ALLOWED_MIME_TYPES_EXACT_MATCH**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	On the same model as the parameter *`TOOL_ASSIGNMENTS_EXACT_MATCH`*, determines if after identification of the format in a file, the MIME type identified must match exactly one of the MIME types sent to the requests via the parameter *`allowedMimeTypes`*, or if the MIME type identified can be a root or have one of these MIME types as a root.
Shipped value	`false`

Parameter name	**SECURITY_CLIENT_ID**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The identifier of the client application for OAuth2 authentication on the ArcFF format control module. The value of this parameter is open-ended.
Shipped value	`arcff`

Parameter name	**SECURITY_CLIENT_SECRET**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	The secret shared by identifier of the client application for OAuth2 authentication on the ArcFF format control module. All application agents connecting to this module must have the same shared secret.
Shipped value	No shipped value.

Parameter name	**SECURITY_PROTOCOL**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The protocol used for exchanges with the ArcFF format control module. Possible values are: • `NONE`: Interactions with the ArcFF format control module are conventional HTTP exchanges. • `SSL`: Exchanges with the ArcFF format control module must be secured via SSL.
Shipped value	NONE

Parameter name	**JETTY_SERVER_HTTPS_PORT**
Parameter taken into account only if:	*SECURITY_PROTOCOL*= SSL
Parameter type	Port number
Parameter change assessment	After module reboot.
Description	Port for the REST web services of the ArcFF format control module for use with SSL protocol.
Shipped value	8085

Parameter name	**SSL_KEYSTORE_TYPE**
Parameter taken into account only if:	*SECURITY_PROTOCOL*= SSL
Parameter type	Characters string. Possible values are: `jks` or `pkcs12`.
Parameter change assessment	After module reboot.
Description	Type of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	jks

Parameter name	**SSL_KEYSTORE_FILE**
Parameter taken into account only if:	*SECURITY_PROTOCOL*= SSL
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the keystore containing the certificate used for SSL/TLS exchanges. For this component, the keystore file is required if SSL is enabled. In this case, ensure that the server hostname matches exactly with the hostname specified in the keystore. If this is not possible, you can add the parameter -Darcsys.arcff.hostname=<hostname> in the status_fileformat_server and shutdown_fileformat_server scripts to "force" the hostname to match the one in the certificate.
Shipped value	No shipped value.

Parameter name	**SSL_KEYSTORE_PASSWORD**
Parameter taken into account only if:	*SECURITY_PROTOCOL*= SSL
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the encrypt encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by ENC(...).
Parameter change assessment	After module reboot.
Description	Password of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**SSL_TRUSTSTORE_TYPE**
Parameter taken into account only if:	*SECURITY_PROTOCOL*= SSL
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Type of truststore used.
Shipped value	jks

Parameter name	**SSL_TRUSTSTORE_FILE**
Parameter taken into account only if:	*SECURITY_PROTOCOL*= SSL
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the truststore used.
Shipped value	No shipped value.

Parameter name	**SSL_TRUSTSTORE_PASSWORD**
Parameter taken into account only if:	*SECURITY_PROTOCOL*= SSL
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the truststore used.
Shipped value	No shipped value.

12.3. Tool Settings

12.3.1. etc/arcff_tool_registry.xml File

12.3.1.1. Definition

This file contains the inventory of the various tools used by the format control module. This file must follow a special structure:

```
<?xml version="1.0"?>
<TOOLS>
    <TOOL>
        <CODE>JHOVE</CODE>
        <NAME>JHOVE</NAME>
        <VERSION>1.28.0</VERSION>
        <CLASS>com.infotel.arcsys.file.format.tools.jhove.Jhove</CLASS>
        <CONFIG_FILES>
            <CONFIG_FILE>
                <FILE_CODE>modules</FILE_CODE>
                <FILE_NAME>jhove/jhove_module_assignments.xml</FILE_NAME>
            </CONFIG_FILE>
            <CONFIG_FILE>
                <FILE_CODE>internal</FILE_CODE>
                <FILE_NAME>jhove/jhove_internal.properties</FILE_NAME>
            </CONFIG_FILE>
        </CONFIG_FILES>
    </TOOL>
    <TOOL>
        <CODE>DROID</CODE>
        <NAME>DROID</NAME>
        <VERSION>6.6.1</VERSION>
        <CLASS>com.infotel.arcsys.file.format.tools.droid.Droid</CLASS>
        <CONFIG_FILES>
            <CONFIG_FILE>
                <FILE_CODE>signature</FILE_CODE>
                <FILE_NAME>droid/DROID_SignatureFile_V112.xml</FILE_NAME>
            </CONFIG_FILE>
            <CONFIG_FILE>
                <FILE_CODE>container</FILE_CODE>
                <FILE_NAME>droid/container-signature-20230510.xml</FILE_NAME>
            </CONFIG_FILE>
        </CONFIG_FILES>
    </TOOL>
```

```
    <TOOL>
        <CODE>VERAPDF</CODE>
        <NAME>VERAPDF</NAME>
        <VERSION>1.24.1</VERSION>
        <CLASS>com.infotel.arcsys.file.format.tools.verapdf.VeraPDF</CLASS>
    </TOOL>
    <TOOL>
        <CODE>INTERNAL_SIGNATURE_VALIDATION</CODE>
        <NAME>DETERMINED_FROM_CONFIGURATION</NAME>
        <VERSION>DETERMINED_FROM_CONFIGURATION</VERSION>
        <CLASS>com.infotel.arcsys.file.format.tools.internalsignaturevalidation.
InternalSignatureValidationTool</CLASS>
        <CONFIG_FILES>
            <CONFIG_FILE>
                <FILE_CODE>signature-validation</FILE_CODE>
                <FILE_NAME>internal-signature-validation/signature-validation.properties</
FILE_NAME>
            </CONFIG_FILE>
        </CONFIG_FILES>
    </TOOL>
</TOOLS>
```

The following must be specified for each tool referenced:

- The code, which must be unique in the file and which can be reused in the `arcff_tool_assignments.xml` file.
- The name.
- The version.
- The corresponding Java class.
- The configuration files specific to the tool (with an internal code and a file path).

12.3.2. etc/arcff_tool_assignments.xml File

12.3.2.1. Definition

This file specifies which tool will be used for a given process (identification or validation) and MIME type. In addition to assignment by MIME type, it is possible to define assignments by default. This file must follow a special structure:

```
<?xml version="1.0"?>
<ASSIGNMENTS>
        <ASSIGNMENT>
            <TYPE>DEFAULT</TYPE>
            <PROCESSES>
                <PROCESS>
                    <PROCESS_TYPE>DETECTION</PROCESS_TYPE>
                    <TOOL_CODE>DROID</TOOL_CODE>
                </PROCESS>
                <PROCESS>
                    <PROCESS_TYPE>VALIDATION</PROCESS_TYPE>
                    <TOOL_CODE>JHOVE</TOOL_CODE>
```

```
                </PROCESS>
                <PROCESS>
                    <PROCESS_TYPE>SIGNATURE_VALIDATION</PROCESS_TYPE>
                    <TOOL_CODE>INTERNAL_SIGNATURE_VALIDATION</TOOL_CODE>
                </PROCESS>
            </PROCESSES>
        </ASSIGNMENT>
        <ASSIGNMENT>
            <TYPE>MIME_TYPE</TYPE>
            <MIME_TYPE>application/pdf</MIME_TYPE>
            <PROCESSES>
                <PROCESS>
                    <PROCESS_TYPE>DETECTION</PROCESS_TYPE>
                    <TOOL_CODE>DROID</TOOL_CODE>
                </PROCESS>
                <PROCESS>
                    <PROCESS_TYPE>VALIDATION</PROCESS_TYPE>
                    <TOOL_CODE>JHOVE</TOOL_CODE>
                </PROCESS>
            </PROCESSES>
        </ASSIGNMENT>
        <ASSIGNMENT>
            <TYPE>MIME_TYPE</TYPE>
            <MIME_TYPE>application/pdf; version=1a</MIME_TYPE>
            <PROCESSES>
                <PROCESS>
                    <PROCESS_TYPE>IDENTIFICATION</PROCESS_TYPE>
                    <TOOL_CODE>DROID</TOOL_CODE>
                </PROCESS>
                <PROCESS>
                    <PROCESS_TYPE>VALIDATION</PROCESS_TYPE>
                    <TOOL_CODE>VERAPDF</TOOL_CODE>
                </PROCESS>
            </PROCESSES>
        </ASSIGNMENT>
        <ASSIGNMENT>
            <TYPE>MIME_TYPE</TYPE>
            <MIME_TYPE>application/pdf; version=1b</MIME_TYPE>
            <PROCESSES>
                <PROCESS>
                    <PROCESS_TYPE>IDENTIFICATION</PROCESS_TYPE>
                    <TOOL_CODE>DROID</TOOL_CODE>
                </PROCESS>
                <PROCESS>
                    <PROCESS_TYPE>VALIDATION</PROCESS_TYPE>
                    <TOOL_CODE>VERAPDF</TOOL_CODE>
                </PROCESS>
            </PROCESSES>
        </ASSIGNMENT>
</ASSIGNMENTS>
```

The following must be specified for each assignment:

- The assignment type, which can have a `mime_type` value (assignment for a given MIME type) or `default` (assignment used if no MIME type assignment was found for a given file).
- The MIME type (for a `mime_type` assignment file).
- The tools used per process. The tools used are referenced via their code defined in the `etc/arcff_tool_registry.xml` file.

12.3.3. etc/jhove/jhove_internal.properties File

12.3.3.1. Definition

This file contains the general parameters for JHOVE operation.

12.3.3.2. Parameters

Parameter name	**MODULE_ASSIGNMENTS_EXACT_MATCH**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Concerns the `jhove_module_assignments.xml`file. On the same model as the *`TOOL_ASSIGNMENTS_EXACT_MATCH`*parameter, determines if the assignment of the JHOVE modules by MIME type must take place strictly or not.
Shipped value	`false`

Parameter name	**BUFFER_SIZE**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Parameter specific to JHOVE operation, used to specify the size of the buffer when analyzing files.
Shipped value	`131072`

12.3.4. etc/jhove/jhove_module_assignments.xml File

12.3.4.1. Definition

This file specifies which JHOVE module will be used for given MIME type. This file must follow a special structure:

```
<?xml version="1.0"?>
<ASSIGNMENTS>
    <ASSIGNMENT>
        <TYPE>MIME_TYPE</TYPE>
        <MIME_TYPE>application/pdf</MIME_TYPE>
        <MODULE_CLASS>edu.harvard.hul.ois.jhove.module.PdfModule</MODULE_CLASS>
    </ASSIGNMENT>
    <ASSIGNMENT>
        <TYPE>MIME_TYPE</TYPE>
        <MIME_TYPE>image/jpeg</MIME_TYPE>
        <MODULE_CLASS>edu.harvard.hul.ois.jhove.module.JpegModule</MODULE_CLASS>
    </ASSIGNMENT>
    <ASSIGNMENT>
```

```
        <TYPE>MIME_TYPE</TYPE>
        <MIME_TYPE>image/gif</MIME_TYPE>
        <MODULE_CLASS>edu.harvard.hul.ois.jhove.module.GifModule</MODULE_CLASS>
    </ASSIGNMENT>
    <ASSIGNMENT>
        <TYPE>MIME_TYPE</TYPE>
        <MIME_TYPE>application/xml</MIME_TYPE>
        <MODULE_CLASS>edu.harvard.hul.ois.jhove.module.XmlModule</MODULE_CLASS>
    </ASSIGNMENT>
<ASSIGNMENTS>
```

The following must be specified for each assignment:

- The assignment type, which can have a `mime_type` value (assignment for a given MIME type) or `default` (assignment used if no MIME type assignment was found for a given file).
- The `MIME type` (for a `mime_type`) assignment file.
- The JAVA class of the JHOVE module.

12.3.5. etc/droid/DROID_SignatureFile_X.xml File

12.3.5.1. Definition

File associated with the DROID tool configuration (see http://www.nationalarchives.gov.uk/aboutapps/pronom/droid-signature-files.htm).

12.3.6. etc/droid/container-signature-X.xml File

12.3.6.1. Definition

File associated with the DROID tool configuration (see http://www.nationalarchives.gov.uk/aboutapps/pronom/droid-signature-files.htm).

12.3.7. etc/internal-signature-validation/signature-validation.properties File

12.3.7.1. Definition

Main configuration file to configure signature validation via another REST web service.

12.3.7.2. Parameters

Parameter name	**sigval.ws.name**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Name of the framework used to validate signatures. This information is traced in Arcsys database during a signature validation.
Shipped value	`DSS`

Parameter name	**sigval.ws.localhost**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Flag to indicate if the web service used to validate signature is on the same server as the client (ArcFF). If this is the case, the files transmitted between these two modules will not be uploaded (the file path will be used by the web service).
Shipped value	`true`

Parameter name	**siglib.ws.url**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Exact URL of the REST API method to target to validate the signature of a file. This URL must have the pattern `<protocol>://<host>:<port>/signature/validation`.
Shipped value	`http://localhost:8086/api/validation/files`

Parameter name	**sigval.ws.security.protocol**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The protocol used for exchanges with the Signature validator module. When set to `NONE`, interactions with the Signature validator module are conventional HTTP exchanges. When set to `SSL`, exchanges with the Signature validator module are secured via SSL.
Possible values	`NONE`, `SSL`
Shipped value	`NONE`
Parameter available since version (if applicable)	2024.1.LTS

Parameter name	**sigval.ws.security.url**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	URL to get a OAuth2 token for the signature validation webservice. If the value is a relative or absolute path, it is resolved from the*`siglib.ws.url`* value (when absolute, only the host part of *`siglib.ws.url`* is kept).
Shipped value	`/api/oauth`

Parameter name	**sigval.ws.security.client.id**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Identifier of the client application for OAuth2 authentication on the Signature validator module. This value must be identical to the value of the *`sigws.security.client.id`* parameter in Signature validator module.
Shipped value	`signature_validator`

Parameter name	**sigval.ws.security.client.secret**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Secret shared by identifier of the client application for OAuth2 authentication on the Signature validator module. See the parameter *`sigws.security.client.secret`* in Signature validator module.
Shipped value	No shipped value.

Parameter name	**sigval.ssl.truststore.file**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	For the exchanges with Signature validator module : Location of the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**sigval.ssl.truststore.password**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	For the exchanges with Signature validator module: Password for the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**sigval.ssl.truststore.type**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	For the exchanges with Signature validator module: Type of the truststore used for SSL/TLS exchanges.
Shipped value	`jks`

Parameter name	**sigval.connection.timeout.ms**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	Define the maximum number of milliseconds for which the client (ArcFF) will wait for a connection to be established with the Signature validator module before a timeout.
Shipped value	`5000`

Parameter name	**sigval.socket.timeout.ms**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	Determine the socket timeout in milliseconds, which is the timeout for waiting for data or, said differently, a maximum period inactivity between two consecutive data packets.
Shipped value	`120000`

Parameter name	**sigval.connection.manager.timeout.ms**
Parameter type	Number of milliseconds.
Parameter change assessment	After module reboot.
Description	Determine the timeout in milliseconds used when requesting a connection from the connection manager.
Shipped value	`120000`

12.4. launch_fileformat_server.sh (or .cmd) Script

This script is used to start the format control module in the form of REST services deployed on an embedded Jetty server.

12.5. shutdown_fileformat_server.sh (or .cmd) Script

This script stops the format control module. For this script to be effective, the configuration of the module concerning the protocol used, the SSL configuration, the port used and the Jetty server stop token must not have changed between the module start and stop.

The host name must match the certificate configuration in case SSL is used. Alternatively, you can customize the host name by setting an additional parameter `-Darcsys.arcff.hostname=...` in the command line of the script.

12.6. status_fileformat_server.sh (or .cmd) Script

This script is used to recover a return code corresponding to the current status of the module. For this script to be effective, the configuration of the module concerning the protocol used, the SSL configuration, the port used and the Jetty server stop token must not have changed between the module start and stop.

The host name must match the certificate configuration in case SSL is used. Alternatively, you can customize the host name by setting an additional parameter `-Darcsys.arcff.hostname=...` in the command line of the script.

Return code value	Agent status	Description
84	STARTED_AND_WORKING	The module is started.
86	SHUTDOWN	The module is stopped.
89	GET_STATUS_ERROR	Error in retrieving the status.

12.7. installService_fileFormatServer.cmd Script

This script deploys the ArcFF format control module as a Windows service.

12.8. removeService_fileFormatServer.cmd Script

This script uninstalls the application service as a Windows service.

13. ArcsysFsComparator File systems comparator

The ArcsysFsComparator File systems comparator is installed in a directory called `ArcsysFsComparator`. This module analyzes a file system in order to detect any files that are missing, that are extra or that are the incorrect size.

13.1. Directory Structure

Type	Name	Description
	`analyze.sh (or .bat)`	Launch script for complete analysis of the differences between one or more physical file systems and the corresponding logical file system.
	`analyzeFS.sh (or .bat)`	Script to run an inventory of a physical file system, i.e. the set of envelopes present physically on the file system.
	`analyzeRef.sh (or .bat)`	Script to run an inventory of a logical file system, i.e. the set of envelopes present on the file system according to the data of the relational database.
	`analyzeZone.sh (or .bat)`	Launch script to analyze the physical and logical file systems of a zone and their inconsistencies.
	`compare.sh (or .bat)`	Launch script for comparison between the two file system inventories.
	`etc`	Directory containing the module configuration files.
	`log4j2.xml`	log4j properties file.
	`ARCFSC.properties`	Configuration file of the module, containing the parameters to communicate with the Arcsys API via RMI, among others.
	`lib`	Directory containing the module's Java libraries (jar).
	`log`	Directory containing a list of all the log files. By default, these

		files are automatically rotated each day.

13.2. etc/log4j2.xml file

13.2.1. Definition

This file contains the settings for the log file. The logs are generated with the log4J Apache library.

13.2.2. Parameters

The parameters of this file depend on the information desired in the logs. Hence please consult the standard documentation of Log4J for any changes.

See: https://logging.apache.org/log4j/2.x/

13.3. etc/ARCFSC.properties File

13.3.1. Definition

This file contains the parameters for the module.

13.3.2. General Parameters

General parameters:

- *CHECKSUM_ALGORITHMS*: Different algorithms used for calculating the hash values of the envelopes during inventory of a physical file system. Possible values: `MD5, SHA-1, SHA-256, SHA-512.`

  If several algorithms are used within a file system, you must specify the list of algorithms as a value, separated by the character ",".

  Although it is not recommended, it is possible to leave the *CHECKSUM_ALGORITHMS* parameter blank. This can be decided when the checksum comparison is not necessary and in case of large files. In this case, leaving this parameter blank avoids calculating checksums and thus improves performance.

Caution

This parameter is mandatory for analyzeZone.sh (or .bat) Script .

- *AUTO_CHECK_REQUEST*: Indicates whether the connector is authorized to launch Arcsys check requests on lots where envelopes are corrupted or missing. Possible values: `true` / `false`.
- *TEMP_FOLDER*: Path to the connector's temporary folder. This folder is used when using certain scripts such the `analyze` or `analyzeZone` script to generate temporary inventories that will be used in generating the end file. The folder is automatically empty after the script has been run.
- *LAST_REQUEST_MIN_TIME*: Minimum period (in seconds) that must separate the moment when a comparison or complete analysis script is run from the moment when one of these actions (amongst: archiving request or migration request or physical modification of one of the envelopes) has been executed last on the lot. If this period is not reached, the lot will be ignored.
- *LOT_MAX_ENVELOPES*: Specifies the maximum number of envelopes that can be present for a lot to optimize the physical reading of these envelopes. This parameter is taken into account only if the program option `lmax` is specified at the execution of the module.

13.3.3. Parameters for Accessing the API

- *API_MODE*: Type of access to the API. Currently the only mode available is RMI.
- *API_DELAY*: Maximum idle time for a connection (in milliseconds). After this delay, the connection is terminated.
- *API_RE_USE_CALLER*: Indicates whether a single connection manager should be used for all connections for a user or whether a manager should be created for each connection. Possible values: Possible values: `true` / `false`.
- *API_SECURITY_PROTOCOL*: Indicates the security protocol to be used for incoming and outgoing RMI connections (to and from the engine). Possible values: `SSL` / `NONE` (default).
- *API_TRUSTSTORE_FILE*: Location of the truststore file.
- *API_KEYSTORE_FILE*: Location of the keystore file.
- *API_TRUSTSTORE_PASSWORD*: Encrypted password of the truststore file (passphrase).
- *API_KEYSTORE_PASSWORD*: Encrypted keystore file password (passphrase).

13.3.4. Specific Parameters for RMI Mode

- *API_SERVER*: Arcsys RMI API host name.
- *API_PORT*: Arcsys RMI API port number.

- *API_USERNAME*: RMI API user name.
- *API_PASSWORD*: Encrypted RMI API user password.
- *API_TARGET*: Name of the Arcsys RMI API service. (it should be the same as *COMPONENT_API_SERVICE_NAME* in the RMI API configuration file).

13.4. analyzeFS.sh (or .bat) Script

13.4.1. Description

This script is used to perform a physical inventory of a file system. All files are browsed and accounted for.

13.4.2. Arguments

To launch an inventory of a physical file system, you need to run the following command:

```
analyseFS.bat(.sh) <path> [-o <fileName>] [-lmin <lotIdMin>] [-lmax <lotIdMax>]
```

The arguments recognized by this script are:

- `path` (mandatory): Path to the folder to analyze.
- `fileName`: Analysis result file (`analysis.csv` by default);
- `lmin`: Minimum identifier of the lot to process
- `lmax`: Maximum identifier of the lot to process.

13.4.3. File Output

This creates a file containing the list of files found in the directory indicated by <path>.

The format of the file content is a line for each file found with the information:

```
Lot, Mover, file, size, last update, checksum (name of the algorithm)
```

Where lot is the lot number, Mover is 1 for manifest and 2 for record, file is the file name, size is the size of the file in bytes, last update is a timestamp corresponding to the last file modification, checksum is the file hash.

At the end of this file, there is a line with the format:

`Lots, files, sizes`, where `Lots` is the total number of lots found in the file system, `Files` is the total number of files found and `size` is the overall size (in bytes).

13.5. analyseRef.sh (or .bat) Script

13.5.1. Description

This script performs a logical inventory of a file system. All files that are counted and are present are accounted for. The Arcsys RMI API has to be launched for it to work.

13.5.2. Arguments

To launch a logical inventory, you need to run the following command:

```
analyzeRef.bat(.sh) <path> [-s <site>] [-o <fileName>] [-lmin <lotIdMin>] [-lmax
 <lotIdMax>]
```

The arguments recognized by this script are:

- `path` (mandatory): Full path of the file system to be analyzed (Note: the <`path`> specification in Windows must be done with the file system path as it appears in the Arcsys graphical interface, i.e. with the double \).
- `site`: Site name ('PRIMARY_SITE' by default).
- `fileName`: Analysis result file (`analyse.csv` by default).
- `lmin`: Minimum identifier of the lot to process.
- `lmax`: Maximum identifier of the lot to process.

13.5.3. File Output

The contents of a file produced has a description similar to that in the file produced by analyseFS.

13.6. compare.sh (or .bat) Script

13.6.1. Description

This script launches the comparison between the two file system analyses. These analyses can be logical or physical and are not necessarily the same type.

13.6.2. Arguments

To launch a comparison, you need to run the following command:

```
compare.bat(.sh) <file1> <file2> [-name <rootName>]
```

The arguments recognized by this script are:

- `file1,file2`: (mandatory): Path to the files to be compared
- `rootName` Root used for the name of the files generated ('compare' by default)

13.6.3. Result

The comparison gives 3 files.

- `<rootName>-diff.csv`
- `<rootName>--missing1.csv`
- `<rootName>--missing2.csv`

The first gives the files found in the two inventories where the checksum differs.

The format of the contents of this file is as follows:

```
Lot, Mover, file, size1, size2, checksum1, checksum2
```

With `Lot`, `Mover`, `file` that correspond to the definition given above, `size1` to the size found in the first inventory, `size2` to the size found in the first inventory, checksum1 to the checksum found in the first inventory and checksum2 to the checksum found in the first inventory.

The two other files contain the list of files present in one of the inventories. Where `file1` is the name of the file produced by the analyseFs script, and `file2` is the name of the file produced by the analyseRef script:

- `<rootName>--missing1.csv` contains the list of files found on the file system and not found in the Arcsys Database. This file can be used, for example, to delete the files that are too numerous on the disk array. Important: Do not delete the file until you have checked that the lots actually contain an error. Initially you should **move** these files to another directory in the disk array.
- `<rootName>--missing2.csv` contains the list of files found in the Arcsys Database and not found on the disk array. This file should be empty. If it is not, you should check that the commands listed previously were correctly passed and contain the correct parameters.

13.7. analyze.sh (or .bat) Script

13.7.1. Description

This script launches an analysis of differences between one or more physical file systems and the logical file system associated with each one. The script can also

generate "check" requests in Arcsys when necessary and when they are activated in the ArcFSC settings.

13.7.2. Arguments

To launch an analysis, you need to run the following command:

```
analyze.bat(.sh) <path1> <path2> ... <pathN> [-name <rootName>] [-s <site>] [-lmin
 <lotIdMin>] [-lmax <lotIdMax>]
```

The arguments recognized by this script are:

- `path1`, `path2`, ...: Set of full file paths for the file systems to be analyzed. *At least one path must be specified.*
- `rootName` Root used for the name of the files generated ('`analysis`' by default).
- `site`: Site name ('`PRIMARY_SITE`' by default).
- `lmin`: Minimum identifier of the lot to process
- `lmax`: Maximum identifier of the lot to process.

13.7.3. Result

The analysis gives a file per file system processed: `<rootName>_fsId.csv`, `<rootName>_fsId.csv`, ...

For each analysis, the file generated gives the incoherences at envelope level between a physical file system and its associated logical file system (absence of envelopes, different envelopes) and suggests an action to be performed. Four different actions may be offered:

- **Envelope to delete**: This action is suggested when an envelope is physically present on the file system whereas it shouldn't be there (e.g. an envelope that is in disposal).
- **Envelope to repair**: This action is suggested when an envelope is physically absent from the file system whereas it should be there, or where the envelope is present but corrupted. According to ArcFSC parameter, this action may launch automatic check requests in such cases.
- **Wait for recovery request**: This action is suggested when an envelope is absent from the relational database whereas it is physically present in the file system because the lot concerned is pending recovery.
- **Unexpected case. Please contact an administrator**: This case occurs when a corrupted or absent envelope cannot be resumed by passing the check requests because the lot is not present on another zone.

The contents of each file is formatted as follows:

```
Lot, File, Code, Meaning, Complementary information
```

With `Lot`, `file` which matches the highest definition given, `Code` matching the code of the action to be performed (1 to 4), `Meaning` giving the definition of the action to be performed and `Complementary information` providing additional comments on the envelope and justifying the proposed action.

13.8. analyzeZone.sh (or .bat) Script

13.8.1. Description

This script launches an analysis of inconsistencies in all the file systems of a specified zone. The script can also generate "check" requests in Arcsys when necessary and when they are activated in the ArcFSC settings, as well as corrections (SQL scripts for the relational database, shell scripts for the file system).

13.8.2. Arguments

To launch an analysis, you need to run the following command:

```
analyzeZone.bat(.sh) <zoneId> [-o <outputDir>] [-lmin <lotIdMin>] [-lmax <lotIdMax>] [-csv
 <inventoriesFolder>]
```

The arguments recognized by this script are:

- `zoneId`: Identifier of the zone concerned
- `outputDir`: Path to the directory in which the script (analysis, scripts) results will appear
- `lmin`: Minimum identifier of the lot to process
- `lmax`: Maximum identifier of the lot to process
- `csv`: Path to the directory containing all or part of the file system inventories in the zone, as these files are used by the script.

  For the files to be taken into account, they must have as a name `physical_<fsId>.csv` for physical inventories (obtained by using the analyzeFS script) and `referential_<fsId>.csv` for logical inventories (obtained by using the analyzeRef script).

  If any file systems inventories for the zone are not present in the specified directory, the script will generate the missing inventories in this directory.

If the path is not specified, the script will generate the inventories in the temporary ArcFSC folder (specified in the settings).

It should be noted that to generate the physical inventory of a file system, ArcFSC must have access to the latter, whereas for the logical inventory of a file system, ArcFSC must have access to an Arcsys API connected to the corresponding relational database.

13.8.3. Result

The script produces an analysis file with the name `analyze_zone-<zoneId>.csv`.

In this file, each row corresponds to an envelope with a problem. For each of these envelopes, the following information is present:

- `Lot`: Lot identifier;
- `File`: Name of the envelope (`lot_<lotId>_seq_<moverId>`);
- `Zone`: Zone identifier;
- `Should be on zone?`: Specifies if the lot should be present on the zone according to the database. Possible values are:
  - **YES**: According to the database, the lot is archived on the zone (archiving lot pool active on one of the filesystems of the zone);
  - **NO**: According to the database, the lot is not archived on the zone (there is no archiving lot pool active on the zone).
  - **PENDING**: According to the database, the lot is waiting recovery (there is an archiving lot pool pending on one of the filesystems of the zone).
- `Other zones for the lot`: Identifiers of other zones where the lot is archived. If there are several other zones where the lot is archived, the zone identifiers are separated by `;`. If the lot is not archived on any other zone, the value of this field `NONE`;
- `Physical characteristics`: Physical characteristics of the envelope on the different file systems of the zone in question. This field contains as many values as the number of times the envelope is found on one of the filesystems of the zone. Possible values for one found envelope on one filesystem are:
  - **OK**: According to the database, the found envelope should be on this filesystem and has a valid checksum;
  - **CHECKSUM_DIFF**: According to the database, the found envelope should be on this filesystem but has a different checksum;

- **ALL_DIFF**: According to the database, the found envelope should not be on this filesystem and has a different checksum.

If there are several values, these are separated by **;**. If the envelope is not found on any filesystem of the zone, the value of the field is **NONE**;

- `Referential characteristics`: Characteristics of the envelope according to the relational database. This field has a unique value with the pattern **FSIdentifier[<filesystemId>]ChecksumType[<hashFunction>]Checksum[<hashValue>]**. If the envelope is not present in the zone according to the relational database, the field will have the value **NONE**;

- `Action`: Suggested actions according to the values of the above-mentioned fields. Possible values for a suggested action are:

  - **Envelope to delete from filesystem [<filesystemId>] (shell command generated)**: The envelope can be safely deleted from the specified filesystem;

  - **Check request created with id [<requestId>]**: A check request was created to trigger a potential repair on the envelope;

  - **Check request with id [<requestId>] already in progress for this lot**: A check request is already processing on this lot to trigger a potential repair on the envelope;

  - **Check request creation disabled, you should repair this envelope**: A check request is needed on the lot of the envelope but the check request creation is disabled in the application.

  - **Change referential filesystem ident to [<filesystemId>] (SQL command generated)**: The filesystem ident of the envelope should be modified in database.

  If there are several values, these are separated by **;**. If there is no suggestion action, the value of the field is **NONE**.

  To complete the suggested action, a number of scripts can be generated:

  - `Scripts fix_fs_<fsId>.sh`: Each of these scripts concern a file system in particular and move the envelopes to be deleted in a `TRASH` folder used as a trash bin and created on the root of the file system. Each of these scripts must therefore have physical access to the file system that it concerns during its execution;

  - `Script fix_zone_<zoneId>.sql`: This SQL script is generated when corrections are to be made directly in the relational database, such as modifications to the identifier of the file system of an envelope.

14. Arcsys standard Clients

14.1. Overview

The Arcsys standard Clients is installed in a directory called `ArcsysStandardClients`. These three clients using the Arcsys RMI API have simple mechanisms to archive, retrieve and search data.

14.2. List of Return Codes Sent

The standard clients send a return code following transmission of the processes to be performed. See the table for the meaning of these codes.

In case of an error (return code other than 0), always consult the log files for more information.

Return code	Meaning
0	OK.
1	A parameter is incorrect.
2	The API is not available.
3	The Arcsys Application Agent specified cannot be found (archiving and retrieval client).
4	An error occurred when making the call to an API method.
10	Technical error.
11	The archiving or retrieval request ended with `ABORTED` status.
13	The archiving or retrieval request ended with `ERROR` status.

14.3. Directory Structure

Type	Name	Description
	`archiveClient.sh (or .bat)`	Archiving standard client launch script.
	`restoreClient.sh (or .bat)`	Retrieval standard client launch script.
	`searchClient.sh (or .bat)`	Search standard client launch script.
	`etc`	Directory containing the module configuration files.
	`ArchiveClient.properties`	Settings file of archiving standard client.
	`log4j2.xml`	log4j configuration file.

	`RestoreClient.properties`	Settings file of retrieval standard client.
	`SearchClient.properties`	Settings file of search standard client.
	`jar`	Directory containing the module Arcsys Java libraries (JAR).
	`libext`	Directory containing the different external Java libraries required in the module.
	`log`	Directory containing a list of all the log files. By default, these files are automatically rotated each day.

14.4. etc/log4j2.xml file

14.4.1. Definition

This file contains the settings for the log file. The logs are generated with the log4J Apache library.

14.4.2. Parameters

The parameters of this file depend on the information desired in the logs. Hence please consult the standard documentation of Log4J for any changes.

See: https://logging.apache.org/log4j/2.x/

14.5. etc/ArchiveClient.properties File

14.5.1. Definition

This file contains the settings for the archiving client.

14.5.2. Parameters for accessing the RMI API

Parameter name	**RMI_API_HOST_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Arcsys RMI API host name. If the client is on the same host as the RMI API, the `localhost` value is authorized.
Shipped value	`localhost`

Parameter name	**RMI_API_SERVICE_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Name of the Arcsys RMI API service. (it should be the same as COMPONENT_API_SERVICE_NAME in the RMI API properties file).
Shipped value	`ArcsysRmiAPI`

Parameter name	**RMI_API_PORT_NUMBER**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Arcsys RMI API port number.
Shipped value	`25040`

Parameter name	**RMI_API_USERNAME**
Parameter type	LDAP user. The user must exist in the used LDAP directory.
Parameter change assessment	After module reboot.
Description	Name of the RMI API LDAP user.
Shipped value	No shipped value.

Parameter name	**RMI_API_USERPASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the RMI API LDAP user.
Shipped value	No shipped value.

Parameter name	**KEYSTORE_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**KEYSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRUSTSTORE_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRUSTSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password for the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

14.5.3. Parameters for accessing the Application Agent

Parameter name	**APP_AGENT_HOSTNAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Host name of the application agent as referenced in the agent host field in the relational database, as shown in the agent list section of the website.
Shipped value	`localhost`

Parameter name	**APP_AGENT_PORT**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Application agent RMI port.
Shipped value	`25030`

Parameter name	**APP_AGENT_SERVICE_NAME**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Publication name of the RMI access interface to the application agent.
Shipped value	`ArcsysAgentAPI`

14.5.4. Archiving Parameters

- *`BATCH_RECREATE_ARCHIVING_REQUESTS`*: Indicates whether you want to recreate archiving requests for each of the lots mentioned in *`BATCH_REPAIR_LOT_LIST`* list (the agent used will be the agent set in *`AGT_HOST`* and *`AGT_PORT`*). Possible values: `true / false`. Default value: `false`.

- *`BATCH_RECREATE_ARCHIVING_REQUESTS_REUSE_APPLICATION_AGENT`*: If *`BATCH_RECREATE_ARCHIVING_REQUESTS`* is `true`, it indicates if the application agent of the last archiving request in error must be reused (true) or not (false). Possible values: `true / false`. Default value: `false`.

- *`BATCH_REPAIR_LOT_LIST`*: If *`BATCH_RECREATE_ARCHIVING_REQUESTS`* is `true`, it lists the identifiers of the lots for which the requests will be recreated. An identifier interval can be done as x-y, and the identifiers or intervals can be separate by commas. Example: 4151-4200,4205,4210-4300.

- *`REPEAT_OPERATION_NUMBER`*: This parameter optionally allows you to create n lots and n archiving requests for the same data. This can be particularly useful for benches. Default value: `1`

- *`REPEAT_OPERATION_DELAY_MS`*: If REPEAT_OPERATION_NUMBER >1, this parameter indicates the interval in milliseconds between each lot creation and archiving request operation.

- *`ARCHIVE_TYPE`*: Record type (directory, file, database, etc.) Possible values: DIRECTORY / FILE / BASE / DIRECTORY_LIST.

- *`LOT_CODE_PREFIX`*: This parameter indicates the code prefix of the lots that will be archived by the archiving standard client. Default value: `LOT_AUTO`. If this parameter does not have a value or is absent from the configuration file, the default value will be taken.

Important

The prefix length is limited to 14 characters. The prefix will be truncated if it exceeds this length.

Note

The code of the archived lots takes the following format:

14-character prefix followed by <_yyyyMMddHHmmssSSS>

Where yyyy presents the year, MM the month, dd the day, HH hours, mm minutes, ss seconds and SSS milliseconds of the archiving start date.

- *LOT_OWNER*: This optional parameter indicates the LDAP user that created the lot, if it is different to that of the API (*RMI_API_USERNAME*).
- *LOT_GROUP/OTHER_READ/WRITE_RIGHT*: These optional parameters indicate the rights configured on the lot for the group and other users. Default values: `true` for LDAP_GROUP_READ_RIGHT and LDAP_OTHER_READ_RIGHT, `false` for LDAP_GROUP_WRITE_RIGHT and LDAP_OTHER_WRITE_RIGHT.
- *ARCH_DIR*: Directory to archive. This parameter is taken into account only if the ARCHIVE_TYPE is `DIRECTORY`. You can still archive a set of files using the wildcards (POSIX standard, for example `/home/user1/*.log`).
- *ARCH_FILE*: File to archive. This parameter is taken into account only if the ARCHIVE_TYPE is `FILE`.
- *ARCH_COMMANDGROUPID*: Identifier of the database commands group controlling archiving. This parameter is taken into account only if the ARCHIVE_TYPE is `BASE`.
- *ARCH_SERVICEBDD_NAME*: Name of the DB service managing archiving. This parameter is taken into account only if the ARCHIVE_TYPE is `BASE`.
- *ARCH_DIR_COLLECTION_LIST*: List of directories to scan, accompanied by the codes of the repository and the collection to use. This parameter is taken into account only if the ARCHIVE_TYPE is `DIRECTORY_LIST`. The folder name and the repository code are separated by a comma, in the same way as the repository code and the collection code. Each triplet is separated by a semicolon ";" character.

  With ARCHIVE_TYPE=DIRECTORY_LIST, the standard client automatically archives the contents of one or more folders specified in ARCH_DIR_COLLECTION_LIST. Its contents is divided into two categories:

  – Indexing files (or metadata), where the extension is set in the METADATA_FILE_EXTENSION parameter of ArchiveClient.properties (default: .ind). These files are not archived but they are used to give a value to the metadata.

  – Data files. The data file matching an indexing file has the same name without the extension. Example: indexing file test.ind matches test.csv but also to test.

- Each indexing file found gives rise to a lot containing the data file matching that indexing file. The lot code is obtained by truncating the name of the original file and suffixing a number if the name is already taken.
- Each indexing file found is renamed during the archiving process. The extension indicates the progress status:
  - .cur: process in progress,
  - .err: process ending in error,
  - .old: process completed without errors.

  The files with one of these extensions are ignored during the scanning of the folder.

  If a file to archive without a metadata file is found, an error is traced in the log file ("Could not find metadata file for file"...).

  If a lot with identical metadata is found in the same collection, a WARN is indicated ("Found already archived lot with same metadata as file").

- COL_ID: Identifier of the collection where the lots are to be stored.
- *METADATA_FILE_EXTENSION*: Extension of the metadata files. This parameter is taken into account only if the ARCHIVE_TYPE is `DIRECTORY_LIST`. Default value: `LOT_AUTO`.

  The structure of a metadata file must be identical to that of a configuration file, i.e.

  keyword_code_1= value

  keyword_code_2= value

  ...

Where *ARCHIVE_TYPE* is not `DIRECTORY_LIST`, to specify the values of the metadata of the lot to archive, the KEYWORD_xx (keyword code, case sensitive) and VALUE_xx (metadata value) parameters are used, suffixing the name of these parameters with "_x" where x is a number. The first keyword should be set as 1 and thus successively for each new keyword to be informed.

Note

For date-type keywords, the format is dd/mm/yyyy.

REQUEST_PRIORITY: Priority of the generated archiving request, ranging from 1 (LOW) to 4 (IMMEDIATE), as described in Arcsys Functional Description Manual. Default value: `1` Possible values:`1 / 2 / 3 / 4`.

REQUEST_WAIT: Indicates if you want the client to wait for the request result (true) or not (false). Default value: `true`. Possible values: `true / false`.

14.6. etc/RestoreClient.properties File

14.6.1. Definition

This file contains the settings of the retrieval client.

14.6.2. Parameters for accessing the RMI API

Parameter name	**RMI_API_HOST_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Arcsys RMI API host name. If the client is on the same host as the RMI API, the `localhost` value is authorized.
Shipped value	`localhost`

Parameter name	**RMI_API_SERVICE_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Name of the Arcsys RMI API service. (it should be the same as COMPONENT_API_SERVICE_NAME in the RMI API properties file).
Shipped value	`ArcsysRmiAPI`

Parameter name	**RMI_API_PORT_NUMBER**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Arcsys RMI API port number.
Shipped value	`25040`

Parameter name	**RMI_API_USERNAME**
Parameter type	LDAP user. The user must exist in the used LDAP directory.
Parameter change assessment	After module reboot.
Description	Name of the RMI API LDAP user.
Shipped value	No shipped value.

Parameter name	**RMI_API_USERPASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the RMI API LDAP user.
Shipped value	No shipped value.

Parameter name	**KEYSTORE_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**KEYSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRUSTSTORE_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRUSTSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password for the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

14.6.3. Parameters for accessing the Application Agent

Parameter name	**APP_AGENT_HOSTNAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Host name of the application agent as referenced in the agent host field in the relational database, as shown in the agent list section of the website.
Shipped value	`localhost`

Parameter name	**APP_AGENT_PORT**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Application agent RMI port.
Shipped value	`25030`

Parameter name	**APP_AGENT_SERVICE_NAME**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Publication name of the RMI access interface to the application agent.
Shipped value	`ArcsysAgentAPI`

14.6.4. Retrieval Parameters

- *REST_DIR*: Retrieval directory. This parameter is only taken into account by FILE or DIRECTORY type objects. If empty, the files are retrieved to their original location.
- *RESTORE_TYPE*: Retrieval type. This parameter can take two values:
  - LOT (default value): Retrieval of a lot based on its ID.
  - *CRITERIA_FILES*: Retrieve lots based on XML search files. These files are renamed during the retrieval process. The extension indicates the progress status:
    - .cur: process in progress,
    - .err: process ending in error,
    - .old: process completed without errors.
- *LOT_ID*: Identifier of the lot to be retrieved (if RESTORE_TYPE = LOT).

- *INPUT_DIR*: Folder where the search files can be read (if RESTORE_TYPE = CRITERIA_FILES).

- *REQUEST_PRIORITY*: Priority of the generated archiving request, ranging from 1 (LOW) to 4 (IMMEDIATE), as described in Arcsys Functional Description Manual. Default value: `1`. Possible values: `1 / 2 / 3 / 4`.

- *REQUEST_WAIT*: Indicates if you want the client to wait for the request result (true) or not (false). Default value: `true`. Possible values: `true / false`.

14.7. etc/SearchClient.properties File

14.7.1. Definition

This file contains the settings of the search client.

14.7.2. Parameters for accessing the RMI API

Parameter name	**RMI_API_HOST_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Arcsys RMI API host name. If the client is on the same host as the RMI API, the `localhost` value is authorized.
Shipped value	`localhost`

Parameter name	**RMI_API_SERVICE_NAME**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Name of the Arcsys RMI API service. (it should be the same as COMPONENT_API_SERVICE_NAME in the RMI API properties file).
Shipped value	`ArcsysRmiAPI`

Parameter name	**RMI_API_PORT_NUMBER**
Parameter type	Positive integer.
Parameter change assessment	After module reboot.
Description	Arcsys RMI API port number.
Shipped value	`25040`

Parameter name	**RMI_API_USERNAME**
Parameter type	LDAP user. The user must exist in the used LDAP directory.
Parameter change assessment	After module reboot.
Description	Name of the RMI API LDAP user.
Shipped value	No shipped value.

Parameter name	**RMI_API_USERPASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the RMI API LDAP user.
Shipped value	No shipped value.

Parameter name	**KEYSTORE_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**KEYSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRUSTSTORE_FILE**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**TRUSTSTORE_PASSWORD**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password for the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

14.7.3. Search Parameters

- COL_ID: Identifier of the collection where the search is to be performed.
- To specify metadata values to search, use KEYWORD_xx triplets (keyword code; note this is case-sensitive), VALUE_xx (value searched) and OPERAT_xx (operator; possible values: = (equal to) / =* (starts with) / <>* (does not start with) / *= (finishes with) / *<> (does not finish with) / *=* (contains) / *<>* (does not contain) /< (lesser than) / <= (lesser than or equal to / <>(different to) / > (greater than) / >= (greater than or equal to)) by suffixing the name of these parameters by "_x" where x is a number. The first keyword should be set as 1 and thus successively for each new search keyword.

Note

For date-type keywords, the format is dd/mm/yyyy.

14.8. archiveClient.sh (or .bat) Script

This script starts up the archiving client. The Arcsys RMI API has to be launched for it to work.

14.9. restoreClient.sh (or .bat) Script

This script starts up the retrieval client. The Arcsys RMI API has to be launched for it to work.

14.10. searchClient.sh (or .bat) Script

This script starts up the search client. The Arcsys RMI API has to be launched for it to work.

15. ArcsysBatchs batch module

ArcsysBatchs batch module are a set of Java programs that perform batch tasks in Arcsys.

Batch applications can be divided in two sets : those dealing with metadata and those dealing with permissions.

Batch applications dealing with metadata allow to:

- export and check Arcsys metadata
- update Arcsys metadata
- update Arcsys controlled keywords

Batch applications dealing with permissions allow to:

- transfer permissions for an Arcsys group
- transfer permissions for an Arcsys user
- add rights to every lot of a collection
- delete rights from every lot of a collection

15.1. Batchs dealing with metadata

15.1.1. Export Batch

The `export` batch exports metadata from one or multiple Arcsys repositories (according to set criteria), and checks their validity.

When checking validity (if activated), the export can generate a log file for the update batch. For each metadata with an error, the generated log file indicates the keyword name and and its invalid value, adding a tag for update.

Data can be exported as CSV or XML formats; the log file is only available in XML format.

In order to export metadata with `export` script, the `etc/ARCSYS_EXPORT.properties` and `etc/MAPPING.properties` files need to be set with the export criteria.

15.1.2. Update Batch

The `update` batch updates Arcsys metadata. The update uses an XML file in the same format as the logs generated by the `export` batch. This XML file contains the corrected values of the metadata.

15.1.3. Update Keywords Batch

The `updatekeywords` batch adds values to the controlled keywords in Arcsys.

This batch is based on a CSV file.

15.1.4. Execution Characteristics

15.1.4.1. Preparation

For update batch applications (`update` and `update keywords` batches), it is recommended to create a mapping adapted to the desired modifications. This mapping is used to perform an export with validation of the metadata.

15.1.4.2. Update Tests

Both update batch applications (`update` and `update keywords`) have a DEBUG mode (`UPDATE_MODE` parameter), which is extremely useful to test an update before running this update. The same outputs are generated whether the update is done or not.

15.1.4.3. Files

Configuration

In the specific configuration files, the parameters with lists of values must not contain any additional spaces before or after the separators.

For Arcsys keywords containing a space, this space must be replaced by a "_" in the batch configuration.

Input

The CSV and XML files used as input in the batch applications must be created in UTF-8 format in order to be interpreted correctly.

Output

The CSV files generated by the batches (export and update keywords) contain an "X-" prefix, where X represents a continue increment starting at 1. The increment limits the number of rows in a single CSV file, so it can be opened in spreadsheet applications limited to 65536 rows.

15.2. Batches dealing with permissions

15.2.1. Update Group Batch

The `changegroup` batch transfers a set of permissions from one Arcsys group to another.

The script is used as follows:

```
changegroup.sh <group_to_replace> <new_group>
```

where:

- `<group_to_replace>` is the name of the group from which permissions are transferred,
- `<new_group>` is the name of the group that receives the permissions.

15.2.2. Update User Batch

The `changeuser` batch transfers a set of permissions from one Arcsys user to another.

The script is used as follows:

```
changeuser.sh <user_to_replace> <new_user>
```

where:

- `<user_to_replace>` is the name of the user that needs to be changed
- `<new_user>` is the name of the user that receives the permissions

15.2.3. Add Rights to Every Lot of a Collection Batch

This batch adds rights for a user or a group to each lot of a collection.

The script is used as follows:

```
addlotpermissionforcollection.sh (-u <user> | -g <group> | --existing-users-and-groups) --
collid <collection_id> [--consult] [--archive] [--restore] [--download] [--management] [--
ifHasConsult] [--ifHasArchive] [--ifHasRestore] [--ifHasDownload] [--ifHasManagement] [--
maxLot <maxLot>] [--startLotId <startLotId>] [--lots-by-transaction <lots_by_transaction>]
 [--domain <lots_domain>]
```

where:

- `<user>` is the name of the user

- `<group>` is the name of the group
- `--existing-users-and-groups` can be used instead of the -u or -g options (these options are exclusive) when it is required to add permissions on all users or groups encountered on each archive ; for example if an archive has a permission on user `user1` and another permission on group `group1`, the permission will be added to `user1` and also to `group1`
- `<collection_id>` is the identifier of the collection
- `--consult` adds visibility rights
- `--archive` adds archive and visibility rights
- `--restore` adds restore and visibility rights
- `--download` adds download and visibility rights
- `--management` adds archive, restore, download, visibility and management rights
- `--ifHasConsult` adds the requested rights to the user or group only if the user or group has the visibility rights
- `--ifHasArchive` adds the requested rights to the user or group only if the user or group has the archive rights
- `--ifHasRestore` adds the requested rights to the user or group only if the user or group has the restore rights
- `--ifHasDownload` adds the requested rights to the user or group only if the user or group has the download rights
- `--ifHasManagement` adds the requested rights to the user or group only if the user or group has the management rights
- `<maxLot>` sets the maximum number of lots to modify in one execution. This parameter counts the lots that are effectively modified. If `--maxLot` option is not specified, all lots are processed
- `<startLotId>` sets the lot at which the script should start adding permissions. To know which lot was processed last, you can search for the INFO log "Created permission for lot LOTID[...]" in the logs.
- `<lots_by_transaction>` is the maximal number of lots processed in each transaction. If `--lots-by-transaction` option is not specified, `1000` lots are processed
- `<lots_domain>` is the domain of the lots to be processed. The possible values are `business`, `technical` or `both` (to process both business and technical lots). If the `--domain` option is not specified, only the business lots are processed.

Note

When adding permissions, if the specified group or user does not have the visibility permission on the collection containing the lots and/or on the repository containing this collection, the permission is automatically added. This is only applied when a specific group or user is defined, not in the case of the `--existing-users-and-groups` option.

Important

If the RMI user specified in the `etc/RMI.properties` file is neither `GRANT_ALL` nor `ADMIN`, the execution of `addlotpermissionforcollection` batch will fail with an error message.

Important

The `addlotpermissionforcollection` script does not check that the user or group exists.

15.2.4. Delete Rights from Every Lot of a Collection Batch

This batch deletes rights for a user or a group from every lot of a collection.

The script is used as follows:

```
deletelotpermissionforcollection.sh (-u <user> | -g <group>) --collid <collection_id>
 [--all] [--archive] [--restore] [--download] [--management] [--lots-by-transaction
 <lots_by_transaction>] [--domain <lots_domain>]
```

where:

- `<user>` is the name of the user
- `<group>` is the name of the group
- `<collection_id>` is the identifier of the collection
- `--all` removes visibility, archive, restore, download and management rights
- `--archive` removes archive and management rights
- `--restore` removes restore and management rights
- `--download` removes download and management rights
- `--management` removes management rights

- `<lots_by_transaction>` is the maximal number of lots processed in each transaction. If `--lots-by-transaction` option is not specified, `1000` lots are processed
- `<lots_domain>` is the domain of the lots to be processed. The possible values are `business`, `technical` or `both` (to process both business and technical lots). If the `--domain` option is not specified, only the business lots are processed.

Important

If the RMI user specified in `etc/RMI.properties` file is neither `GRANT_ALL` nor `ADMIN`, the execution of `deletelotpermissionforcollection` batch will fail with an error message.

Important

- **The `deletelotpermissionforcollection` script does not check that the user or group exists.**
- **The `deletelotpermissionforcollection` script does not check that there is still a management permission on each lot.**

15.3. General Operation

Batch configuration enables you to:

- launch some of the batch applications (`update` and `update keywords` ones) in `DEBUG` mode, meaning no update is performed (see *`UPDATE_MODE`* parameter)
- export data in CSV or XML format
- restrict the export to archived data
- restrict the export to a list of data owner-user;
- check each collection of a repository

Note

To provide more flexibility in their use, it is recommended you use specific configuration files if several environments/repositories are targeted.

15.3.1. Operation Traceability

All tasks performed by the batch applications are written to log files using the log4j library (already used in the Arcsys application). These logs supplement the logs produced by the Arcsys application.

15.3.2. Execution

Important

Each batch application is a separate program. However it is not recommended you run the programs simultaneously (regardless of whether they are the same batch or not) if they target the same repositories or collections. Batches are designed to be run occasionally, and for update batches, outside Arcsys activity.

15.3.3. Return code

The batch applications all return the same return codes after performing the task:

CODE	DEFINITION
0	The process has been performed without error.
1	An error occurred during processing.

In the event of an error (return code other than 0), the logs display the detail of problems encountered.

15.3.4. Restrictions

The batch applications are designed to perform bulk tasks in Arcsys. However, these jobs are limited by the RMI APIs capabilities and certain functionalities have yet to be developed:

- Replacement batch applications are limited to repository, collection and lot permissions. More specifically, it is not possible to change the owner of a record.
- The export and update batches do not process metadata with multiple values; in export, only one value will be retrieved. For import, its behavior is not guaranteed; one or more values may be updated.

15.4. Directory Structure

Type	Name	Description
	`etc`	Directory containing the files
	`ARCSYS_EXPORT.properties`	Configuration file of the metadata export batch
	`ARCSYS_UPDATE_KEYWORDS.properties`	Configuration file of the update keywords batch

	ARCSYS_UPDATE.properties	Configuration file of the metadata update batch
	log4j2.xml	log4j configuration file.
	MAPPING.properties	Configuration file of the mapping to be used to validate the metadata
	RMI.properties	Configuration file for connection to the Arcsys RMI API
	jar	Directory containing the application core (arcsys-batchs.jar) as well as the Arcsys Java (jar) libraries of the module.
	libext	Directory containing the third-party Java libraries (jar) of the module.
	log	Directory containing a list of all the log files.
	xsd	Directory of the XSD files used to validate transfers.
	addlotpermissionforcollection.sh (or .bat)	Script to add rights for each lot of a collection
	changegroup.sh (or .bat)	Script to replace permissions of an Arcsys group
	changeuser.sh (or .bat)	Script to replace permissions of an Arcsys user
	deletelotpermissionforcollection.sh (or .bat)	Script to delete rights for each lot of a collection
	export.sh (or .bat)	Script to export metadata
	update.sh (or .bat)	Script to update metadata
	updatekeyword.sh (or .bat)	Script to update controlled keywords

15.5. etc/log4j2.xml file

15.5.1. Definition

This file contains the settings for the log file. The logs are generated with the log4J Apache library.

15.5.2. Parameters

The parameters of this file depend on the information desired in the logs. Hence please consult the standard documentation of Log4J for any changes.

See: https://logging.apache.org/log4j/2.x/

15.6. RMI.properties File

The `RMI.properties` file defines the connection to the Arcsys RMI API.

15.6.1. Parameters

The main parameters are:

- *SERVER* is the Arcsys RMI API host name.
- *PORT* is the Arcsys RMI API port number.
- *TARGET* is the Arcsys RMI API service name.
- *USERNAME* is the Arcsys RMI API connection user name.
- *PASSWORD* is the Arcsys RMI API connection password (encrypted).
- *DELAY* indicates the maximum authorized timeout (in ms) for a RMI transaction to the Arcsys API (by default, 60000).
- *SECURITY_PROTOCOL* indicates the security protocol to be used for inbound and outbound RMI connections (to or from the API). Possible values: `SSL` /`NONE`. (default: blank).
- *TRUSTSTORE_FILE*: location of the truststore file.
- *TRUSTSTORE_PASSWORD*: encrypted password of the truststore file (this is a passphrase).
- *KEYSTORE_FILE*: location of the keystore file.
- *KEYSTORE_PASSWORD*: encrypted password of the keystore file.

15.7. EXPORT.properties File

Apart from the common files, the export batch uses the `ARCSYS_EXPORT.properties` file. This file contains some parameters to filter the export, and other parameters to format the output file (in XML or CVS depending on the configuration).

15.7.1. Filtering the Export

- *`BASE_LIST`* Indicates the list of Arcsys repositories targeted by the export (comma as separator). By default or where left blank, all repositories are selected.
- *`COLLECTION_LIST`* Indicates the list of Arcsys collections targeted by the export (comma as separator). By default or where left blank, all collections of the repositories selected are taken.
- *`CREATOR_LIST`* Indicates the list of lot owners targeted by the export (comma as separator). By default or where left blank, all lots of the collections selected are taken.
- *`KEYWORD_LIST`* Indicates the list of keywords to extract from Arcsys for export (comma as separator). The protected keywords must not be listed here (see *`CSV_EXPORT_LIST`*).
- *`ARCHIVED_ONLY`* Indicates whether the export only contains archived lots (default: `true`). When set to `false`, both archived and non-archived lots are exported.
- *`OBJECT_EXPORT`* Indicates whether the export also contains the metadata of objects (default: `false`).
- *`CHECK_VALIDITY`* Indicates whether the validity check must be performed on exported lots / objects (default: `false`). When set to `true`, `MAPPING.properties` and XSD files will be used (for more details, see MAPPING.properties File or XSD File chapters).

15.7.2. XML Logs

By default, XML export is activated.

- *`GENERATE_LOG`* Indicates whether the validity check must generate logs for each error found (default: `false`).
- *`LOG_FILENAME`* Indicates the name of the XML file to use for the export of validation errors (default: `log.xml`, if activated).

15.7.3. Export

By default, the export is an XML export.

- *FILENAME* Indicates the filename to use for export. The file extension (xml or csv) is automatically generated. Note that this filename can be specified with a directory path, and if this directory does not exist, it will be automatically created.

- *CSV_EXPORT* Indicates if the export should be done as CSV instead of XML (default: `false`).

- *CSV_HEADER_LIST* Indicates the list of column headings to use in the exported CSV file (comma as separator, no spaces).

  If *CSV_EXPORT_LIST* is not defined, *CSV_HEADER_LIST* and *KEYWORD_LIST* must have the same number of elements.

- *CSV_EXPORT_LIST* Indicates the list of keywords to extract from Arcsys for export (comma as separator, no spaces).

  CSV_EXPORT_LIST and *KEYWORD_LIST* must have the same number of elements. If this parameter is not defined, the keyword list is used (see *KEYWORD_LIST*).

Note

Among the list of keywords to be listed for export, it may be necessary to add protected keywords for information purposes, such as `base`, `id_base`, `collection`, `id_collection`, `id_lot`, `lot`, `id_object`, `object`, `lot_creator`, `is_archived`, `is_valid`.

- *CSV_DATEHOUR_FORMAT* Indicates the export format of dates stored as metadata (default: " `dd-MM-yyyy`").

- *CSV_SEPARATOR* Specifies the separator used in the CSV export file (default: ";").

15.8. MAPPING.properties File

If data validation is activated, it uses the `MAPPING.properties` file. It uses the following parameters:

- *XSD_ROOT* Indicates the root tag used in the XML file to be generated for the validation of a lot / object (default: `Arcsys`).

- *XSD_META* Indicates the tag used for grouping metadata in the XML file to be generated for the validation of a lot / object (default: `Meta`).

- *XSD_FOLDER* Indicates the storage folder of the XSD to be used for the validation of a lot / object (default: `./`).

Apart from these parameters, a mapping relies on the following elements (which can be repeated several times in the `properties`) file:

- *BASE_X* Indicates the name of the target repository of the mapping (mandatory).
- *COLL_X* Indicates the list of target collections of the mapping (comma separator).
- *KEYWORDS_LOT_X* Indicates the list of keywords at lot level to extract for the validation of the mapping (comma as separator, no spaces) (mandatory).
- *NAMES_LOT_X* Indicates the list of names of the keywords at lot level to extract for the validation of the mapping (comma as separator, no spaces) (mandatory).
- *MANDATORIES_LOT_X* Indicates the list of mandatory keywords at lot level for the validation of the mapping (comma as separator, no spaces).
- *XSDFILE_LOT_X* Indicates the name of the XSD file to use for the validation of the mapping, at lot level.
- *KEYWORDS_OBJ_X* Indicates the list of keywords at object level to extract for the validation of the mapping (comma as separator, no spaces) (mandatory).
- *NAMES_OBJ_X* Indicates the list of names of the keywords at lot level to extract for the validation of the mapping (comma as separator, no spaces) (mandatory).
- *MANDATORIES_OBJ_X* Indicates the list of mandatory keywords at object level for the validation of the mapping (comma as separator, no spaces).
- *XSDFILE_OBJ_X* Indicates the name of the XSD file to be used for the validation of the mapping, at object level.

Important

For all these parameters, the "X" suffix is replaced by a continue increment (starting at 1), identical for all parameters of the same mapping.

15.9. Update.properties File

Apart from the common files, the update batch application uses the export mapping file (`MAPPING.properties`) and the `ARCSYS_UPDATE.properties` file, which contains the following parameters:

15.9.1. Parameters

- *KEYWORD_LIST* Indicates the list of keywords to extract from Arcsys for validation of the import (comma as separator). The protected keywords should not be listed here (see export - CSV_EXPORT_LIST).

- *UPDATE_MODE* Indicates whether the updates should be applied to the target (default: false).
- *FILENAME* Name of the XML file containing the keyword updates (the structure of the file is specified in the user guide) (default: "update.xml").
- *CSV_SEPARATOR* Specifies the separator used in the CSV export file (default: ";").

15.10. XSD File

The export and update batches use the mapping configuration file to validate the data. The configuration is based on the XSD files, used to validate the metadata extracted or to update the lots or objects.

The keyword types used to index the data determine the checks performed by Arcsys as regards the keyword value. The export batch checks the archiving done; the update batch updates the metadata, according to the validation.

The validation of metadata can be performed at lot and/or lot object level; the batches perform separate validations for both levels. For each collection (or list of collections, as required) of the mapping, you must therefore create two XSDs: one for lot level and one for file level.

15.10.1. Example

To illustrate XSD building, the following example is based on a collection "COL" whose indexing mask is as follows:

Nom mot-clé	Type	Obligatoire	Niveau du mot-clé	Supp
REGION	Chaîne	oui	Lot	
METIER	Chaîne	non	Lot	
MOIS	Numérique	oui	Lot	
ANNEE	Numérique	oui	Lot	
DATE LANCEMENT	Date	oui	Lot	

Figure 15.1. Mask of the collection COL.

You can freely choose the name for the XSD. Nonetheless, it is recommended you call the lot-level validation XSD `nom_collection.xsd`. The file-level validation XSD must use the name of the lot-level XSD, prefixed by `OBJ_`, i.e. in the example `OBJ_nom_collection.xsd`.

15.10.2. Lot-Level XSD

The lot-level validation XSD for the collection uses the five keywords of the collection mask:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="ROOT" type="RootType" />
    <xsd:complexType name="RootType">
        <xsd:all minOccurs="1" maxOccurs="1">
            <xsd:element name="META" type="MetaType" minOccurs="1" maxOccurs="1" />
        </xsd:all>
    </xsd:complexType>
    <xsd:complexType name="MetaType">
        <xsd:all minOccurs="1" maxOccurs="1">
            <xsd:element name="REGION" type="TenStringType" minOccurs="1" maxOccurs="1" />
            <xsd:element name="METIER" type="TwentyStringType" minOccurs="0" maxOccurs="1" />
            <xsd:element name="MOIS" type="TwoIntegerType" minOccurs="1" maxOccurs="1" />
            <xsd:element name="ANNEE" type="FourIntegerType" minOccurs="1" maxOccurs="1" />
            <xsd:element name="DATE_LANCEMENT" type="dateOrDateTimeType" minOccurs="1" maxOccurs="1" />
        </xsd:all>
    </xsd:complexType>
    <xsd:simpleType name="dateOrDateTimeType">
        <xsd:union memberTypes="xsd:date xsd:dateTime"/>
    </xsd:simpleType>
    <xsd:simpleType name="TenStringType">
        <xsd:restriction base="xsd:string">
            <xsd:minLength value='1' />
            <xsd:maxLength value='10' />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="TwentyStringType">
        <xsd:restriction base="xsd:string">
            <xsd:minLength value='1' />
            <xsd:maxLength value='20' />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="TwoIntegerType">
        <xsd:restriction base="xsd:integer">
            <xsd:minExclusive value='-100' />
            <xsd:maxExclusive value='100' />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="FourIntegerType">
        <xsd:restriction base="xsd:integer">
            <xsd:minExclusive value='-10000' />
            <xsd:maxExclusive value='10000' />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

Figure 15.2. Lot-Level XSD Schema for the COL Collection.

A number of important elements are to be noted in this file:

- The root tag of the document is configured in the mapping (XSD_ROOT parameter),
- The "META" tag (*XSD_META* parameter) contains the various keyword tags defined for the mask,
- The "`META`" tag type lists the keyword tags in no particular order (use of "<xsd:all />"),
- Set the "`minOccurs`" attribute to 1 to make a keyword tag mandatory.

In addition, the XSD is used to refine the restrictions defined by Arcsys for the mask keywords:

- Definition of string length: the "`TenStringType`" type limits the string to between 1 and 10 characters,
- Definition of limits between integers: the "`TwoIntegerType`" type limits the integer to two figures,
- Management of date time: the "`dateOrDateTimeType`" type accepts dates with or without the time.

15.10.3. File-Level XSD

As the mask for the collection does not define a file-level keyword, the following XSD does not define any tags at this level:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="ROOT" type="RootType" />
    <xsd:complexType name="RootType">
        <xsd:all>
            <xsd:element name="META" type="MetaType" minOccurs="0" maxOccurs="1" />
        </xsd:all>
    </xsd:complexType>
    <xsd:complexType name="MetaType">
        <xsd:all />
    </xsd:complexType>
</xsd:schema>
```

Figure 15.3. File-level XSD schema for the collection COL

In this XSD, the "META" tag does not contain any elements; it may therefore be absent from the XML at file level.

15.11. Input and Output Files

15.11.1. Export

15.11.1.1. Output Files

The XML output files all have the same structure, including protected keyword values. The list of keywords extracted from Arcsys can vary.

For example, the following result shows this structure with validation:

```
<?xml version="1.0" encoding="UTF-8"?>
<EXPORT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <BASE_LIST>
    <BASE ID="3">
      <CODE>OnDemand</CODE>
      <COLLECTION_LIST>
        <COLLECTION ID="3">
          <CODE>COL</CODE>
          <LABEL>Base de test XML</LABEL>
          <LOT_LIST>
            <LOT ID="1114">
              <CODE>ONDEMAND_20110902141806466</CODE>
              <CREATOR>end_user</CREATOR>
              <LABEL>Archivage en exemple complet</LABEL>
              <IS_ARCHIVED>false</IS_ARCHIVED>
              <META_DATA_LIST>
                <META_DATA>
                  <KEYWORD>ANNEE</KEYWORD>
                  <VALUE>2010</VALUE>
                </META_DATA>
                <META_DATA>
                  <KEYWORD>DATE_LANCEMENT</KEYWORD>
                  <VALUE>2010-11-10</VALUE>
                </META_DATA>
                <META_DATA>
                  <KEYWORD>METIER</KEYWORD>
                  <VALUE>FOURNISSEUR</VALUE>
                </META_DATA>
                <META_DATA>
                  <KEYWORD>REGION</KEYWORD>
                  <VALUE>0000000001</VALUE>
                </META_DATA>
                <META_DATA>
                  <KEYWORD>MOIS</KEYWORD>
                  <VALUE>11</VALUE>
                </META_DATA>
              </META_DATA_LIST>
              <IS_VALID>true</IS_VALID>
            </LOT>
```

Figure 15.4. Output Example of an XML Export.

The CSV output files do not have any specific structure. According to the keywords chosen (as well as protected keywords), the output file can vary.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	base	id_base	collection	id_collection	id_lot	lot	REGION	METIER	MOIS	ANNEE	DATE_LANCEMENT	lot_creator	is_archived	is_valid
2	OnDemand	3	COL	3	1114	ONDEMAND_20110902141806466	1	FOURNISSEUR	11	2010	10/11/2010	end_user	false	false
3	OnDemand	3	COL	3	134	ONDEMAND_20110330173416763	1	FOURNISSEUR	11	2010	10/11/2010	admin	false	false
4	OnDemand	3	COL	3	133	ONDEMAND_20110330172042357	1	FOURNISSEUR	11	2010	10/11/2010	admin	false	false
5	OnDemand	3	COL	3	132	ONDEMAND_20110330170546694	1	FOURNISSEUR	11	2010	10/11/2010	admin	false	false
6	OnDemand	3	COL	3	131	ONDEMAND_20110330162724471	1	FOURNISSEUR	11	2010	10/11/2010	admin	false	false
7	OnDemand	3	COL	3	129	ONDEMAND_20110329103456839	1	FOURNISSEUR	11	2010	10/11/2010	admin	false	false
8	OnDemand	3	COL	3	128	ONDEMAND_20110329102046559	1	FOURNISSEUR	11	2010	10/11/2010	admin	false	false

Figure 15.5. Output Example of a CSV Export (in table).

15.11.1.2. Validation Errors File

For a validation error (and according to configuration), an error file is generated. It always has the same structure.

For example, the result below shows this structure with a modification of the XSD (`TenStringType` changed to 9 characters):

```
<?xml version="1.0" encoding="UTF-8"?>
<EXPORT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ERROR_LIST>
    <LOT ID="1114" BASE="OnDemand" COLLECTION="COL">
      <META_DATA>
        <KEYWORD>REGION</KEYWORD>
        <VALUE>0000000001</VALUE>
        <NEW_VALUE></NEW_VALUE>
      </META_DATA>
    </LOT>
    <LOT ID="134" BASE="OnDemand" COLLECTION="COL">
      <META_DATA>
        <KEYWORD>REGION</KEYWORD>
        <VALUE>0000000001</VALUE>
        <NEW_VALUE></NEW_VALUE>
      </META_DATA>
    </LOT>
    <LOT ID="133" BASE="OnDemand" COLLECTION="COL">
      <META_DATA>
        <KEYWORD>REGION</KEYWORD>
        <VALUE>0000000001</VALUE>
        <NEW_VALUE></NEW_VALUE>
      </META_DATA>
```

Figure 15.6. Output Example of an XML Log File.

15.11.2. Update

The input file of the update batch is no other than the validation errors output file from the export batch. Before using it, a *NEW_VALUE* must be entered for each error.

15.11.3. Update Keywords

15.11.3.1. Input File

The file used to update controlled keywords is a simple CSV file. The structure of this file is similar to that generated by the export batch (CSV), without the columns indicating the protected keywords.

Only controlled keywords are updated. Controlled values are not replaced, rather new controlled values are added. The input file therefore may contain only one column with the name of the controlled keyword as the heading, and on each subsequent row, a new value to add.

Like for the export file, the file must be complete; the rows must have the same number of elements (possibly blank). In the same way, the last column of each row must not contain any additional separators.

15.11.3.2. MISSING Output File

If necessary, a `X-MISSING.csv` file may be generated on running the update batch. This file contains as a header the list of keywords with fewer values in Arcsys compared to the update file, and the missing values in each row that follows. The file generated may contain empty values if several keywords are updated simultaneously.

15.11.3.3. MORE Output File

If necessary, a `X-MORE.csv` file may be generated on running the update batch. This file contains as a header the list of keywords with more values in Arcsys compared to the update file, and the additional values in each row that follows. The file generated may contain empty values if several keywords are updated simultaneously.

16. Signature validator module

Important

This features set can only be used by buying ArcVerif.

The Signature validator module is installed in a directory called `signature-validator-ws`. This module is an embedded web server hosting a web service, that allows to validate electronic signatures of the documents analyzed by ArcFF module in Arcsys.

16.1. Directory Structure

Type	Name	Description
	`bin`	Directory containing the scripts of the module. The `appenv.sh(cmd)`, `runner-java.sh(cmd)`, `setenv.sh(cmd)`, `util.sh(cmd)` scripts are internal scripts, write-protected and should not be modified. The `overenv.sh(cmd)` script can be used to override some of the environment configuration needed for the module to work, as explained below.
	`cert-to-import`	Directory containing the certificates that may be required to access data for the validation of specific signatures. These certificates must be imported into the truststore used by the application.
	`anssi.p7c`	Certificate that allows the access to the ANSSI (https://www.ssi.gouv.fr/[1]) french trusted list.
	`etc`	Directory containing the module configuration files.
	`log4j2.xml`	Main log4j configuration file.
	`signature-validator-ws.properties`	Main configuration file of the module.

[1] https://www.ssi.gouv.fr/

	`lib`	Directory containing the module Java libraries (jar).
	`signature-validator-ws.sh (or cmd)`	Main script of the module permitting `start`, `stop` and `status` operations.

16.2. bin/overenv.sh (or cmd) Script

This script can be modified and used to override the environment configuration of the Signature validator module that are defined in the internal scripts of the module.

To be executable, the module needs three specific JVM parameters to be defined:

- `log4j2.configurationFile` to specify the Log4j configuration file path. By default, this parameter has the value `etc/log4j2.xml` which is a relative path from the Signature validator module root path.
- `sigws.configuration.file` to specify the module main configuration file path. By default, this parameter has the value `etc/signature-validator-ws.properties` which is a relative path from the Signature validator module root path.
- `sigws.secret.key.file` to specify the path of the symmetrical private key that is used to encrypt the value of any parameter of the module. By default, this parameter has the value `etc/secret-key` which is a relative path from the Signature validator module root path. There is no key delivered with the module and as explained in the Arcsys Installation Manual, you can either generate a new key or use an existing one from your Arcsys installation.

As you can see in the `appenv.sh(cmd)` script, these JVM parameters are defined in the `APP_OPTIONS` environment variable. You can also see that for each of these parameters, an environment variable is used to define their value (`APP_LOG_CONFIG_FILE`, `APP_CONFIG_FILE`, `APP_SECRET_KEY_FILE`).

So you can, for example, modify these three JVM parameters at once by overriding the environment variable `APP_OPTIONS` in the `overenv.sh(cmd)` script:

```
APP_OPTIONS="-Dlog4j2.configurationFile=/home/user/signature/log4j2.xml -Dsigws.
configuration.file=/home/user/signature/signature-validator-ws.properties -Dsigws.secret.
key.file=/home/user/signature/secret-key"
```

Otherwise, you can override only one or more of these JVM parameters values, for example the secret key path:

```
APP_SECRET_KEY_FILE="/home/user/signature/secret-key"
```

16.3. etc/log4j2.xml file

16.3.1. Definition

This file contains the settings for the log files. The logs are generated with the Apache library log4j.

16.3.2. Parameters

The parameters of this file depend on the information desired in the logs. For any changes, consult the standard documentation of Log4J.

See: https://logging.apache.org/log4j/2.x/

16.4. etc/signature-validator-ws.properties file

16.4.1. Definition

This file contains the main parameters for the module.

16.4.2. Parameters

Parameter name	**sigws.security.protocol**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	The protocol used for exchanges with the Signature validator module. Possible values are: • `NONE`: Interactions with the Signature validator module are conventional HTTP exchanges. • `SSL`: Exchanges with the Signature validator module must be secured via SSL.
Shipped value	`NONE`

Parameter name	**sigws.http.port**
Parameter taken into account only if:	*`sigws.security.protocol`*`=NONE`
Parameter type	Port number
Parameter change assessment	After module reboot.
Description	HTTP port used for the REST web services of Signature validator module.
Shipped value	`8086`

Parameter name	**sigws.https.port**
Parameter taken into account only if:	*sigws.security.protocol*=SSL
Parameter type	Port number
Parameter change assessment	After module reboot.
Description	HTTPS port used for the REST web services of Signature validator module.
Shipped value	8445

Parameter name	**sigws.stop.token**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Stop token of the embedded Jetty server. This token is represented by any character string but must be unique with respect to embedded Jetty tokens present on the machine.
Shipped value	9lFVfB9ES32=

Parameter name	**sigws.security.client.id**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Identifier of the client application for OAuth2 authentication on the Signature validator module. The value of this parameter is open-ended.
Shipped value	signature_validator

Parameter name	**sigws.security.client.secret**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Secret shared by identifier of the client application for OAuth2 authentication on the Signature validator module. The applications connecting to this module must have the same shared secret.
Shipped value	No shipped value.

Parameter name	**sigws.ssl.keystore.file**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**sigws.ssl.keystore.password**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**sigws.ssl.keystore.certalias**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Alias of the certificate in the keystore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**sigws.ssl.keymanager.password**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Key manager password (if any) used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**sigws.ssl.keystore.type**
Parameter type	Characters string. Possible values are: `jks` or `pkcs12`.
Parameter change assessment	After module reboot.
Description	Type of the keystore containing the certificate used for SSL/TLS exchanges.
Shipped value	`jks`

Parameter name	**sigws.ssl.truststore.file**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Location of the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**sigws.ssl.truststore.password**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password for the truststore used for SSL/TLS exchanges.
Shipped value	No shipped value.

Parameter name	**sigws.ssl.truststore.type**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Type of the truststore used for SSL/TLS exchanges.
Shipped value	`jks`

Parameter name	**sigws.tmp.dir**
Parameter type	Absolute path of a directory. The path must correspond to an existing directory, with adequate access rights (read, write and execute). Use Unix-like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Path to the temporary directory used by the application to store certificates, cache, uploaded files, ...
Shipped value	No shipped value.

Parameter name	**sigws.dss.truststore.file**
Parameter type	Absolute path of a file. Use Unix like separators ("/"), even on Windows.
Parameter change assessment	After module reboot.
Description	Optional truststore file which will be used by DSS to validate specific signatures. This file can contain specific certificates to validate these signatures.
Shipped value	No shipped value.

Parameter name	**sigws.dss.truststore.password**
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the optional truststore used by DSS.
Shipped value	No shipped value.

Parameter name	**sigws.dss.truststore.type**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Type of the optional truststore used by DSS.
Shipped value	`jks`

Parameter name	**sigws.dss.lotl.url**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Base URL used for the European trusted list retrieval.
Shipped value	`https://ec.europa.eu/tools/lotl/eu-lotl.xml`

Parameter name	**sigws.dss.official.journal.url**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Base URL used for official journal retrieval.
Shipped value	`https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.C_.2019.276.01.0001.01.ENG`

Parameter name	**sigws.dss.cache.crl.min**
Parameter type	Number of minutes.
Parameter change assessment	After module reboot.
Description	Cache duration in minutes for the CRL (Certificate Revocation List) responses. A duration value of zero or lower is interpreted as no caching.
Shipped value	`1440 (1 day)`

Parameter name	**sigws.dss.cache.ocsp.min**
Parameter type	Number of minutes.
Parameter change assessment	After module reboot.
Description	Cache duration in minutes for the OCSP (Online Certificate Status Protocol) responses. A duration value of zero or lower is interpreted as no caching.
Shipped value	`1440 (1 day)`

Parameter name	**sigws.dss.validation.level**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	This parameter allows to define the validation level (which rulesets are activated) to use for signature validation. The recommended level is `ARCHIVAL_DATA` (maximal level of the validation).
Possible values	`BASIC_SIGNATURES`, `TIMESTAMPS`, `LONG_TERM_DATA`, `ARCHIVAL_DATA`
Shipped value	`ARCHIVAL_DATA`

Parameter name	**sigws.dss.ssl.validation.enabled**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	This parameter allows to disable (`false`) or enable (`true`) the SSL certificates validation of the URLs used within the DSS process.
Shipped value	`false`

Parameter name	**sigws.dss.validation.policy**
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	This parameter allows to define the default validation policy (what is the severity of a failed rule) to use. This parameter is optional.
Possible values	`DEFAULT`, `IGNORE_ALL`, `IGNORE_CERTIFICATE`, `IGNORE_TIMESTAMP`
Shipped value	`DEFAULT`

Parameter name	**sigws.dss.offline.crl.enabled**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	This parameter allows to configure an offline CRL server. This parameter is optional.
Shipped value	`false`

Parameter name	**sigws.dss.offline.crl.url**
Parameter taken into account only if:	*`sigws.dss.offline.crl.enabled`*=`true`.
Parameter type	URL, formed like: `<protocol>://<domain_name or ip_address>[:<port_number>][/<uri>]`
Parameter change assessment	After module reboot.
Description	This parameter allows to configure the offline CRL server URL. This parameter is optional.
Shipped value	No shipped value.

Parameter name	**sigws.dss.offline.crl.user**
Parameter taken into account only if:	*sigws.dss.offline.crl.enabled*=true.
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	This parameter allows to configure the offline CRL server user. This parameter is optional.
Shipped value	No shipped value.

Parameter name	**sigws.dss.offline.crl.password**
Parameter taken into account only if:	*sigws.dss.offline.crl.enabled*=true.
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	This parameter allows to configure the offline CRL server password. This parameter is optional.
Shipped value	No shipped value.

Parameter name	**sigws.dss.offline.ocsp.enabled**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	This parameter allows to configure an offline OCSP server. This parameter is optional.
Shipped value	`false`

Parameter name	**sigws.dss.offline.ocsp.url**
Parameter taken into account only if:	*sigws.dss.offline.ocsp.enabled*=true.
Parameter type	URL, formed like: `<protocol>://<domain_name or ip_address>[:<port_number>][/<uri>]`
Parameter change assessment	After module reboot.
Description	This parameter allows to configure the offline OCSP server URL. This parameter is optional.
Shipped value	No shipped value.

Parameter name	**sigws.dss.offline.ocsp.user**
Parameter taken into account only if:	*sigws.dss.offline.ocsp.enabled*=true.
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	This parameter allows to configure the offline OCSP server user. This parameter is optional.
Shipped value	No shipped value.

Parameter name	**sigws.dss.offline.ocsp.password**
Parameter taken into account only if:	*sigws.dss.offline.ocsp.enabled*=true.
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	This parameter allows to configure the offline OCSP server password. This parameter is optional.
Shipped value	No shipped value.

Parameter name	**sigws.http.proxy.enabled**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Flag to indicate if a proxy must be used for the application HTTP outgoing connections.
Shipped value	`false`

Parameter name	**sigws.http.proxy.host**
Parameter taken into account only if:	*sigws.http.proxy.enabled*=true
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Host of the proxy used for the application HTTP outgoing connections.
Shipped value	No shipped value.

Parameter name	**sigws.http.proxy.port**
Parameter taken into account only if:	*sigws.http.proxy.enabled*=true
Parameter type	Port number
Parameter change assessment	After module reboot.
Description	Port of the proxy used for the application HTTP outgoing connections.
Shipped value	No shipped value.

Parameter name	**sigws.http.proxy.user**
Parameter taken into account only if:	*sigws.http.proxy.enabled*=true
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	User of the proxy used for the application HTTP outgoing connections.
Shipped value	No shipped value.

Parameter name	**sigws.http.proxy.password**
Parameter taken into account only if:	*sigws.http.proxy.enabled*=true
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the proxy used for the application HTTP outgoing connections.
Shipped value	No shipped value.

Parameter name	**sigws.http.proxy.excluded.hosts**
Parameter taken into account only if:	*sigws.http.proxy.enabled*=true
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	List of the hosts to exclude of the proxy used for the application HTTP outgoing connections (separated by ',' or ';' or ' ').
Shipped value	No shipped value.

Parameter name	**sigws.https.proxy.enabled**
Parameter type	Boolean which can be either `true` or `false` .
Parameter change assessment	After module reboot.
Description	Flag to indicate if a proxy must be used for the application HTTPS outgoing connections.
Shipped value	`false`

Parameter name	**sigws.https.proxy.host**
Parameter taken into account only if:	*`sigws.https.proxy.enabled`*=`true`
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	Host of the proxy used for the application HTTPS outgoing connections.
Shipped value	No shipped value.

Parameter name	**sigws.https.proxy.port**
Parameter taken into account only if:	*`sigws.https.proxy.enabled`*=`true`
Parameter type	Port number
Parameter change assessment	After module reboot.
Description	Port of the proxy used for the application HTTPS outgoing connections.
Shipped value	No shipped value.

Parameter name	**sigws.https.proxy.user**
Parameter taken into account only if:	*`sigws.https.proxy.enabled`*=`true`
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	User of the proxy used for the application HTTPS outgoing connections.
Shipped value	No shipped value.

Parameter name	**sigws.https.proxy.password**
Parameter taken into account only if:	*`sigws.https.proxy.enabled`*=`true`
Parameter type	Encrypted characters string. To encrypt a clear characters string, use the `encrypt` encryption utility provided in Arcsys Installation and Update Package or in Arcsys Engine. The encrypted characters string will be surrounded by `ENC(...)`.
Parameter change assessment	After module reboot.
Description	Password of the proxy used for the application HTTPS outgoing connections.
Shipped value	No shipped value.

Parameter name	**sigws.https.proxy.excluded.hosts**
Parameter taken into account only if:	*sigws.https.proxy.enabled*=true
Parameter type	Characters string.
Parameter change assessment	After module reboot.
Description	List of the hosts to exclude of the proxy used for the application HTTPS outgoing connections (separated by ',' or ';' or ' ').
Shipped value	No shipped value.

16.5. signature-validator-ws.sh (or cmd) Script

This script is the main script of the Signature validator module.

The script must be followed by a specific parameter to:

- Start the Signature validator module on localhost:

```
signature-validator-ws.sh (or cmd) start
```

- Stop the Signature validator module on localhost:

```
signature-validator-ws.sh (or cmd) stop
```

- Get the status of the Signature validator module on localhost:

```
signature-validator-ws.sh (or cmd) status
```

Possible statuses are:

Return code value	Signature Validator status	Description
94	STARTED_AND_WORKING	The signature validator is started and is able to handle requests.
96	SHUTDOWN	The signature validator is stopped.
99	GET_STATUS_ERROR	Error while trying to get the signature validator status.

- Get the Signature validator module script possible parameters:

```
signature-validator-ws.sh (or cmd) -h (or --help)
```

- Get the Signature validator module version:

```
signature-validator-ws.sh (or cmd) -v (or --version)
```

16.6. Import missing certificates

Generally the module (and more specifically DSS) downloads Trusted Lists and data used to validate signature by using the SSL protocol (for resources using HTTPS extension), that requires to have a certificate of a remote source in the Java trust store.

However, some of the certificates necessary for the recovery of these data could be missing or expired (in the case of an old JDK for example). In this case, your should have this kind of error in the module logs:

```
Unable to process GET call for url [https://xxx]. Reason : [PKIX path building failed: sun.
security.provider.certpath.SunCertPathBuilderException: unable to find valid certification
 path to requested target]
```

To solve this issue, it is necessary to firstly retrieve the missing certificate. This can be easily done by accessing the URL specified in these logs via your browser, accessing the certificate data and then exporting the certificate. Then, the easiest solution is to import this certificate into the JDK truststore (`cacerts` file) using the following command:

```
keytool -import -alias <certificate_alias> -file <certificate_path> -keystore
 <cacerts_path> -storepass <cacerts_password>
```

In this command, the `certificate_path` is the path of the `cacerts` file of your JDK (generally `<jdk_root_path>/lib/security/cacerts`) and the `cacerts_password` is `changeit` by default in JDKs.

For example, the certificate allowing access to the French trusted list is not present by default on some JDKs. This certificate is delivered with the module: `cert-to-import/anssi.p7c`. An example of command to import this certificate in the used JDK would be:

```
keytool -import -trustcacerts -alias anssi -file <signature_validator_path>/cert-to-import/
anssi.p7c -keystore <jdk_root_path>/lib/security/cacerts -storepass changeit
```

Another solution is to use your own truststore which should contain the certificates allowing access to the data to validate your signature. The disadvantage of this solution is that you will probably have to import some certificates that were already present in your JDK's truststore, such as the certificate allowing access to the European trusted list (https://ec.europa.eu/tools/lotl/eu-lotl.xml).

Once this truststore is ready to be used, it is necessary to fill in its properties in the `signature-validator-ws.properties` file via the `sigws.dss.truststore.*` options.

Glossary

Access Zone

An access zone is an independent entity within Arcsys that defines a controlled network area from which resources can be accessed. These entities can then be attached to permissions (at the repository, collection, lot, or class level) to restrict or grant access based on the client's IP address when authenticating to the Arcsys REST API, the Arcsys Web Agent or ArcWeb Module.

API *(Application Programming Interface)*

The APIs provided by Arcsys enable the product holder to fully customize a new application or user interface according to the specific ergonomic needs of their use case. Arcsys exposes several types of APIs:

- REST APIs are the recommended interface. They offer broad coverage of Arcsys's functionalities, including administration, operations, archiving, search, and archive retrieval.
- Legacy APIs based on RMI and SOAP protocols are still available for compatibility purposes but are deprecated and should no longer be used in new developments.

Application Agent

There are two different types of agents at archiving level: application interface agents and user interface agents. An **application agent** can archive all the objects specific to an application (files, RDBMS table records, etc.), whereas a **web agent** performs both administration functions and manual archiving functions initiated by the user.

Archiving By Reference

Archiving by reference is a method in which data remains in its original storage location when added to an archive system, and the system generates references and metadata entries for the files. Eventually, the files are transferred to the archive system's defined storage using the copy and migration mechanism.

Archive Restitution

Archive restitution is the return and transfer of archived documents to their originator, or to a duly appointed person or organization. An Archive Restitution is in Arcsys an Archive Retrieval operation that ends with a Destruction. An Archive restitution operation can be created through the appropriate operation in the REST API, or by using ArcWeb or ArcEP module.

See Also **Archive Retrieval**, **Destruction**.

Archive Retrieval

Archive retrieval is an operation that makes a copy of a record available to a record requester. This term takes precedence over the term *restore*, which has another meaning at archiving level (restore in the sense of handing back the documents to the organization that created them or to its representatives, then destroying them). Archive retrieval can be

complete (misleadingly called a "complete retrieval") or partial (*Partial Archive Retrieval*, misleadingly called a "partial retrieval").
See Also **Archive Restitution**.

Arcsys
ERM published by Infotel. Arcsys refers to both the Arcsys Core product and all of its connectors and options.

Arcsys Connector
An Arcsys connector is an operational module generally requiring an additional license used to interface with an external software package (ECM, ERP, Mail) for archiving and/or archive retrieval to and from Arcsys.

Arcsys Core
The Arcsys Core represents all "essential" Arcsys modules, which are: Arcsys Database, the Arcsys RMI, TCP/IP and SOAP API, the Arcsys REST API, the Arcsys Transfer Server, the Arcsys Transfer Service, the Arcsys Engine, the Arcsys Web Agent, the Arcsys Application Agent and the Arcsys Auto-Archive Agent.
See Also **Arcsys**.

Arcsys Engine
Central archiving platform on which synchronous and asynchronous archiving, indexing and retrieval processes operate. The engine can spread threads over multiple processors. This guarantees dialogue and traceability between the agents that are associated to it.

Arcsys Option
Arcsys options are added to the Arcsys Core for additional functionalities. They do not necessarily require an additional architectural module. They may be subject to a separate license. The main options are:

- ArcAFP Option (AFP format management)
- ArcMover Tape Option (media manager managing file systems and tape libraries)
- ArcIP (record ingestion)
- ArcEP (record extractor)
- ArcPak Option (record compression on ArcMover and native ingestion of compressed files)
- ArcRFT Option (full text search)
- ArcSign Option (internal digital signature generation) and ArcVerif (external digital signature verification)
- ArcCrypt Option (encryption of data at rest)

- ArcCFN (digital vault)
- ArcRef Option (record ingestion by reference)
- ArcMover S3 Option (media manager allowing to archive and retrieve data on any Cloud media compatible with the Amazon S3 REST API)

Attestation policy
An attestation policy allows to define which type of attestation can be generated as well as a set of parameters concerning their generation.

Classification Scheme
A classification scheme in archiving and digital preservation refers to an organized framework for categorizing records and archival materials based on a hierarchical structure. It facilitates systematic retrieval, management, and preservation of information. In the context of Arcsys, the classification scheme is defined as the structural entity that contains a hierarchy of classes. These classes are used for organizing archives and records and for implementing specific archival policies such as retention schedules and format management. Within Arcsys, a classification scheme is linked to a specific repository, providing an organizational backbone for multiple collections. It also serves as a navigational tool for end users, enabling them to explore archives through the hierarchical structure of classes, alongside navigation by repository and collection.

Collection
Set of rules that a record must comply with. The collection is defined via the Web agent or Arcsys API, and comprises information contained in the relational database tables. A collection always refers to two rules: one concerning the **storage policy** and one relating to the **indexing mask**. A collection is assigned to the record on the initial record request.
See Also **Storage policy**, **Indexing mask**.

Deletion
MOREQ2010 provides the following definition for this concept: the act of deleting data from the relational database so that no trace remains. Generally speaking, an entity can only be deleted if is not used in a stored record. Otherwise, it can only be destroyed and not deleted, thus leaving a residual entity.
See Also **Destruction**.

Destruction
Irreversible action that deletes the documents by applying disposal criteria. It can be associated with the retention of residual information in the relational database.

Disposal
Outcome of archived documents when the retention period ends, i.e. generally, destruction or transfer.
See Also **Destruction**.

Disposal due date (or retention end date)
Scheduled end of retention date.

Disposal Hold

Arcsys can be used to place a "disposal hold" on one or more lots archived in the application. This prevents certain sensitive operations, such as transitioning the lots to end-of-life status or migrating them to a different storage medium. All other operations remain authorized. The disposal hold guarantees that no irreversible change affecting the archival integrity or status of the lot can occur while the hold is active.

Electronic Attestation

Document produced to attest that an action or an electronic transaction has occurred.

Envelope

Arcsys groups documents stored in the system in envelopes, either created by Arcsys during the archiving process (in this case, files in TAR format), or created prior to Arcsys processing by the user or third-party processes (*native envelopes* in AFP or ZIP format, for example). The representation of an envelope in the Arcsys Database is called a **logical envelope**. Its technical implementation is also called *MoverReference*. Last but not least, the representation of information of where the envelope is physically stored in the optional ArcMover module is called *MoverMedia*.

Event

In Arcsys, a retention schedule can associate the start of record retention with an event with a known or unknown date. This event, created in an Arcsys repository, can thus be attached to a number of different retention schedules.
See Also **Retention schedule**.

Feature preview

A Preview status on a feature enables early access to non-production features, allowing users to explore and provide feedback for improvement.

Features in Preview status should not be used in production environment, as they are not fully implemented yet.

Fixity

The quality of a document that has not been subject to intentional or accidental destruction, alteration or modification.

Format policy

A format policy is used to define a set of rules concerning format checks for a given file type. These rules are used to specify the action that will be performed, the expected results of these actions, as well as the error cases, triggering archiving failure.

Hash value

Also called an "integrity certificate" in cryptography, the hash value is the digest of a message which guarantees a practically unique result that is impossible to reverse calculate. The most commonly used algorithms are MD5 (128 bits), SHA-1 (160 bits), SHA256 (256 bits) and SHA512 (512 bits). Arcsys includes a module that is capable of dynamically calling

several algorithms. The choice of an algorithm type is valid for all archived objects within the same Arcsys product version; compatibility with algorithms from the previous version is guaranteed. The associated term *hash function* is also used.

Indexing mask

As is the case with the storage policy, an indexing mask is a rule that is referenced by a collection. An indexing mask can be referenced by several collections. An indexing mask refers to the use of a set of Keyword = Value pairs. The keyword component is set to make sense in a specific business application (e.g. Accounting Day, Department, Account No., Account Holder, etc.). The value component can be either unrestricted, or restricted to a set of acceptable values (e.g. A, B or C), or in date format, or restricted by an input mask. Some pairs are defined as mandatory whereas others may be optional.

An application which uses an indexing mask through a collection must supply all Keyword=Value pairs as they are defined using this mask. Any indexing-related errors lead to the record being rejected for conformity. This record is then added to the list of records with errors.

The indexing mask is defined by an administrator via the Arcsys interface or APIs. It is comprised of a set of keywords.
See Also **Keyword**.

Journal

A journal is an XML file which contains a list of PREMIS events.

Keyword

Component of an indexing mask. The keyword in particular defines the type of metadata (date, string, numerical, controlled) and its default value, for example.
See Also **Indexing mask**.

Lot

Arcsys can consolidate several different objects that form a functional set in a client application in the same physical record. It is comprised of different types of objects: files, databases, or any other type of object managed by Arcsys. It is possible to retrieve the entire lot or one of the objects contained in the lot. The MOREQ2010 record is translated in Arcsys implementation by a lot; the lot, as opposed to a MOREQ2010 record, can represent documents that are not yet archived.

Lot enrichment

The process of adding new objects to an existing archive.

Manifest

The manifest is an XML file that defines precisely the content of a record. The manifest contains: metadata associated with the record, structure metadata, a description of the physical files of the record(s) that follow, the object-by-object content of the record, object formats, object names, their size, hash value, the algorithm used to calculate the hash value, etc. The manifest is a type of complete ID card for the record.

The manifest is always written on the storage media and precedes the record that it describes. This process is used to automatically describe storage media (irrespective of the medium). With this system, users can understand media content and metadata without installing the software that generated the records.

Object
The object is a basic archived unit that can be retrieved via Arcsys. Lots contain one or more objects. An object can be: a file, a directory, a table, a relational table, etc. The MOREQ2010 component is implemented by this object concept; the object, as opposed to a MOREQ2010 component, can represent a document that has not yet been archived.

Online
Storage level, which must be disk type, that makes records permanently available within an extremely reduced time period.

Permissions
Permissions refer to the user profiles or groups authorized to access documents or sets of documents archived in the system.

Program exit
Place in the standard workflow for picking up and executing specific code.
See Also **Workflow**.

Proof folder
A proof folder consists of a record, a proof slip, and, where appropriate, additional items (common signature or timestamp response, for example) that are used, by demonstrating the fixity and the authenticity of a document, for admission as proof. A proof slip can be generated using Arcsys Web Agent, ArcWeb Module, or Arcsys REST API. A proof folder can only be generated using ArcEP.

Record
A record is an evidential document that is deemed sufficiently important by the creator to be managed by an ERM that will manage its life cycle (retention, disposal, etc.). A record represents an archived lot. A record is archived via a *record request*. Archiving a document *creates a record*.

Relational database (or referential)
Essential component of the system, it contains all the data (excluding archived data) used by Arcsys for its operation. It includes logical entities called "repositories" (see definition).

Repository
Logical entity used to segment and isolate information within Arcsys. Each repository has its own storage profiles, indexing masks, keywords, and configuration, all fully independent from those of other repositories.

A repository also defines its own permission scope.
See Also **Storage policy**, **Indexing mask**, **Keyword**, **Permissions**.

Restore (or retrieval)
This term is used misleadingly in Arcsys to refer to the concept of archive retrieval. It is not accepted in archiving terminology as to mean transfer and then destruction.
See Also **Archive Retrieval**.

Retention and disposal schedule
This comprises all the rules defining the record retention period for a company or an organization, according to risks of unavailability and information system access requirements. It specifies the disposal after these time periods.
See Also **Retention schedule**.

Retention period
A duration expressed in days, months or years of object retention. The retention period is a concept used notably in MOREQ2010.

Retention schedule
A retention schedule defines the start and the end of the retention of records that are attached to it, either directly or through their class.

Retention start date
Date from which a retention period must be taken into account. The retention start date is a concept used notably in MOREQ2010.

Slug
A tenant slug is a short, URL-friendly identifier used to represent a tenant in a stable and readable way within application URLs. It uniquely identifies the tenant using only lowercase letters, digits, and hyphens.
See Also **Tenant**.

Storage policy
A storage policy is a rule that is referenced by a collection. The policy dictates the storage media which are successively implemented to hold a record, as well as the retention period for each media. The storage policy is defined through the graphical interface. Applications or business users use it indirectly through the reference to a collection. A storage policy can be changed over time to reflect new retention periods or new storage media. The policy covers storage units by logical pool.

Storage pool
Logical storage pool, characterized in particular by its time period (e.g. 10 years). The storage policy assigns a "zone" to a "policy".

Storage zone
The storage zone is a logical entity representing a physical storage space (e.g. set of file systems, tape libraries, cloud storage).

Synchronous retrieval
Archive retrieval that takes place in the form of a direct retrieval of a document (for direct viewing or downloading) in a Web browser.

See Also **Archive Retrieval**.

Tenant

A tenant is a distinct logical environment within a shared Arcsys installation, fully isolated in terms of customization while running on a common technical platform. It enables each client or organization to benefit from a fully personalized interface and functional setup without impacting others.

Tenants do not isolate or partition the data stored in Arcsys.
See Also **Slug**.

Timestamping

Timestamping is a technique used to associate a document with a certain date in reference to a given and recognized time system. The date set in this way is an essential element for document authentication. Timestamping can be performed internally or by a third-party time stamp.

Tracking

Result of continuously creating, capturing and maintaining information about the movement and use of the system and objects (ISO 15489-1:2001, 3.19).

Transit Zone

Entity logically connecting an application agent and a directory, along with additional configuration.

Workflow

A set of operations carried out from the beginning to the end of a process. In Arcsys, this refers to all actions carried out on archives and objects, either directly or indirectly, in the case of archives, from their pre-archiving or preparation to their removal from the system (after they have reached end-of-life). There are other workflows in Arcsys besides the archiving workflow, which are more administration-oriented. Customized workflow involves the use of at least one drop-off point to carry out customer processing.

Registered Trademarks

Firefox is a registered trademark of the Mozilla Foundation.

Linux is a registered trademark of Linus Torvalds.

UNIX is a registered trademark of the Open Group.

Microsoft Windows, Windows NT, Windows Server, SQL Server, Internet Explorer are registered trademarks of Microsoft Corporation in the United States and/or other countries.

SAP is a registered trademark of SAP AG in Germany and other countries.

MySQL is a registered trademark of Oracle and/or its subsidiaries. MariaDB is a registered trademark of Monty Program Ab.

Java is a registered trademark of Oracle and/or its subsidiaries in the United States and other countries.

Infotel is a registered trademark of Infotel SA.

All trademarks mentioned are the property of their respective owners.

Infotel Technical Support

https://techsupport.infotel.com