Universal Data Mover

Reference Guide

Universal Products

Version 3.2.0

udm-ref-3207

Universal Data Mover

Reference Guide

Universal Products 3.2.0

Document Name	Universal Data Mover 3.2.0 Reference Guide				
Document ID	udm-ref-3207				
Products	**z/OS**	**UNIX**	**Windows**	**OS/400**	**HP NonStop**
Universal Data Mover Manager	√	√	√	√	
Universal Data Mover Server	√	√	√	√	

Stonebranch Documentation Policy

Stonebranch, Inc.
950 North Point Parkway, Suite 200
Alpharetta, GA 30005 USA
Tel: (678) 366-7887
Fax: (678) 366-7717

SAP Certified Integration

Summary of Changes

Changes for Universal Data Mover 3.2.0 Reference Guide (udm-ref-3207) February 19, 2010

- Added _execrc built-in variable in Table 7.7 Built-In Variables.

Changes for Universal Data Mover 3.2.0 Reference Guide (udm-ref-3206) September 8, 2009

Universal Data Mover 3.2.0.6

- Specified information about added support for the UTF-8 codepage in:
  - UDM Manager CODE_PAGE configuration option
  - UDM Server CODE_PAGE configuration option
- Added the UDM Server LOGON_METHOD configuration option.
- Added the following code pages in Section 7.12 Character Code Pages:
  - IBM875
  - IBM4971

Changes for Universal Data Mover 3.2.0 Reference Guide (udm-ref-3205) July 29, 2009

Universal Data Mover 3.2.0.1 for OS/400

- Modified document for upgrade from Universal Data Mover 3.1.1 for OS/400 to Universal Data Mover 3.2.0 for OS/400, including:
  - Changed the following OS/400 names throughout the document:
    - Universal Broker subsystem name from **UBROKER** to **UNVUBR320**.

    - Universal Broker user profile name from **UBROKER** to **UNVUBR320**.
    - Universal Products installation library name from **UNIVERSAL** to **UNVPRD320**.
    - Universal Products spool library name from **UNVSPOOL** to **UNVSPL320**.
    - Universal Products temporary directory from **UNVTMP** to **UNVTMP320**.
- Added the following OS/400 configuration option in Chapter 2 Universal Data Mover Manager Configuration Options:
    - CODEPAGE_TO_CCSID_MAP
- Specified the following configuration options for OS/400 in Chapter 2 Universal Data Mover Manager Configuration Options:
    - ACTIVITY_MONITORING
    - CA_CERTIFICATES
    - CERTIFICATE
    - CERTIFICATE_REVOCATION_LIST
    - COMMENT
    - EVENT_GENERATION
    - OPEN_RETRY
    - OPEN_RETRY_COUNT
    - OPEN_RETRY_INTERVAL
    - PLF_DIRECTORY
    - PRIVATE_KEY
    - PRIVATE_KEY_PWD
    - PROXY_CERTIFICATES
- Added a STRUDM parameter to the following configuration options for OS/400 in Chapter 2 Universal Data Mover Manager Configuration Options:
    - CTL_SSL_CIPHER_LIST
    - DATA_SSL_CIPHER_LIST
    - FRAME_INTERVAL
    - MODE_TYPE
    - OUTBOUND_IP
- Added character translation information for OS/400 to the following configuration option in Chapter 2 Universal Data Mover Manager Configuration Options:
    - PRIVATE_KEY_PWD
- Added the following OS/400 configuration option in Chapter 3 Universal Data Mover Server Configuration Options.
    - CODEPAGE_TO_CCSID_MAP
- Specified the following configuration options for OS/400 in Chapter 3 Universal Data Mover Server Configuration Options:
    - ACTIVITY_MONITORING
    - EVENT_GENERATION
    - TMP_DIRECTORY
- Added the following Universal Access Control List entry for OS/400 in Chapter 5 Universal Data Mover UACL Entries:
    - UDM_MGR_ACCESS

Changes for Universal Data Mover 3.2.0 Reference Guide (udm-ref-3204) April 1, 2009

- Added an example of the upper command.

Universal Data Mover 3.2.0.3

- Added the TCP_NO_DELAY configuration option in Chapter 2 Universal Data Mover Manager Configuration Options.
- Added the TCP_NO_DELAY configuration option in Chapter 3 Universal Data Mover Server Configuration Options .
- Added the following commands in Chapter 6 UDM Commands:
  - appenddata
  - closelog
  - echolog
  - logdata
  - move
  - openlog
  - savedata
- Added the following parameters to the exec command in Chapter 6 UDM Commands:
  - stdout
  - stderr
- Added the following attributes in Table 6.4 Common File System Attributes:
  - srccreatetime
  - srcmodtime
  - srcaccesstime
- Added the following variables in Section 7.6 Built-In Variables:
  - _uuid
  - _lastmsg
- Added Section 7.7 _file Built-in Variable – Special Attributes.

Changes for Universal Data Mover 3.2.0 Reference Guide (udm-ref-3203) December 17, 2008

- Specified, in the following tables, that the **trans** attribute of the attrib command is valid only under the **hfs** file system for z/OS and OS/400:
  - Table 6.4 Common File System Attributes
  - Table 7.2 Common File System Attributes
- Changed the name of the environment variable for the Universal Data Mover Manager SYSTEM_ID configuration option from **UDMSYSTEM** to **UDMSYSTEMID**.

Changes for Universal Data Mover 3.2.0 Reference Guide (udm-ref-3202) October 17, 2008

- Added a note about incorrect character translations for the Universal Data Mover Manager for OS/400 PRIVATE_KEY_PWD option.
- Changed **JCL SNTYPE** value to **type** for the **dsntype** attribute in Table 7.3 z/OS attrib Command - Dynamic Allocation Attributes.

Changes for Universal Data Mover 3.2.0 Reference Guide (udm-ref-3201) September 5, 2008

- Added toll-free telephone number for North America in Appendix A Customer Support.

Changes for Universal Data Mover 3.2.0 Reference Guide (udm-ref-320) May 16, 2008

Universal Data Mover 3.2.0.3

- Consolidated UDM Manager options and UDM Manager Invocation options into a single chapter, Chapter 2 Universal Data Mover Manager Configuration Options.
- Added the following configuration options in Chapter 2 Universal Data Mover Manager Configuration Options:
  - ACTIVITY_MONITORING
  - BIF_DIRECTORY
  - CA_CERTIFICATES
  - CERTIFICATE
  - CERTIFICATE_REVOCATION_LIST
  - COMMENT
  - EVENT_GENERATION
  - OPEN_RETRY_COUNT
  - OPEN_RETRY_INTERVAL
  - PLF_DIRECTORY
  - PRIVATE_KEY
  - PRIVATE_KEY_PWD
  - PROXY_CERTIFICATES
  - SAF_KEY_RING
  - SAF_KEY_RING_LABEL
  - SERVER_STOP_CONDITIONS
  - SSL_IMPLEMENTATION
  - SYSTEM_ID

- Added the following configuration options in Chapter 3 Universal Data Mover Server Configuration Options.
  - ACTIVITY_MONITORING
  - EVENT_GENERATION
  - TMP_DIRECTORY
- Added Chapter 4 Universal Data Mover Component Definition Options.
- Added Chapter 5 Universal Data Mover UACL Entries.
- Added 17 new commands in Chapter 6 UDM Commands.
- Modified the following commands in Chapter 6 UDM Commands:
  - attrib: Added the **`mode`** attribute to list of Common File System Attributes.
  - open: Added **`comment`** parameter.
  - set.
- Removed the following specification methods for all UDM Server configuration options:
  - Command Line, Short Form
  - Command Line, Long Form
  - Environment Variable
- Added Configuration File Keyword as a specification method for Windows options.

Contents

Summary of Changes **5**

Contents **10**

List of Tables **17**

Preface **19**

Document Structure 19
Format 19
Conventions 20
Vendor References 21

Document Organization 22

Chapter 1 Overview **23**

Chapter 2 Universal Data Mover Manager Configuration Options **24**

2.1 Overview 24
2.2 Configuration Options Information 25
2.3 Configuration Options List 27
2.4 ACTIVITY_MONITORING 30
2.5 ALLOC_ABNORMAL_DISP 31
2.6 ALLOC_BLKSIZE 32
2.7 ALLOC_DATACLAS 33
2.8 ALLOC_DIR_BLOCKS 34
2.9 ALLOC_DSORG 35

2.10 ALLOC_INPUT_STATUS 36
2.11 ALLOC_LRECL 37
2.12 ALLOC_MGMTCLAS 38
2.13 ALLOC_NORMAL_DISP 39
2.14 ALLOC_OUTPUT_STATUS 40
2.15 ALLOC_PRIM_SPACE 41
2.16 ALLOC_RECFM 42
2.17 ALLOC_SEC_SPACE 43
2.18 ALLOC_SPACE_UNIT 44
2.19 ALLOC_STORCLAS 45
2.20 ALLOC_UNIT 46
2.21 ALLOC_VOLSER 47
2.22 BIF_DIRECTORY 48
2.23 CA_CERTIFICATES 49
2.24 CERTIFICATE 50
2.25 CERTIFICATE_REVOCATION_LIST 51
2.26 CODE_PAGE 52
2.27 CODEPAGE_TO_CCSID_MAP 54
2.28 COMMENT 56
2.29 CTL_SSL_CIPHER_LIST 57
2.30 DATA_COMPRESSION 58
2.31 DATA_SSL_CIPHER_LIST 60
2.32 EVENT_GENERATION 61
2.33 FRAME_INTERVAL 63
2.34 HELP 64
2.35 IDLE_TIMEOUT 65
2.36 INSTALLATION_DIRECTORY 66
2.37 KEEP_ALIVE_INTERVAL 67
2.38 MERGE_LOG 68
2.39 MESSAGE_LANGUAGE 69
2.40 MESSAGE_LEVEL 70
2.41 MODE_TYPE 72
2.42 NETWORK_DELAY 73
2.43 NETWORK_FAULT_TOLERANT 74

2.44 NLS_DIRECTORY 75
2.45 OPEN_RETRY 76
2.46 OPEN_RETRY_COUNT 78
2.47 OPEN_RETRY_INTERVAL 79
2.48 OUTBOUND_IP 80
2.49 PLF_DIRECTORY 81
2.50 PRIVATE_KEY 82
2.51 PRIVATE_KEY_PWD 83
2.52 PROXY_CERTIFICATES 84
2.53 RECONNECT_RETRY_COUNT 85
2.54 RECONNECT_RETRY_INTERVAL 86
2.55 RECV_BUFFER_SIZE 87
2.56 REMOTE_PORT 88
2.57 SAF_KEY_RING 89
2.58 SAF_KEY_RING_LABEL 90
2.59 SCRIPT 91
2.60 SCRIPT_FILE 92
2.61 SCRIPT_OPTIONS 93
2.62 SEND_BUFFER_SIZE 94
2.63 SERVER_STOP_CONDITIONS 95
2.64 SIZE_ATTRIB 96
2.65 SSL_IMPLEMENTATION 97
2.66 SYSTEM_ID 98
2.67 TCP_NO_DELAY 99
2.68 TRACE_FILE_LINES 100
2.69 TRACE_TABLE 102
2.70 UCMD_PATH 104
2.71 UMASK 105
2.72 USAP_PATH 106
2.73 VERSION 107

Chapter 3 Universal Data Mover Server Configuration Options 108
3.1 Overview 108
3.2 Configuration Options Information 109

3.3 Configuration Options List 111
3.4 ACTIVITY_MONITORING 113
3.5 ALLOC_ABNORMAL_DISP 114
3.6 ALLOC_BLKSIZE 115
3.7 ALLOC_DATACLAS 116
3.8 ALLOC_DIR_BLOCKS 117
3.9 ALLOC_DSORG 118
3.10 ALLOC_INPUT_STATUS 119
3.11 ALLOC_LRECL 120
3.12 ALLOC_MGMTCLAS 121
3.13 ALLOC_NORMAL_DISP 122
3.14 ALLOC_OUTPUT_STATUS 123
3.15 ALLOC_PRIM_SPACE 124
3.16 ALLOC_RECFM 125
3.17 ALLOC_SEC_SPACE 126
3.18 ALLOC_SPACE_UNIT 127
3.19 ALLOC_STORCLAS 128
3.20 ALLOC_UNIT 129
3.21 ALLOC_VOLSER 130
3.22 CODE_PAGE 131
3.23 CODEPAGE_TO_CCSID_MAP 133
3.24 DATA_COMPRESSION 135
3.25 DATA_SSL_CIPHER_LIST 136
3.26 EVENT_GENERATION 137
3.27 FRAME_INTERVAL 139
3.28 INSTALLATION_DIRECTORY 140
3.29 LOGON_METHOD 141
3.30 MESSAGE_LEVEL 142
3.31 NETWORK_DELAY 144
3.32 NLS_DIRECTORY 145
3.33 OUTBOUND_IP 146
3.34 RECONNECT_RETRY_COUNT 147
3.35 RECONNECT_RETRY_INTERVAL 148
3.36 RECV_BUFFER_SIZE 149

3.37 SEND_BUFFER_SIZE 150
3.38 SIZE_ATTRIB 151
3.39 TCP_NO_DELAY 152
3.40 TMP_DIRECTORY 153
3.41 TRACE_DIRECTORY 154
3.42 TRACE_FILE_LINES 155
3.43 TRACE_TABLE 157
3.44 UMASK 158
3.45 USER_SECURITY 159

Chapter 4 Universal Data Mover Component Definition Options 160
4.1 Overview 160
4.2 Component Definition Information 161
4.3 Component Definition Options List 163
4.4 AUTOMATICALLY_START 164
4.5 COMPONENT_NAME 165
4.6 CONFIGURATION_FILE 166
4.7 RUNNING_MAXIMUM 167
4.8 START_COMMAND 168
4.9 WORKING_DIRECTORY 169

Chapter 5 Universal Data Mover UACL Entries. 170
5.1 Overview 170
5.2 UACL Entries Information 171
5.3 UACL Entries List 173
5.4 UDM_ACCESS 174
5.5 UDM_MGR_ACCESS 176

Chapter 6 UDM Commands 178
6.1 Overview 178
6.1.1 UDM Commands List 178
6.2 appenddata 181
6.3 attrib 182
6.4 break 187
6.5 call 188

6.6 cd 190
6.7 close 191
6.8 closelog 192
6.9 compare 193
6.10 copy 194
6.11 copydir 196
6.12 data 198
6.13 debug 199
6.14 delete 200
6.15 deletestring 201
6.16 echo 203
6.17 echolog 204
6.18 exec 206
6.19 execsap 210
6.20 exit 213
6.21 filesys 214
6.22 filetype 216
6.23 find 218
6.24 format 220
6.25 insertstring 222
6.26 loaddata 224
6.27 logdata 225
6.28 lower 226
6.29 mode 227
6.30 move 228
6.31 open 230
6.32 openlog 235
6.33 pad 237
6.34 parse 239
6.35 print 241
6.36 query 242
6.37 quit 243
6.38 rename 244
6.39 replace 245

6.40 report 247
6.41 resetattribs 248
6.42 return 249
6.43 reverse 250
6.44 savedata 251
6.45 set 253
6.46 status 254
6.47 strip 255
6.48 substring 257
6.49 truncate 259
6.50 upper 260

Chapter 7 Additional Information 261
7.1 Overview 261
7.2 Common File System Attributes 262
7.3 z/OS Dynamic Allocation Attributes 265
7.4 z/OS Attributes for Allocating Temporary Data Sets when Copying Load Modules 268
7.5 OS/400-Specific File System Attributes 269
7.5.1 LIB File System Attributes 269
7.5.2 HFS Attributes 271
7.6 Built-In Variables 272
7.7 _file Built-in Variable – Special Attributes 273
7.8 Global Variable Attributes 274
7.9 UDM Statements 276
7.10 if Statement Comparators 278
7.11 UDM Command Expression Operators 279
7.12 Character Code Pages 280
7.13 UTT Files 282
7.14 SSL Cipher Suites 283
7.15 DD Statements 284

Appendix A Customer Support 285

List of Tables

Chapter 2 Universal Data Mover Manager Configuration Options 24

Table 2.1 UDM Manager Configuration Options 29

Chapter 3 Universal Data Mover Server Configuration Options 108

Table 3.1 UDM Server Configuration Options 112
Table 3.2 UTT File Locations 132

Chapter 4 Universal Data Mover Component Definition Options 160

Table 4.1 Universal Data Mover Component Definition Options 163

Chapter 5 Universal Data Mover UACL Entries. 170

Table 5.1 Universal Data Mover UACL Entries 173

Chapter 6 UDM Commands 178

Table 6.1 UDM Commands 180
Table 6.2 appenddata Command Parameters 181
Table 6.3 attrib Command Parameters 182
Table 6.4 Common File System Attributes 186
Table 6.5 call Command Parameters 188
Table 6.6 cd Command Parameters 190
Table 6.7 compare Command Parameters 193
Table 6.8 copy Command Parameters 195
Table 6.9 copydir Command Parameters 197
Table 6.10 data Command Parameters 198
Table 6.11 debug Command Parameters 199
Table 6.12 delete Command Parameters 200
Table 6.13 deletestring Command Parameters 202
Table 6.14 echo Command Parameters 203
Table 6.15 echolog Command Parameters 204
Table 6.16 exec Command Parameters 209
Table 6.17 execsap Command Parameters 212

Table 6.18 filesys Command Parameters 215
Table 6.19 filetype Command Parameters 216
Table 6.20 find Command Parameters 219
Table 6.21 format Command Parameters 220
Table 6.22 insertstring Command Parameters 223
Table 6.23 loaddata Command Parameters 224
Table 6.24 logdata Command Parameters 225
Table 6.25 lower Command Parameters 226
Table 6.26 mode Command Parameters 227
Table 6.27 move Command Parameters 229
Table 6.28 open Command Parameters 233
Table 6.29 `file` / `xfile` Format 234
Table 6.30 openlog Command Parameters 236
Table 6.31 pad Command Parameters 237
Table 6.32 parse Command Parameters 239
Table 6.33 print Command Parameter 241
Table 6.34 rename Command Parameter 244
Table 6.35 replace Command Parameters 246
Table 6.36 report Command Parameters 247
Table 6.37 resetattribs Command Parameters 248
Table 6.38 return Command Parameters 249
Table 6.39 reverse Command Parameters 250
Table 6.40 savedata Command Parameters 251
Table 6.41 set Command Parameters 253
Table 6.42 strip Command Parameters 255
Table 6.43 substring Command Parameters 258
Table 6.44 truncate Command Parameters 259
Table 6.45 upper Command Parameters 260

Chapter 7 Additional Information 261

Table 7.1 Universal Data Mover - Additional Information 261
Table 7.2 Common File System Attributes 264
Table 7.3 z/OS attrib Command - Dynamic Allocation Attributes 267
Table 7.4 z/OS Attributes for Allocating Temporary Data Sets when Copying Load Modules 268
Table 7.5 OS/400-Specific LIB File Attributes for Creating New Files 271
Table 7.6 OS/400-Specific HFS File Attributes for Creating New Files 271
Table 7.7 Built-In Variables 272
Table 7.8 _file Built-in Variable – Special Attributes 273
Table 7.9 UDM Global Variable Attributes 275
Table 7.10 UDM Statements 277
Table 7.11 if Statement Comparators 278
Table 7.12 UDM Command Expressions - Operators 279
Table 7.13 Character Code Pages 281
Table 7.14 UTT File Locations 282
Table 7.15 SSL Cipher Suites for UDM 283
Table 7.16 Universal Data Mover Batch JCL DD Statements 284

Preface

Document Structure

This document is written using specific conventions for text formatting and according to a specific document structure in order to make it as useful as possible for the largest audience. The following sections describe the document formatting conventions and organization.

Format

Starting with the Universal Products 3.2.0 release, this Universal Data Mover Reference Guide serves as a companion document to the Universal Data Mover User Guide.

Links to detailed information this reference guide have been created in the user guide.

In order for the links between these documents to work correctly:

- Place the documents in the same folder.
- In Adobe Reader / Adobe Acrobat, de-select **Open cross-document link in same window** in the **General** category of your **Preferences** dialog (selected from the **Edit** menu).

Conventions

Specific text formatting conventions are used within this document to represent different information. The following conventions are used.

Typeface and Fonts

This document provides tables that identify how information is used. These tables identify values and/or rules that are either pre-defined or user-defined:

- *Italics* denotes user-supplied information.
- **Boldface** indicates pre-defined information.

Elsewhere in this document, **`This Font`** identifies specific names of different types of information, such as file names or directories (for example, **`\abc\123\help.txt`**).

Operating System-Specific Text

Most of this document describes the product in the context of all supported operating systems. At times, it is necessary to refer to operating system-specific information. This information is introduced with a special header, which is followed by the operating system-specific text in a different font size from the normal text.

z/OS

This text pertains specifically to the z/OS line of operating systems.

This text resumes the information pertaining to all operating systems.

Tips from the Stoneman

Stoneman's Tip

Look to the Stoneman for suggestions
or for any other information
that requires special attention.

Vendor References

References are made throughout this document to a variety of vendor operating systems. We attempt to use the most current product names when referencing vendor software.

The following names are used within this document:

- **z/OS** is synonymous with IBM z/OS and IBM OS/390 line of operating systems.
- **Windows** is synonymous with Microsoft's Windows 2000 / 2003 / 2008, Windows XP, Windows Vista, and Windows 7 lines of operating systems. Any differences between the different systems will be noted.
- **UNIX** is synonymous with operating systems based on AT&T and BSD origins and the Linux operating system.
- **OS/400** is synonymous with IBM OS/400, IBM i/5, and IBM i operating systems.
- **AS/400** is synonymous for IBM AS/400, IBM iSeries, and IBM System i systems.

Note: These names do not imply software support in any manner. For a detailed list of supported operating systems, see the Universal Products 3.2.0 Installation Guide.

Document Organization

The document is organized into the following chapters:

- Overview (Chapter 1)
  Overview of the information provided in this document.
- Universal Data Mover Manager Configuration Options (Chapter 2)
  Detailed information about the configuration options used with the Universal Data Mover Manager.
- Universal Data Mover Server Configuration Options (Chapter 3)
  Detailed information about the configuration options used with the Universal Data Mover Server.
- Universal Data Mover Component Definition Options (Chapter 4)
  Detailed information about the component definition options used with the Universal Data Mover.
- Universal Data Mover UACL Entries (Chapter 5)
  Detailed information about the Universal Access Control List (UACL) entries available for use with the Universal Data Mover.
- UDM Commands (Chapter 6)
  Detailed information about the Universal Data Mover commands available for use with Universal Data Mover.
- Additional Information (Chapter 7)
  Additional technical information relative to Universal Data Mover.
- Customer Support (Appendix A)
  Customer support contact information for users of Universal Data Mover

Chapter 1
Overview

The Universal Data Mover 3.2.0 Reference Guide is a companion document to the Universal Data Mover 3.2.0 User Guide.

It provides technical detail for the information and procedures presented in that document:

- Universal Data Mover Manager options
- Universal Data Mover Server options
- Universal Data Mover component definition options
- Universal Data Mover UACL entries
- Universal Data Mover commands
- Additional information

Chapter 2
Universal Data Mover Manager Configuration Options

2.1 Overview

This chapter provides detailed information on the configuration options available for use with the Universal Data Mover Manager.

The options are listed alphabetically, without regard to any specific operating system.

Information on how these options are used is documented in the Universal Data Mover 3.2 User Guide.

Section 2.2 Configuration Options Information provides a guideline for understanding the information presented or each option.

2.2 Configuration Options Information

For each configuration option, this chapter provides the following information.

Description

Describes the configuration option and how it is used.

Usage

Provides a table of the following information:

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	<Format / Value>					
Command Line Option, Long Form	<Format / Value>					
Environment Variable	<Format / Value>					
Configuration File Keyword	<Format / Value>					
STRUDM Parameter	<Format / Value>					

Method

Identifies the different methods used to specify Universal Data Mover Manager configuration options:

- Command Line Option, Short Form
- Command Line Option, Long Form
- Environment Variable
- Configuration File Keyword
- STRUDM Parameter

Note: Each option can be specified using one or more methods.

Syntax

Identifies the syntax of each method that can be used to specify the option:

- Format Specific characters that identify the option.
- Value Type of value(s) to be supplied for this method.

Note: If a Method is not valid for specifying the option, the Syntax field contains **n/a**.

(Operating System)

Identifies (with a ✔) the operating systems for which each method of specifying the option is valid:

- OS/400
- HP NonStop
- UNIX
- Windows
- z/OS

Values

Identifies all possible values for the specified value type.

Defaults are identified in **[bracketed bold type]**.

<Additional Information>

Identifies any additional information specific to the option.

2.3 Configuration Options List

Table 2.1, below, identifies all UDM Manager configuration options.

Option	Description	Page
ACTIVITY_MONITORING	Specification for whether or not product activity monitoring events are generated.	30
ALLOC_ABNORMAL_DISP	Disposition of data set when an abnormal ending occurs.	31
ALLOC_BLKSIZE	Block size used for newly allocated data sets.	32
ALLOC_DATACLAS	SMS data class used for newly allocated data sets.	33
ALLOC_DIR_BLOCKS	Number of directory blocks for newly allocated partitioned data sets.	34
ALLOC_DSORG	Data set organization used for newly allocated data sets.	35
ALLOC_INPUT_STATUS	Status of data sets being allocated for input.	36
ALLOC_LRECL	Logical record length used for newly allocated data sets.	37
ALLOC_MGMTCLAS	SMS management class used for newly allocated data sets.	38
ALLOC_NORMAL_DISP	Disposition of data set when normal ending occurs.	39
ALLOC_OUTPUT_STATUS	Status of existing data sets being allocated for output.	40
ALLOC_PRIM_SPACE	Primary space allocation used for newly allocated data sets.	41
ALLOC_RECFM	Record format used for newly allocated data sets.	42
ALLOC_SEC_SPACE	Secondary space allocation used for newly allocated data sets.	43
ALLOC_SPACE_UNIT	Space unit in which space is allocated for newly allocated data sets.	44
ALLOC_STORCLAS	SMS storage class used for newly allocated data sets.	45
ALLOC_UNIT	Unit used for newly allocated data sets.	46
ALLOC_VOLSER	Volume serial number used for newly allocated data sets.	47
BIF_DIRECTORY	Broker Interface Directory that specifies the location of the Universal Broker interface file.	48
CA_CERTIFICATES	File name / ddname of the PEM-formatted trusted CA X.509 certificates.	49
CERTIFICATE	File name / ddname of UDM Manager's PEM-formatted X.509 certificate.	50
CERTIFICATE_REVOCATION_LIST	File name / ddname of the PEM-formatted CRL.	51
CODE_PAGE	Character code page used to translate text data.	52
CODEPAGE_TO_CCSID_MAP	Specification to use the internal or external table for code page to CCSID mapping.	54
COMMENT	User-defined string.	56
CTL_SSL_CIPHER_LIST	Acceptable and preferred SSL cipher suites to use for the control session between UDM components.	57

Option	Description	Page
DATA_COMPRESSION	Specification for whether or not data is compressed on all standard I/O files.	58
DATA_SSL_CIPHER_LIST	SSL cipher suites to use for data session between UDM primary and secondary servers.	60
EVENT_GENERATION	Events to be generated as persistent events.	61
FRAME_INTERVAL	Number of UDM transfer blocks transferred before a frame-sync message is sent with network fault tolerance turned on.	63
HELP	Prints a description of the command options and their format.	64
IDLE_TIMEOUT	Number of seconds of inactivity in an interactive UDM session after which the UDM Manager will close the session.	65
INSTALLATION_DIRECTORY	Directory in which UDM Manager is installed.	66
KEEP_ALIVE_INTERVAL	Default interval at which a keep alive message is sent from the UDM Manager to the transfer server(s).	67
MERGE_LOG	Specification for merging standard out and standard error output streams from a remote command to the UDM transaction log.	68
MESSAGE_LANGUAGE	Universal Message Catalog (UMC) file used to format messages.	69
MESSAGE_LEVEL	Level of messages that UDM will write.	70
MODE_TYPE	Default transfer mode type for UDM sessions.	72
NETWORK_DELAY	Expected network latency (in seconds).	73
NETWORK_FAULT_TOLERANT	Setting for whether or not UDM transfer sessions are network fault tolerant by default.	74
NLS_DIRECTORY	Directory name where the UDM Manager can find its message catalog and code page tables.	75
OPEN_RETRY	Level of fault tolerance for the open command.	76
OPEN_RETRY _COUNT	Maximum number of attempts that will be made to establish a session by the open command.	78
OPEN_RETRY_INTERVAL	Number of seconds that UDM will wait between each open retry attempt.	79
OUTBOUND_IP	Host or IP address that UDM binds to when initiating outgoing connections to another UDM server.	80
PLF_DIRECTORY	Program Lock File directory that specifies the location of the UDM Manager program lock file.	81
PRIVATE_KEY	ddname of Manager’s PEM formatted RSA private key.	82
PRIVATE_KEY_PWD	Password for the Manager’s PRIVATE_KEY.	83
PROXY_CERTIFICATES	Specification for whether or not UDM will use proxy certificates in three-party transfer sessions if a certificate is supplied to the UDM Manager.	83
RECONNECT_RETRY_COUNT	Number of attempts that the UDM Manager will make to re-establish a transfer session when a network fault occurs.	85

Option	Description	Page
RECONNECT_RETRY_INTERVAL	Number of seconds that UDM will wait between each successive attempt to re-establish a transfer session when a network fault occurs.	86
RECV_BUFFER_SIZE	Size of the TCP receive buffer for UDM.	87
REMOTE_PORT	TCP port number on the remote computer used for invoking UDM Server instances.	88
SAF_KEY_RING	SAF certificate key ring name.	89
SAF_KEY_RING_LABEL	SAF key ring certificate label.	90
SCRIPT	ddname from which to read a UDM script command file.	91
SCRIPT_FILE	Script file containing UDM commands to execute.	92
SCRIPT_OPTIONS	Script options to pass into the script command file.	93
SEND_BUFFER_SIZE	Size of the TCP send buffer for UDM.	94
SERVER_STOP_CONDITIONS	Exit codes that cause Universal Broker to cancel the corresponding UDM Server of the exited UDM Manager.	95
SIZE_ATTRIB	Default size for file creation of physical files for both data and source file types.	96
SSL_IMPLEMENTATION	SSL implementation.	97
SYSTEM_ID	Local Universal Broker with which the UDM Manager must register.	98
TCP_NO_DELAY	Specification for whether or not to use TCP packet coalescing.	99
TRACE_FILE_LINES	Maximum number of lines to write to the trace file.	100
TRACE_TABLE	Size of a wrap-around trace table maintained in memory.	102
UCMD_PATH	Complete path to Universal Command for calls by the **exec** command.	104
UMASK	File mode creation mask.	105
USAP_PATH	Complete path to USAP for calls by the **execsap** command.	106
VERSION	Specification for writing of program version information and copyright.	107

Table 2.1 UDM Manager Configuration Options

2.4 ACTIVITY_MONITORING

Description

The ACTIVITY_MONITORING option specifies whether or not product activity monitoring events are generated.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	n/a					
Environment Variable	n/a					
Configuration File Keyword	activity_monitoring *option*	√		√	√	√
STRUDM Parameter	n/a					

Values

option is the specification for whether or not product activity monitoring events are generated.

Valid values for *option* are:

- **yes**
  Activate product activity monitoring events
- **no**
  Deactivate product activity monitoring events

[Default is yes.]

2.5 ALLOC_ABNORMAL_DISP

Description

The ALLOC_ABNORMAL_DISP option is a dynamic allocation option that specifies the disposition of data set when an abnormal ending occurs.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_abnormal_disp *disposition*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_abnormal_disp *disposition*					√
STRUDM Parameter	n/a					

Values

disposition is equivalent to the third positional parameter of the JCL DD statement's DISP parameter.

Valid values for *disposition* are:

- **keep**
  Keep the data set.
- **delete**
  Delete the data set.
- **catlg**
  Catalog the data set.
- **uncatlg**
  Un-catalog the data set.

[Default is delete.]

References

Refer to the IBM JCL Reference manual for complete details.

2.6 ALLOC_BLKSIZE

Description

The ALLOC_BLKSIZE option is a dynamic allocation option that specifies the block size used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_blksize *size*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_blksize *size*					√
STRUDM Parameter	n/a					

Values

size is equivalent to the JCL DD statement's BLKSIZE parameter.

Valid values for *size* are any number (size of a block).

[Default is *27998*.]

References

Refer to the IBM JCL Reference manual for complete details.

2.7 ALLOC_DATACLAS

Description

The ALLOC_BLKSIZE option is a dynamic allocation option that specifies the SMS data class used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_dataclas *class*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_dataclas *class*					√
STRUDM Parameter	n/a					

Values

class is equivalent to the JCL DD statement's DATACLAS parameter.

Valid values for *class* are any SMS data classes defined in the local environment.

[There is no default.]

References

Refer to the IBM JCL Reference manual for complete details.

2.8 ALLOC_DIR_BLOCKS

Description

The ALLOC_DIR_BLOCKS option is a dynamic allocation option that specifies the number of directory blocks for newly allocated partitioned data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_dir_blocks *number*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_dir_blocks *number*					√
STRUDM Parameter	n/a					

Values

number is equivalent to the third positional parameter of the second positional parameter of the JCL DD statement's SPACE parameter.

Valid values for *number* are any number (number of directory blocks to allocate).

[Default is *20*.]

References

Refer to the IBM JCL Reference manual for complete details.

2.9 ALLOC_DSORG

Description

The ALLOC_DSORG option is a dynamic allocation option that specifies the data set organization used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_dsorg *organization*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_dsorg *organization*					√
STRUDM Parameter	n/a					

Values

organization is equivalent to the JCL DD statement's DSORG parameter.

Valid values for *organization* are:

- **po**
  Partitioned organization
- **ps**
  Physically sequential

[Default is ps.]

References

Refer to the IBM JCL Reference manual for complete details.

2.10 ALLOC_INPUT_STATUS

Description

The ALLOC_INPUT_STATUS option is a dynamic allocation option that specifies the status of data sets being allocated for input.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_input_status *status*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_input_status *status*					√
STRUDM Parameter	n/a					

Values

status is equivalent to the first positional parameter of the JCL DD statement's DISP parameter.

Valid values for *status* are:

- **old**
  Allocate the data set exclusively.
- **shr**
  Allocate the data set non-exclusively.

[Default is old.]

References

Refer to the IBM JCL Reference manual for complete details.

2.11 ALLOC_LRECL

Description

The ALLOC_LRECL option is a dynamic allocation option that specifies the logical record length used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_lrecl *length*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_lrecl *length*					√
STRUDM Parameter	n/a					

Values

length is equivalent to the first positional parameter of the JCL DD statement's LRECL parameter.

Valid values for *length* are any number (length of the record).

[Default is *1024*.]

References

Refer to the IBM JCL Reference manual for complete details.

2.12 ALLOC_MGMTCLAS

Description

The ALLOC_MGMTCLAS option is a dynamic allocation option that specifies the SMS management class used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_mgmtclas *class*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_mgmtclas *class*					√
STRUDM Parameter	n/a					

Values

class is equivalent to the first positional parameter of the JCL DD statement's MGMTCLAS parameter.

Valid values for *class* are any SMS management classes defined in the local environment.

[There is no default.]

References

Refer to the IBM JCL Reference manual for complete details.

2.13 ALLOC_NORMAL_DISP

Description

The ALLOC_NORMAL_DISP option is a dynamic allocation option that specifies the disposition of data set when normal ending occurs.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_normal_disp *disposition*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_normal_disp *disposition*					√
STRUDM Parameter	n/a					

Values

disposition is equivalent to the second positional parameter of the JCL DD statement's DISP parameter.

Valid values for *disposition* are:

- **keep**
  Keep the data set.
- **delete**
  Delete the data set.
- **catlg**
  Catalog the data set.
- **uncatlg**
  Un-catalog the data set.

[Default is catlg.]

References

Refer to the IBM JCL Reference manual for complete details.

2.14 ALLOC_OUTPUT_STATUS

Description

The ALLOC_OUTPUT_STATUS option is a dynamic allocation option that specifies the status of data sets being allocated for output.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_output_status *status*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_output_status *status*					√
STRUDM Parameter	n/a					

Values

status is equivalent to the first positional parameter of the JCL DD statement's DISP parameter.

Valid values for *status* are:

- **new**
  Create new data set.
- **shr**
  Allocate the data set non-exclusively.
- **old**
  Allocate the data set exclusively.
- **mod**
  Either create a new data set, for exclusive use, or allocate a sequential data set exclusively and add records to the end of it.

[Default is old.]

References

Refer to the IBM JCL Reference manual for complete details.

2.15 ALLOC_PRIM_SPACE

Description

The ALLOC_PRIM_SPACE option is a dynamic allocation option that specifies the primary space allocation used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_prim_space *space*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_prim_space *space*					√
STRUDM Parameter	n/a					

Values

space is equivalent to the first sub-parameter of the second sub-parameter of the JCL DD statement's SPACE parameter.

Valid values for space are any number (number of space units to allocate).

[Default is 15.]

References

Refer to the IBM JCL Reference manual for complete details.

2.16 ALLOC_RECFM

Description

The ALLOC_RECFM option is a dynamic allocation option that specifies the record format used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_recfm *format*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_recfm *format*					√
STRUDM Parameter	n/a					

Values

format is equivalent to the JCL DD statement's **RECFM** parameter.

Valid values for *format* are dependent on the data set organization and access method used. The following values are valid for both partitioned and sequential data sets:

- **F**[**B**][**A**|**M**]
  Fixed, optionally blocked, and optionally either ANSI or Machine control characters.
- **V**[**B**][**A**|**M**|**S**]
  Variable, optionally blocked, and optionally either ANSI or Machine control characters, or spanned.

[Default is VB.]

References

Refer to the IBM JCL Reference manual for complete details.

2.17 ALLOC_SEC_SPACE

Description

The ALLOC_SEC_SPACE option is a dynamic allocation option that specifies the secondary space allocation used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_sec_space *space*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_sec_space *space*					√
STRUDM Parameter	n/a					

Values

space is equivalent to the second sub-parameter of the second sub-parameter of the JCL DD statement's SPACE parameter.

Valid values for *space* are any number (number of space units to allocate).

[Default is 15.]

References

Refer to the IBM JCL Reference manual for complete details.

2.18 ALLOC_SPACE_UNIT

Description

The ALLOC_SPACE_UNIT option is a dynamic allocation option that specifies the space unit in which space is allocated for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_space_unit *space*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_space_unit *space*					√
STRUDM Parameter	n/a					

Values

space is equivalent to the first sub-parameter of the JCL DD statement's SPACE parameter.

Valid values for *space* are:

- *number*
  Block length or record length
- **cyl**
  Cylinder allocation
- **trk**
  Track allocation

[Default is trk.]

References

Refer to the IBM JCL Reference manual for complete details.

2.19 ALLOC_STORCLAS

Description

The ALLOC_STORCLAS option is a dynamic allocation option that specifies the SMS storage class used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_storclas *class*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_storclas *class*					√
STRUDM Parameter	n/a					

Values

class is equivalent to the JCL DD statement's STORCLAS parameter.

Valid values for *class* are any SMS storage classes defined in the local environment.

[There is no default.]

References

Refer to the IBM JCL Reference manual for complete details.

2.20 ALLOC_UNIT

Description

The ALLOC_UNIT option is a dynamic allocation option that specifies the unit used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_unit *unit*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_unit *unit*					√
STRUDM Parameter	n/a					

Values

unit is equivalent to the JCL DD statement's UNIT parameter.

Valid values for *unit* are:

- *number*
  Device number
- *name*
  Unit generic or group name

[Default is SYSALLDA.]

References

Refer to the IBM JCL Reference manual for complete details.

2.21 ALLOC_VOLSER

Description

The ALLOC_VOLSER option is a dynamic allocation option that specifies the volume serial number used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-alloc_volser *number*					√
Environment Variable	n/a					
Configuration File Keyword	alloc_volser *number*					√
STRUDM Parameter	n/a					

Values

number is equivalent to the sub-parameter SER of the JCL DD statement's VOL parameter.

Valid values for *number* are any Volume serial number.

[There is no default.]

References

Refer to the IBM JCL Reference manual for complete details.

2.22 BIF_DIRECTORY

Description

The BIF_DIRECTORY option specifies the Broker Interface File (BIF) directory where the Universal Broker interface file, **`ubroker.bif`**, is located.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-bif_directory *directory*			√		
Environment Variable	UDMBIFDIRECTORY=*directory*			√		
Configuration File Keyword	n/a					
STRUDM Parameter	n/a					

Values

directory is the name of the BIF directory.

[Default is `/var/opt/universal`.]

2.23 CA_CERTIFICATES

Description

The CA_CERTIFICATES option specifies the location of the PEM-formatted trusted Certificate Authority (CA) X.509 certificates file.

Trust CA certificates are required if Universal Broker certificate authentication and verification is desired.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-ca_certs *ddname* or *file*			√	√	√
Environment Variable	UDMCACERTS=*file*	√		√	√	
Configuration File Keyword	ca_certificates *ddname* or *file*	√		√	√	√
STRUDM Parameter	CACERTS(*file [lib]*) [CACERTSMBR (*member*)]	√				

Values

z/OS

ddname is the ddname of the X.509 certificates. The value is used only when the SSL_IMPLEMENTATION option is set to *OPENSSL*.

Allocated to the ddname must be either a sequential data set or a member of a PDS that has a variable record format.

UNIX and Windows

file is the path name of the X.509 certificates file. Relative paths are relative the current working directory.

OS/400

file is the qualified file name of the X.509 certificates file. The file name can be qualified by a library name. If not, the library list ***LIBL** is searched for the first occurrence of the file name.

2.24 CERTIFICATE

Description

The CERTFIICATE option specifies the file / ddname name of the PEM-formatted X.509 certificate that identifies the UDM Manager.

A UDM Manager X.509 certificate is required if the Universal Broker requires client authentication.

Note: If the CERTIFICATE option is used, the PRIVATE_KEY option also is required.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-cert *ddname* or *file*			√	√	√
Environment Variable	UDMCERT=*file*	√		√	√	
Configuration File Keyword	certificate *ddname* or *file*	√		√	√	√
STRUDM Parameter	CERT(*file [lib]*) [CERTMBR (*member*)]	√				

Values

z/OS

ddname is the ddname of the X.509 certificate. The value is used only when the SSL_IMPLEMENTATION option is set to *OPENSSL*.

Allocated to the ddname must be either a sequential data set or a member of a PDS that has a variable record format.

UNIX and Windows

file is the path name of the X.509 certificate file. Relative paths are relative to the current working directory.

OS/400

file is the qualified file name of the X.509 certificate file. The file name can be qualified by a library name. If not, the library list ***LIBL** is searched for the first occurrence of the file name.

2.25 CERTIFICATE_REVOCATION_LIST

Description

The CERTIFICATE_REVOCATION_LIST option specifies the file name / ddname of the PEM-formatted file containing the Certificate Revocation List (CRL) issued by the trusted Certificate Authority.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-crl *ddname* or *file*			√	√	√
Environment Variable	UDMCRL=*file*	√		√	√	
Configuration File Keyword	crl *ddname* or *file*	√		√	√	√
STRUDM Parameter	CRLFILE(*file [lib]*) [CRLMBR(*member*)]	√				

Values

z/OS

ddname is the ddname of the file containing the CRL. The value is used only when the SSL_IMPLEMENTATION option is set to *OPENSSL*.

UNIX and Windows

file is the path name of the file containing the CRL. Relative paths are relative to the current working directory.

OS/400

file is the qualified file name of the CRL file. The file name can be qualified by a library name. If not, the library list ***LIBL** is searched for the first occurrence of the file name.

2.26 CODE_PAGE

Description

The CODE_PAGE option specifies the character code page that is used to translate text data received and transmitted over the network.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	-t *codepage*			√	√	√
Command Line Option, Long Form	-codepage *codepage*			√	√	√
Environment Variable	UDMCODEPAGE=*codepage*	√		√	√	
Configuration File Keyword	codepage *codepage*	√		√	√	√
STRUDM Parameter	CODEPAGE(*codepage*)	√				

Values

codepage is the character code page that is used to translate data.

codepage references a Universal Translate Table (UTT) file provided with the product (see Section 7.13 UTT Files for information on UTT files). UTT files are used to translate between Unicode and the local single-byte code page. (All UTT files end with an extension of **`.utt`**.)

Note: UTF-8 is not a supported codepage value for CODE_PAGE. UTF-8 codepage is valid only for text file data translation. Consequently, it can be specified only on the UDM open script statement.

Default

OS/400

- IBM037

UNIX

- ISO8859-1 (8-bit ASCII) ASCII-based operating systems
- IBM1047 (EBCDIC) Non-OS/400, EBCDIC-based operating system

z/OS

- ISO8859-1 (8-bit ASCII) ASCII-based operating systems
- IBM1047 (EBCDIC) Non-OS/400, EBCDIC-based operating system

(See Section 7.12 Character Code Pages for a complete list of character code pages provided by Stonebranch Inc. for use with Universal Products.)

2.27 CODEPAGE_TO_CCSID_MAP

Description

CAUTION: This option is intended only for use by OS/400 specialists who fully understand code pages, CCSIDs, how the two relate to each other, and how OS/400 uses CCSIDs for data translation between data streams and files. If a code page and CCSID are not correctly matched, data corruption will occur.

The CODEPAGE_TO_CCSID_MAP option specifies whether to use the internal table or external table for code page to CCSID mapping.

An internal table provides code page to CCSID mapping for the code page specified by the open command. The mapping only occurs if the CCSID is required for text file mapping. For the LIB file system, this includes mapping text to source physical files or to data files with an associated DDS file. All files in the root and QOpenSys file systems have associated CCSIDs.

This CCSID is not the same as the CCSID attribute associated with the attrib command; the UDM CCSID attribute determines the CCSID of the target file if the file does not exist. This CCSID in the mapping table is used as the CCSID associated with the attrib command when the default value, CODEPAGE, is specified; it also identifies the data stream CCSID, allowing the operating system to translate the code page translated data stream to the file. However, data written to or read from a file, record, or field with a CCSID of 65535 (or 'HEX') will not be translated.

The code page (and the mapped CCSID) from the open command is used for data mapping between the two parties involved in a data transfer. For OS/400, the code page also is used for mapping the data stream to or from the OS/400 file, whether a LIB or HFS transfer.

Under normal circumstances, the external mapping table will not be needed. This external table replaces the internal table, so all potentially needed code page to CCSID mappings must be provided in the external table.

Universal Products for OS/400 provides an example external table in file **CP2CCSID_X**, in product library **UNVPRD320**. For UDM to use the external table, create a single member file with the name **CDPG2CCSID** in the installation library. The example file may be used as a template by copying it to **CDPG2CCSID** in the same library.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	n/a					
Environment Variable	n/a					
Configuration File Keyword	codepage_to_ccsid_map_opt *table*	√				
STRUDM Parameter	n/a					

Value

table is the specification for which table to use:

- **error**
  Use the external table; if it is not found, report an error.
- **quiet**
  Use the external table; if it is not found, use the internal table. No message is issued.
- **internal**
  Use the internal table.

[Default is internal.]

2.28 COMMENT

Description

The COMMENT option specifies a user-defined string that can contain any value.

This comment will appear for the server(s) in a transfer session.

Note: You also can create a comment for the servers in a single session (or override the comment specified by this option) via the open command.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-comment *text*			√	√	√
Environment Variable	n/a					
Configuration File Keyword	n/a					
STRUDM Parameter	COMMENT (*user-defined string*)	√				

Value

text is the user-defined string.

2.29 CTL_SSL_CIPHER_LIST

Description

The CTL_SSL_CIPHER_LIST option specifies the acceptable and preferred SSL cipher suites to use for the control session between UDM components.

The SSL protocol uses the cipher suites to specify which encryption and message authentication (or message digest) algorithms to use.

Usage

Method Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-ctl_ssl_cipher_list *list*			√	√	√
Environment Variable	UDMCTLSSLCIPHERLIST=*list*	√		√	√	
Configuration File Keyword	ctl_ssl_cipher_list *list*	√		√	√	√
STRUDM Parameter	CTLCPHRLST(*cipherlist*)	√				

Values

list is a comma-separated list of SSL cipher suites. The list should be ordered with the most preferred suite first and the least preferred suite last.

Valid *list* values are:

- RC4-SHA 128-bit RC4 encryption and SHA-1 message digest
- RC4-MD5 128-bit RC4 encryption and MD5 message digest
- AES256-SHA 256-bit AES encryption and SHA-1 message digest
- AES128-SHA 128-bit AES encryption and SHA-1 message digest
- DES-CBC3-SHA 128-bit Triple-DES encryption and SHA-1 message digest
- DES-CBC-SHA 128-bit DES encryption and SHA-1 message digest

[Default is RC4-SHA,RC4-MD5,AES256-SHA,AES128-SHA,DES-CBC3-SHA, DES-CBC-SHA]

2.30 DATA_COMPRESSION

Description

The DATA_COMPRESSION option specifies whether or not data in standard I/O file transmissions across the network should be compressed.

Optionally, it also can specify the compression method to use.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	-k *option[,method]*			√	√	√
Command Line Option, Long Form	-compress *option[,method]*			√	√	√
Environment Variable	UDMCOMPRESS=*option[,method]*	√		√	√	
Configuration File Keyword	compress *option[,method]*	√		√	√	√
STRUDM Parameter	COMPRESS(**option*) and CMPRSMTH(**method*)	√				

Values

option is either of the following values:

- **yes**
  Data compression is required. All data in standard I/O file transmissions is compressed regardless of the UDM Server DATA_COMPRESSION option value.
- **no**
  Data compression is not required. However, data compression still can be requested via the UDM Server DATA_COMPRESSION option.

[Default is no.]

method is either of the following values:

- **zlib**
  Data is compressed using ZLIB compression algorithm. This method usually results in a very high compression rate, but tends to be somewhat CPU-intensive. It is recommended in environments where controlling a process's CPU usage is not necessarily a priority.
- **hasp**
  Data is compressed using the HASP compression algorithm. This method is less CPU-intensive than the ZLIB method. It is recommended in environments where controlling CPU usage is a priority. With HASP, the compression rate, while still very good, tends to be a little less than what is possible with the ZLIB.

[Default is zlib.]

2.31 DATA_SSL_CIPHER_LIST

Description

The DATA_SSL_CIPHER_LIST option specifies the acceptable and preferred SSL cipher suites to use for the data session on which file data is transferred between UDM primary and secondary servers.

The SSL protocol uses the cipher suites to specify which encryption and message authentication (or message digest) algorithms to use.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-data_ssl_cipher_list *list*			√	√	√
Environment Variable	UDMDATASSLCIPHERLIST=*list*	√		√	√	
Configuration File Keyword	data_ssl_cipher_list *list*	√		√	√	√
STRUDM Parameter	DTACPHRLST(*cipherlist*)	√				

Values

list is a comma-separated list of SSL cipher suites. The list should be ordered with the most preferred cipher suite first and the least preferred cipher suite last.

Table 7.15 SSL Cipher Suites for UDM identifies the list of SSL cipher suites provided for UDM by Stonebranch Inc.

Default

The default list of SSL cipher suites is:

RC4-SHA,RC4-MD5,AES256-SHA,AES128-SHA,DES-CBC3-SHA,DES-CBC-SHA, NULL-SHA,NULL-MD5

2.32 EVENT_GENERATION

Description

The EVENT_GENERATION option specifies which events are to be generated and processed as persistent events.

A persistent event record is saved in a Universal Enterprise Controller (UEC) database for long-term storage.

(For a list of all event types for all Universal Products components, see the Universal Event Subsystem 3.2.0 Event Definitions Guide.)

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	n/a					
Environment Variable	n/a					
Configuration File Keyword	event_generation *types*	√		√	√	√
STRUDM Parameter	n/a					

Values

type specifies a comma-separated list of event types. It allows for all or a subset of all potential event message types to be selected.

Event type ranges can be specified by separating the lower and upper range values with a dash (**-**) character.

Event types can be selected for inclusion or exclusion:

- Inclusion operator is an asterisk (*****).
- Exclusion operator is **X** or **x**.

Examples

- 100,101,102
  Generate event types 100, 101, and 102.
- 100-102
  Generate event types 100 through 102.
- 100-102,200
  Generate event types 100 through 102 and 200.
- *
  Generate all event types.
- *,X100
  Generate all event types except for 100.
- x*
  Generate no event types.
- *,X200-250,X300
  Generate all event types except for 200 through 250 and 300.

[Default is *X (no event types).]**

2.33 FRAME_INTERVAL

Description

The FRAME_INTERVAL options sets the number of UDM transfer blocks transferred before a frame-sync message is sent when UDM is operating with network fault tolerance on (see 2.43 NETWORK_FAULT_TOLERANT).

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-frame_interval *number*			√	√	√
Environment Variable	UDMFRAMEINTERVAL *number*	√		√	√	√
Configuration File Keyword	frame_interval *number*	√		√	√	√
STRUDM Parameter	FRAMEINT(*number*)	√				

Values

number can be any number.

[Default is *100*.]

Note: This value should not be changed without direction from Stonebranch, Inc. Customer Support. Changing this value could degrade UDM performance.

2.34 HELP

Description

The HELP option prints a description of the command options and their format.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	-h			√	√	√
Command Line Option, Long Form	-help			√	√	√
Environment Variable	n/a					
Configuration File Keyword	n/a					

Values

There are no values used with this option.

2.35 IDLE_TIMEOUT

Description

The IDLE_TIMEOUT option sets the number of seconds of inactivity in an interactive UDM session after which the UDM Manager will close the session.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-idle_timeout *seconds*			√	√	√
Environment Variable	UDMIDLETIMEOUT=*seconds*	√		√	√	√
Configuration File Keyword	idle_timeout *seconds*	√		√	√	√
STRUDM Parameter	IDLTIMOUT(*seconds*)	√				

Values

seconds can be any number.

[Default is *1200*.]

2.36 INSTALLATION_DIRECTORY

Description

The INSTALLATION_DIRECTORY option specifies the directory in which Universal Data Mover is installed.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	n/a					
Environment Variable	n/a					
Configuration File Keyword	installation_directory *directory*			√	√	
STRUDM Parameter	n/a					

Values

directory is any directory.

Defaults

UNIX

[Default is `/opt/universal/udmmgr`.]

Windows

[Default is UDM Manager installation file: `c:\Program Files\Universal\udmmgr`.]

2.37 KEEP_ALIVE_INTERVAL

Description

The KEEP_ALIVE_INTERVAL option sets the default interval (in seconds) at which a keep alive message is sent from the UDM Manager to the transfer server(s). If the transfer server(s) do not respond to the keep alive message within 3X this interval, a network fault is registered.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-keep_alive_interval *seconds*			√	√	√
Environment Variable	UDMKEEPALIIVEINTERVAL= *seconds*	√		√	√	√
Configuration File Keyword	keep_alive_interval *seconds*	√		√	√	√
STRUDM Parameter	KEEPALIVE(*seconds*)	√				

Values

seconds can be any number.

[Default is *120*.]

2.38 MERGE_LOG

Description

The MERGE_LOG option specifies whether or nor to merge standard out and standard error output streams from a remote command to the UDM transaction log.

(See Section Chapter 15 Remote Execution in the Universal Data Mover 3.2.0 User Guide).

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	n/a					
Environment Variable	UDMMERGELOG=*option*	√		√	√	√
Configuration File Keyword	mergelog *option*	√		√	√	√
STRUDM Parameter	n/a					

Values

Valid *option* values are:

- **yes**
  Merge the standard out and standard error output streams.
- **no**
  Do not merge the standard out and standard error output streams.

[Default is no.]

2.39 MESSAGE_LANGUAGE

Description

The MESSAGE_LANGUAGE option specifies the Universal Message Catalog (UMC) file used to format messages.

UDM provides specific UMC files for specific languages. The first three characters of the language name are used as a three-character suffix in the UMC file base name, **UDMMC**. All UMC files then have a **.UMC** extension.

UMC files are located in the **UNVNLS** file in the Universal Products installation library (default is **UNVPRD320**).

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	-L *language*			√	√	√
Command Line Option, Long Form	-lang *language*			√	√	√
Environment Variable	UDMLANG=*language*	√		√	√	
Configuration File Keyword	language *language*	√		√	√	√
STRUDM Parameter	MSGLANG(*language*)	√				

Values

language is any UMC file provided for UDM by Stonebranch Inc.

Note: For the current release of UDM, English is the only available language.

2.40 MESSAGE_LEVEL

Description

The MESSAGE_LEVEL option specifies the level of messages to write.

It also specifies, optionally, whether or not to write a date and time stamp with each message.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	-l *level[,time]*			√	√	√
Command Line Option, Long Form	-level *level[,time]*			√	√	√
Environment Variable	UDMLEVEL=*level[,time]*	√		√	√	
Configuration File Keyword	message_level *level[,time]*	√		√	√	√
STRUDM Parameter	MSGLEVEL(**level[*time]*)	√				

Values

level indicates either of the following level of messages:

- **trace**
  Activates tracing and generates a trace file to which UDM writes trace messages used for debugging.
  - UNIX Trace file (**`udm.trc`**) is created in the current working directory.
  - Z/OS Trace file is written to the UNVTRACE ddname.
  - OS/400 Trace file name is **`*CURLIB/UNVTRCUDM(Txxxxxx)`** where **`xxxxxx`** is the process ID number of the job invoking Universal Data Mover.

  Note: Use **trace** only as directed by Stonebranch, Inc. Customer Support.
- **audit**
  Issues audit, informational, warning, and error messages.
- **info**
  Issues informational, warning, and error messages.
- **warn**
  Issues warning and error messages.
- **error**
  Issues error messages only.

OS/400 and z/OS

[Default is info.]

UNIX and Windows

[Default is warn.]

time specifies either of the following:

- **time**
  Include a time and date stamp on each message.
- **notime**
  Do not include a time and date stamp on each message.

OS/400

time specifies either of the following:

- **yes**
  Include a time and date stamp on each message.
- **no**
  Do not include a time and date stamp on each message.

[Default is **yes**.]

z/OS

[Default is time.]

UNIX and Windows

[Default is notime.]

2.41 MODE_TYPE

Description

The MODE_TYPE option specifies the default transfer mode type for UDM sessions.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-mode_type *type*			√	√	√
Environment Variable	UDMMODETYPE=*type*	√		√	√	√
Configuration File Keyword	mode_type *type*	√		√	√	√
STRUDM Parameter	MODETYPE (**type*)	√				

Values

Valid *type* values are:

- **binary**
  Default transfer mode type is binary.
- **text**
  Default transfer mode type is text.

[Default is binary.]

OS/400

Valid *type* values are:

- **bin**
  Default transfer mode type is binary.
- **binary**
  Default transfer mode type is binary.
- **text**
  Default transfer mode type is text.

[Default is bin.]

2.42 NETWORK_DELAY

Description

The NETWORK_DELAY option sets the expected network latency (in seconds).

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-delay *seconds*			√	√	√
Environment Variable	UDMDELAY=*seconds*	√		√	√	√
Configuration File Keyword	network_delay *seconds*	√		√	√	√
STRUDM Parameter	DELAY(*seconds*)	√				

Values

seconds is any number.

[Default is *120*.]

2.43 NETWORK_FAULT_TOLERANT

Description

The NETWORK_FAULT_TOLERANT option sets whether or not UDM transfer sessions are network fault tolerant by default.

The Network Fault Tolerant (NFT) feature allows UDM to recover from network faults and continue processing without interruption. NFT is turned on or off with the UDM script open command.

NETWORK_FAULT_TOLERANT sets the default NFT value if it is not specified in the open command.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-network_fault_tolerant *option*			√	√	√
Environment Variable	UDMNETWORKFAULTTOLERANT= *option*	√		√	√	√
Configuration File Keyword	network_fault_tolerant *option*	√		√	√	√
STRUDM Parameter	NETWORKFT(**option*)	√				

Values

option can be either of the following values:

- **yes**
  Network fault tolerance is on for UDM sessions, allowing UDM to attempt to recover from a network fault and resume the session.
- **no**
  Network fault tolerance is off for UDM sessions.

[Default is yes.]

2.44 NLS_DIRECTORY

Description

The NLS_DIRECTORY option specifies the name of the directory where the UDM Manager message catalog and code page tables are located.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	n/a					
Environment Variable	n/a					
Configuration File Keyword	nls_directory *directory*			√	√	
STRUDM Parameter	n/a					

Values

directory can be any directory.

Full path names are recommended.

Relative path names are relative to the **`universal`** installation directory.

UNIX

[Default is **`/opt/universal/nls`**.]

Windows

[Default is **`..\nls`**.]

2.45 OPEN_RETRY

Description

The OPEN_RETRY option provides a level of fault tolerance for the **open** command.

If UDM cannot establish a transfer session due to network error — because a remote Broker was not running or, basically, for any reason other than an invalid user name or password — UDM will wait a period of time (specified by the OPEN_RETRY_INTERVAL option) and then retry to open a session.

UDM will attempt to establish a session until it is successful or it reaches the retry limit (as specified by the OPEN_RETRY_COUNT option). If the retry limit is reached, UDM will stop attempting to establish a session and will return an error.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	-open_retry *option*			√	√	√
Command Line Option, Long Form	-open_retry *option*			√	√	√
Environment Variable	UDMOPENRETRY=*option*	√		√	√	√
Configuration File Keyword	open_retry *option*	√		√	√	√
STRUDM Parameter	OPENRETRY (**enable count interval*)	√				

Values

Valid *option* values are:

- **yes**
  OPEN_RETRY used by UDM Manager.
- **no**
  OPEN_RETRY not used by UDM Manager.

[Default is no.]

OS/400

The STRUDM parameter (**OPENRETRY**) value contains three elements: *enable*, *count*, and *interval*. These are the same values specified by the OPEN_RETRY, OPEN_RETRY_COUNT, and OPEN_RETRY_INTERVAL options in the configuration file or when using environment variables.

Valid values for *enable* are:

- **yes**
  Enable open retry.
- **no**
  Disable open retry.

[Default is no.]

count is the maximum number of attempts that will be made to establish a session by the open command.

[Default is 20.]

interval is the number of seconds that UDM will wait between each open retry attempt.

[Default is 60.]

For example: **`OPENRETRY(*yes 20 45)`**.

2.46 OPEN_RETRY_COUNT

Description

The OPEN_RETRY_COUNT option sets the maximum number of attempts that will be made to establish a session by the open command.

This option is used only if the OPEN_RETRY option is being used by UDM (value=*yes*)

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-open_retry_count *number*			√	√	√
Environment Variable	UDMOPENRETRYCOUNT=*number*	√		√	√	√
Configuration File Keyword	open_retry_count *number*	√		√	√	√
STRUDM Parameter	n/a (see OPEN_RETRY option)					

Values

number is any number.

{Default is *20*.]

2.47 OPEN_RETRY_INTERVAL

Description

The OPEN_RETRY_INTERVAL option sets the number of seconds that UDM will wait between each open retry attempt (see OPEN_RETRY_COUNT).

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-open_retry_interval *number*			√	√	√
Environment Variable	UDMOPENRETRYINTERVAL= *number*	√		√	√	√
Configuration File Keyword	open_retry_interval *number*	√		√	√	√
STRUDM Parameter	n/a (see OPEN_RETRY option)					

Values

number is any number (of seconds).

[Default is *60*.]

2.48 OUTBOUND_IP

Description

The OUTBOUND_IP option sets the host or IP address that UDM binds to when initiating outgoing connections to another UDM server.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-outboundip *host*			√	√	√
Environment Variable	UDMOUTBOUNDIP=*host*	√		√	√	√
Configuration File Keyword	outbound_ip *host*	√		√	√	√
STRUDM Parameter	OUTBOUNDIP (*host*)	√				

Values

host is the required IP address.

[There is no default.]

2.49 PLF_DIRECTORY

Description

The PLF_DIRECTORY option specifies the Program Lock File (PLF) directory where the program lock files are located.

A program lock file is created and used by the UDM Manager process to store manager process termination information for the Universal Broker.

OS/400

Do not include this directory in any system or backup that requires an exclusive lock on the directory while UDM is running.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-plf_directory *directory*			√		
Environment Variable	UDMPLFDIRECTORY=*directory*			√		
Configuration File Keyword	n/a					
STRUDM Parameter	PLFDIR (*directory*)	√				

Values

directory is the name of the PLF directory.

A full path name must be specified.

Defaults

UNIX

[Default is `/var/opt/universal/tmp`.]

OS/400

[Default is `/tmp`.]

2.50 PRIVATE_KEY

Description

The PRIVATE_KEY option specifies the location of the PEM-formatted RSA private key that corresponds to the X.509 certificates specified by the CERTIFICATE option.

Note: PRIVATE_KEY is required only if a certificate is specified by CERTIFICATE.

z/OS

PRIVATE_KEY is used only when the SSL_IMPLEMENTATION option is set to *OPENSSL*.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-private_key *ddname* or *file*			√	√	√
Environment Variable	UDMPRIVATEKEY=*file*	√		√	√	
Configuration File Keyword	private_key *ddname* or *file*	√		√	√	√
STRUDM Parameter	PVTKEYF(*file [lib]*) [PVTKEYFMBR (*member*)]	√				

Values

z/OS

ddname is the ddname of the PEM-formatted RSA private key that corresponds to the X.509 certificates.

Allocated to the ddname must be either a sequential data set or a member of a PDS that has a variable record format.

UNIX and Windows

file is the path of the PEM-formatted RSA private key file that corresponds to the X.509 certificates.

OS/400

file is the qualified name of the PEM-formatted RSA private key file that corresponds to the X.509 certificates.

The file name can be qualified by a library name. If not, the library list ***LIBL** is searched for the first occurrence of the file name.

2.51 PRIVATE_KEY_PWD

Description

The PRIVATE_KEY_PWD option specifies the password or passphrase for the PEM-formatted RSA private key specified with the PRIVATE_KEY option.

Note: Whether or not the password is required or not depends on whether or not it is required by the private key.

z/OS

PRIVATE_KEY_PWD is used only when the SSL_IMPLEMENTATION option is set to *OPENSSL*.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-private_key_pwd *password*			√	√	√
Environment Variable	UDMPRIVATEKEYPWD=*password*	√		√	√	
Configuration File Keyword	private_key_password=*password*	√		√	√	√
STRUDM Parameter	PVTKEYPWD (*password*)	√				

Values

password is the password for the private key.

OS/400

Characters may be incorrectly translated due to reverse representations under 037 and 1047 CCSIDs:

- hat (circumflex) logical not
- left bracket Y acute
- right bracket diaeresis (umlaut)

The hex/decimal exchanges are:

- 5F/95 B0/176
- AD/173 BA/186
- BD/189 BB/187

2.52 PROXY_CERTIFICATES

Description

The PROXY_CERTFIICATES option specifies whether or not UDM will use the managers certificate in a three-party transfer session if a certificate is supplied to the UDM Manager.

Proxy certificates are used only for three-party transfer sessions. All components, manager, primary and secondary, must be version 3.2 or later and must be using OpenSSL (System SSL does not support proxy certificates).

If PROXY_CERTIFICATES is set to yes, the UDM Manager's certificate is used to create a proxy certificate for the primary to use when connecting to the secondary. The proxy certificate has the same subject name as the managers certificate, so the certificate ACL for the secondary can be set up to look just like the primary's ACL.

Note: For more information on X509 proxy certificates, see the RFC at:

http://www.globus.org/alliance/publications/papers/pki04-welch-proxy-cert-final.pdf

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-proxy_certificates *option*			√	√	√
Environment Variable	UDMPROXYCERTIFICATES=*option*	√		√	√	
Configuration File Keyword	proxy_certificates *option*	√		√	√	√
STRUDM Parameter	PROXYCERT (*option*)	√				

Values

option is the specification for whether or not UDM will use proxy certificates.

Valid values for *option* are:

- **yes**
  UDM will use proxy certificates.
- **no**
  UDM will not use proxy certificates.

[Default is no.]

2.53 RECONNECT_RETRY_COUNT

Description

The RECONNECT_RETRY_COUNT option sets the number of attempts that the UDM Manager will make to re-establish a transfer session when a network fault occurs.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-retry_count *number*			√	√	√
Environment Variable	UDMRETRYCOUNT=*number*	√		√	√	√
Configuration File Keyword	reconnect_retry_count *number*	√		√	√	√
STRUDM Parameter	RETRYCNT(*number*)	√				

Values

number is any number.

{Default is *20*.]

2.54 RECONNECT_RETRY_INTERVAL

Description

The RECONNECT_RETRY_INTERVAL option sets the number of seconds that UDM will wait between each successive attempt to reestablish a transfer session when a network fault occurs.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-retry_interval *seconds*			√	√	√
Environment Variable	UDMRETRYINTERVAL=*seconds*	√		√	√	√
Configuration File Keyword	reconnect_retry_interval *seconds*	√		√	√	√
STRUDM Parameter	RETRYINT(*seconds*)	√				

Values

seconds is any number of seconds to wait.

[Default is *60*.]

2.55 RECV_BUFFER_SIZE

Description

The RECV_BUFFER_SIZE option sets the size (in bytes) of the TCP receive buffer for UDM.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-recvbuffersize *size*			√	√	√
Environment Variable	UDMRECVBUFSSIZE=*size*	√		√	√	√
Configuration File Keyword	recv_buffer_size *size*	√		√	√	√
STRUDM Parameter	RCVBUFSIZE(*size*)	√				

Values

size is the number of bytes.

[Default is *0*.]

(The default, **0,** instructs UDM to use the operating system default.)

Note: The size of the TCP receive buffer should be changed only when performance tweaking is necessary. Changing this value could affect performance adversely.

2.56 REMOTE_PORT

Description

The REMOTE_PORT option specifies the TCP port on the remote computer used for invoking UDM Server instances.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	-p *port*			√	√	√
Command Line Option, Long Form	-port *port*			√	√	√
Environment Variable	UDMPORT=*port*	√		√	√	
Configuration File Keyword	port *port*	√		√	√	√
STRUDM Parameter	PORT(*port*)	√				

Values

port is the TCP port on the remote computer.

Valid values for *port* are:

- Number
- Service name (for example, **ubroker**)

[Default is *7887*.]

2.57 SAF_KEY_RING

Description

The SAF_KEY_RING option specifies the SAF (RACF is a SAF implementation) certificate key ring name that the Universal Data Mover job should used for its certificate.

The key ring must be associated with the user profile with which the Universal Data Mover job executes.

Note: SAF_KEY_RING is required if the SSL_IMPLEMENTATION option is set to **system**.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-saf_key_ring *name*					√
Environment Variable	UDMSAFKEYRING=*name*					√
Configuration File Keyword	saf_key_ring *name*					√
STRUDM Parameter	n/a					

Values

name is the name of the SAF certificate key ring.

2.58 SAF_KEY_RING_LABEL

Description

The SAF_KEY_RING_LABEL option specifies the label of the certificate in the SAF (RACF is a SAF implementation) certificate key ring that the Universal Data Mover job should use for its certificate.

(The key ring is specified by the SAF_KEY_RING option.)

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-saf_key_ring_label *label*					√
Environment Variable	UDMSAFKEYRINGLABEL=*label*					√
Configuration File Keyword	saf_key_ring_label *label*					√
STRUDM Parameter	n/a					

Values

label is the label of the SAF certificate key ring.

[Default is the default certificate in the key ring.]

2.59 SCRIPT

Description

The SCRIPT option specifies the ddname from which to read a UDM script command file.

Note: This option overrides the default **UNVSCR** ddname.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	-s *ddname*					√
Command Line Option, Long Form	-script *ddname*					√
Environment Variable	n/a					
Configuration File Keyword	n/a					
STRUDM Parameter	n/a					

Values

ddname is the ddname from which to read the file.

2.60 SCRIPT_FILE

Description

The SCRIPT option specifies a script file containing UDM commands to execute.

UDM requires a member name; however, the value *FILE is valid and provides the OS/400 default file search order.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	-s *filename*			√	√	
Command Line Option, Long Form	-script *filename*			√	√	
Environment Variable	UDMSCRIPT=*filename*	√		√	√	
Configuration File Keyword	n/a					
STRUDM Parameter	SCRFILE([*library*/] *filename*) [SCRMBR (*member*)]	√				

Values

filename is the name of the file from which the script is read.

2.61 SCRIPT_OPTIONS

Description

The SCRIPT_OPTIONS option specifies options to pass to the script command file.

Note: This option is valid only if a script file is specified.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	-o *options*			√	√	√
Command Line Option, Long Form	-options *options*			√	√	√
Environment Variable	UDMOPTIONS=*options*	√		√	√	
Configuration File Keyword	n/a					
STRUDM Parameter	OPTIONS (*options*)	√				

Values

options is the name of the options to pass to the script file.

OS/400, UNIX, and Windows

If *options* contain spaces, it must be enclosed in double (**"**) quotation marks. If a quotation mark is part of the value, prefix it with the Windows escape character, backslash (\ ).

z/OS

The format of *options* is the same as the UDM script call command.

2.62 SEND_BUFFER_SIZE

Description

The SEND_BUFFER_SIZE option sets the size (in bytes) of the TCP send buffer for UDM.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-sendbuffersize *size*			√	√	√
Environment Variable	UDMSENDBUFSSIZE=*size*	√		√	√	√
Configuration File Keyword	Send_buffer_size *size*	√		√	√	√
STRUDM Parameter	SNDBUFSIZE (*size*)	√				

Values

size is the number of bytes.

[Default is *0*.]

(The default, **0,** instructs UDM to use the operating system default.)

Note: The size of the TCP send buffer should be changed only when performance tweaking is necessary. Changing this value could affect performance adversely.

2.63 SERVER_STOP_CONDITIONS

Description

The SERVER_STOP_CONDITIONS option specifies one or more exit codes of the executing UDM Manager that should trigger the locally running Universal Broker to cancel the corresponding UDM Server for the exited UDM Manager.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-server_stop_conditions *codes*					√
Environment Variable	UDMSERVERSTOPCONDITIONS= *codes*					√
Configuration File Keyword	server_stop_conditions *codes*					√
STRUDM Parameter	n/a					

Values

codes is an exit code, or a comma-separated list of exit codes, that should cause the UDM Server to be cancelled.

z/OS ABEND codes are specified in two different formats:

- System ABEND code Starts with S followed by a 3-character hexadecimal value.
- User ABEND code Starts with U followed by a 4-character decimal value.

For example, when a job is terminated with the CANCEL console command, the job ends with a system ABEND code of S222.

[There is no default.]

2.64 SIZE_ATTRIB

Description

The SIZE_ATTRIB option sets the default size (number of records) for file creation of physical files for both data and source file types.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	n/a					
Environment Variable	UDMSIZEATTRIB=*size*	√				
Configuration File Keyword	size_attrib *size*	√				
STRUDM Parameter	n/a					

Values

size is the number of records.

Other than the leading * for *NOMAX, it can be any valid value. (To specify *NOMAX, enter 'NOMAX'.)

The values for Initial Number of Records, Increment Number of Records, and Maximum increments are comma-separated, not space-separated.

Examples:

- 1000,100,100
- 10000,499
- 50000

UDM uses this value as the default file SIZE attribute. (See Table 14.2 OS/400-Specific LIB File Attributes for Creating New Files in the Universal Data Mover 3.2.0 User Guide for details.)

[default is *empty string*].

2.65 SSL_IMPLEMENTATION

Description

The SSL_IMPLEMENTATION option specifies the Secure Socket Layer (SSL) implementation to be used for network communications.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-ssl_implementation *option*					√
Environment Variable	UDMSSLIMPLEMENTATION=*option*					√
Configuration File Keyword	ssl_implementation *option*					√
STRUDM Parameter	n/a					

Values

option is the SSL implementation to be used.

Valid values for option are:

- **openssl**
  OpenSSL SSL library is used for the SSL protocol.
- **system**
  z/OS System SSL library is used for the SSL protocol. The z/OS System SSL library has installation and configuration prerequisites. (See the Universal Products 3.2.0 Installation Guide for a description of the prerequisites before using System SSL.)

[Default is openssl.]

2.66 SYSTEM_ID

Description

The SYSTEM_ID option identifies the local Universal Broker with which the UDM Manager must register before the Manager performs any request.

Each Universal Broker running on a system is configured with a system identifier that uniquely identifies the Broker.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-system_id *ID*					√
Environment Variable	UDMSYSTEMID=*ID*					√
Configuration File Keyword	n/a					
STRUDM Parameter	n/a					

Values

ID is the system identifier of the local Universal Broker.

References

Refer to the local Universal Broker administrator for the appropriate system ID to use.

2.67 TCP_NO_DELAY

Description

The TCP_NO_DELAY option specifies whether or not to use TCP packet coalescing.

The packet coalescing algorithm, which can delay the sending of small amounts of data over the network, is designed to improve network congestion. However, since it can have a significantly negative effect on the performance of UDM, TCP_NO_DELAY specifies – by default – not to use TCP packet coalescing.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-tcp_no_delay *option*			√	√	√
Environment Variable	UDMTCPNODELAY=*option*			√	√	√
Configuration File Keyword	tcp_no_delay *option*			√	√	√
STRUDM Parameter	n/a					

Values

option is the specification for whether or not to use TCP packet coalescing.

Valid values for *option* are:

- **yes**
  Do not use TCP packet coalescing.
- **no**
  Use TCP packet coalescing.

[Default is yes.]

2.68 TRACE_FILE_LINES

Description

The TRACE_FILE_LINES option specifies the maximum number of lines to write to the trace file.

When the maximum number of lines has been reached, the trace file will wrap around and start writing trace entries after the trace header lines.

Tracing is activated, and a trace file is generated, when the MESSAGE_LEVEL option is set to **trace**.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-tracefilelines *number*			√	√	√
Environment Variable	UDMTRACEFILELINES=*number*	√		√	√	
Configuration File Keyword	trace_file_lines *number*	√		√	√	√
STRUDM Parameter	TRCLINES(*number*)	√				

Values

number is the maximum number of lines.

[Default is *200,000*.]

z/OS

Note: This option has no effect when the UNVTRACE ddname allocates a JES SYSOUT data set. The average size of a trace file line is 50 characters.

UNIX

The average size of a trace file line is 50 characters.

OS/400

The current record length is 384 characters.

As allocated on the AS/400 (i5), the maximum number of records is 509,000. To increase the value of TRACE_FILE_LINES beyond this number, a manual adjustment of the file size via the CHGPF command is required.

Note:

1. The file is not created until the first time Universal Data Mover Manager trace is used.
2. Deleting the trace file will reset the maximum number of records to 509,000.
3. To clear the trace file without resetting the maximum number of records that the file may contain, use the **RMVM FILE(*CURLIB/UNVTRCUDM) MBR(*ALL)** command. Substitute an appropriate library for ***CURLIB** if needed.

Note: Setting a number of records larger than the maximum number of records allocated to the trace file will result in a prompt sent to **QSYSOPR**. Unless a default response is set for this message (CPA5305), the PROCESS WILL HANG waiting for the **QSYSOPR** response.

2.69 TRACE_TABLE

Description

The TRACE_TABLE option specifies the size of a wrap-around trace table maintained in memory.

The trace table is written when the program ends under the conditions specified by this option.

Tracing is activated, and a trace file is generated, when the MESSAGE_LEVEL option is set to **trace**.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-trace_table *size,condition*			√	√	√
Environment Variable	UDMTRACETABLE=*size,condition*	√		√	√	
Configuration File Keyword	trace_table *size,condition*	√		√	√	√
STRUDM Parameter	TRCTBL(*size, *condition)*	√				

Values

size is the size (in bytes) of the table. (A value of *0* indicates that the trace table is not used.)

The size can be suffixed with either of the following characters:

- **M** indicates that the size is specified in megabytes
- **K** indicates that the size is specified in kilobytes

For example, **50M** indicates that 50 X 1,048,576 bytes of memory is allocated for the trace table.

condition is the condition under which the trace table is written.

Possible values for *condition* are:

- **error**
  Write the trace table if the program ends with a non-zero exit code.
- **always**
  Write the trace table when the program ends regardless of the exit code.
- **never**
  Never write the trace table.

2.70 UCMD_PATH

Description

The UCMD_PATH option sets the complete path to Universal Command (UCMD) for calls by the **exec** command.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	n/a					
Environment Variable	UDMUCMDPATH=*path*			√	√	
Configuration File Keyword	ucmd_path *path*			√	√	
STRUDM Parameter	n/a					

Values

path must contain the UCMD file itself (for example: **/opt/universal/bin/ucmd**).

By default, the UCMD_PATH option is not set; UDM uses the **PATH** environment variable to call UCMD.

2.71 UMASK

Description

The UMASK option specifies the file mode creation mask. It affects the file permission bits of newly created files.

All files are created with a permission mode of 666, which is read-write permission for the owner, group, and other permission categories. UDM uses UMASK to turn off selected permission bits by subtracting the value of UMASK from mode 666.

OS/400

UMASK applies only to the **HFS** file system.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	-umask *number*			√	√	√
Environment Variable	UDMUMASK=*number*	√		√	√	√
Configuration File Keyword	umask *number*	√		√	√	√
STRUDM Parameter	n/a					

Values

number can be any number, 001 to 666.

[Default is *026*.]

The default value (**026**) results in file permission 640 (666 - 026 = 640), which is:

- read-write for the owner
- read for the group
- none for others

References

Refer to the UNIX man page umask(1) for complete details.

2.72 USAP_PATH

Description

The USAP_PATH option sets the complete path to USAP for calls by the **execsap** command.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	n/a					
Command Line Option, Long Form	n/a					
Environment Variable	UDMUSAPDPATH=*path*			√		
Configuration File Keyword	usap_path *path*			√		
STRUDM Parameter	n/a					

Values

path must contain the USAP file itself (for example: **/opt/universal/bin/usap**).

By default, the USAP_PATH option is not set; UDM uses the **PATH** environment variable to call USAP.

2.73 VERSION

Description

The VERSION option writes the program version information and copyright.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Command Line Option, Short Form	-v			√	√	√
Command Line Option, Long Form	-version			√	√	√
Environment Variable	n/a					
Configuration File Keyword	n/a					
STRUDM Parameter	VERSION(**option*)	√				

Values

There are no values used with this option.

OS/400

Valid values for *option* are:

- **yes**
  Write the program version information and copyright.
- **no**
  Do not write the program version information and copyright.

Chapter 3
Universal Data Mover Server Configuration Options

3.1 Overview

This chapter provides detailed information on the configuration options available for use with the Universal Data Mover Server.

The options are listed alphabetically, without regard to any specific operating system.

Information on how these options are used is documented in the Universal Data Mover 3.2 User Guide.

Section 3.2 Configuration Options Information provides a guideline for understanding the information presented or each option.

3.2 Configuration Options Information

For each configuration option, this chapter provides the following information.

Description

Describes the configuration option and how it is used.

Usage

Provides a table of the following information:

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	<Format / Value>					

Method

Identifies the method used to specify Universal Data Mover Server configuration options:

- Configuration File Keyword

Syntax

Identifies the syntax of the method used to specify the option:

- Format Specific characters that identify the option.
- Value Type of value(s) to be supplied for this method.

(Operating System)

Identifies (with a ✔) the operating systems for which each method of specifying the option is valid:

- OS/400
- HP NonStop
- UNIX
- Windows
- z/OS

Values

Identifies all possible values for the specified value type.

Defaults are identified in **[bracketed bold type]**.

<Additional Information>

Identifies any additional information specific to the option.

3.3 Configuration Options List

Table 3.1 identifies all UDM Server configuration options.

Option	Description	Page
ACTIVITY_MONITORING	Specification for whether or not product activity monitoring events are generated.	113
ALLOC_ABNORMAL_DISP	Disposition of data set when an abnormal ending occurs.	112
ALLOC_BLKSIZE	Block size used for newly allocated data sets.	115
ALLOC_DATACLAS	SMS data class used for newly allocated data sets.	116
ALLOC_DIR_BLOCKS	Number of directory blocks for newly allocated partitioned data sets.	117
ALLOC_DSORG	Data set organization used for newly allocated data sets.	118
ALLOC_INPUT_STATUS	Status of data sets being allocated for input.	119
ALLOC_LRECL	Logical record length used for newly allocated data sets.	120
ALLOC_MGMTCLAS	SMS management class used for newly allocated data sets.	121
ALLOC_NORMAL_DISP	Disposition of data set when normal ending occurs.	122
ALLOC_OUTPUT_STATUS	Status of existing data sets being allocated for output.	123
ALLOC_PRIM_SPACE	Primary space allocation used for newly allocated data sets.	124
ALLOC_RECFM	Record format used for newly allocated data sets.	125
ALLOC_SEC_SPACE	Secondary space allocation used for newly allocated data sets.	126
ALLOC_SPACE_UNIT	Space unit in which space is allocated for newly allocated data sets.	127
ALLOC_STORCLAS	SMS storage class used for newly allocated data sets.	128
ALLOC_UNIT	Unit used for newly allocated data sets.	129
ALLOC_VOLSER	Volume serial number used for newly allocated data sets.	130
CODE_PAGE	Character code page used to translate text data.	131
CODEPAGE_TO_CCSID_MAP	Specification to use the internal or external table for code page to CCSID mapping.	133
DATA_COMPRESSION	Specification for whether or not data is compressed on all standard I/O files.	135
DATA_SSL_CIPHER_LIST	SSL cipher suites to use for data session between UDM primary and secondary servers.	136
EVENT_GENERATION	Events to be generated as persistent events.	137
FRAME_INTERVAL	Number of UDM transfer blocks transferred before a frame-sync message is sent with network fault tolerance turned on.	139
INSTALLATION_DIRECTORY	Directory in which UDM Server is installed.	140
LOGON_METHOD	Specification for how users are logged onto the system.	141
MESSAGE_LEVEL	Level of messages that UDM will write.	142
NETWORK_DELAY	Expected network latency (in seconds).	144

Option	Description	Page
NLS_DIRECTORY	Directory where the UDM Server message catalog and code page tables are located.	145
OUTBOUND_IP	Host or IP address that UDM binds to when initiating outgoing connections to another UDM Server.	146
RECONNECT_RETRY_COUNT	Number of attempts that the UDM Manager will make to re-establish a transfer session when a network fault occurs.	147
RECONNECT_RETRY_INTERVAL	Number of seconds that UDM will wait between each successive attempt to re-establish a transfer session when a network fault occurs.	148
RECV_BUFFER_SIZE	Size of the TCP receive buffer for UDM.	149
SEND_BUFFER_SIZE	Size of the TCP send buffer for UDM.	150
SIZE_ATTRIB	Default size for file creation of physical files for both data and source file types.	151
TCP_NO_DELAY	Specification for whether or not to use TCP packet coalescing.	152
TMP_DIRECTORY	Name of directory that UDM Server uses for temporary files.	153
TRACE_DIRECTORY	Name of directory that UDM Server uses for its trace files.	154
TRACE_FILE_LINES	Maximum number of lines to write to the trace file.	155
TRACE_TABLE	Size of a wrap-around trace table maintained in memory.	157
UMASK	File mode creation mask.	158
USER_SECURITY	User security option.	159

Table 3.1 UDM Server Configuration Options

3.4 ACTIVITY_MONITORING

Description

The ACTIVIITY_MONITORING option specifies whether or not product activity monitoring events are generated.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	activity_monitoring *option*	√		√	√	√

Values

option is the specification for whether or not product activity monitoring events are generated.

Valid values for *option* are:

- **yes**
  Activate monitoring events.
- **no**
  Deactivate monitoring events.

[Default is no.]

3.5 ALLOC_ABNORMAL_DISP

Description

The ALLOC_ABNORMAL_DISP option is a dynamic allocation option that specifies the disposition of data set when an abnormal ending occurs.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_abnormal_disp *disposition*					√

Values

disposition is equivalent to the third positional parameter of the JCL DD statement's DISP parameter.

Valid values for *disposition* are:

- **keep**
  Keep the data set.
- **delete**
  Delete the data set.
- **catlg**
  Catalog the data set.
- **uncatlg**
  Un-catalog the data set.

[Default is delete.]

References

Refer to the IBM JCL Reference manual for complete details.

3.6 ALLOC_BLKSIZE

Description

The ALLOC_BLKSIZE option is a dynamic allocation option that specifies the block size used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_blksize *size*					√

Values

size is equivalent to the JCL DD statement's BLKSIZE parameter.

Valid values for *size* are any number (size of a block).

[Default is *27998*.]

References

Refer to the IBM JCL Reference manual for complete details.

3.7 ALLOC_DATACLAS

Description

The ALLOC_BLKSIZE option is a dynamic allocation option that specifies the SMS data class used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_dataclas *class*					√

Values

class is equivalent to the JCL DD statement's DATACLAS parameter.

Valid values for *class* are any SMS data classes defined in the local environment.

[There is no default.]

References

Refer to the IBM JCL Reference manual for complete details.

3.8 ALLOC_DIR_BLOCKS

Description

The ALLOC_DIR_BLOCKS option is a dynamic allocation option that specifies the number of directory blocks for newly allocated partitioned data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_dir_blocks *number*					√

Values

number is equivalent to the third positional parameter of the second positional parameter of the JCL DD statement's SPACE parameter.

Valid values for *number* are any number (number of directory blocks to allocate).

[Default is *20*.]

References

Refer to the IBM JCL Reference manual for complete details.

3.9 ALLOC_DSORG

Description

The ALLOC_DSORG option is a dynamic allocation option that specifies the data set organization used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_dsorg *organization*					√

Values

organization is equivalent to the JCL DD statement's DSORG parameter.

Valid values for *organization* are:

- **po**
  Partitioned organization
- **ps**
  Physically sequential

[Default is ps.]

References

Refer to the IBM JCL Reference manual for complete details.

3.10 ALLOC_INPUT_STATUS

Description

The ALLOC_INPUT_STATUS option is a dynamic allocation option that specifies the status of data sets being allocated for input.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_input_status *status*					√

Values

status is equivalent to the first positional parameter of the JCL DD statement's DISP parameter.

Valid values for *status* are:

- **old**
  Allocate the data set exclusively.
- **shr**
  Allocate the data set non-exclusively.

[Default is old.]

References

Refer to the IBM JCL Reference manual for complete details.

3.11 ALLOC_LRECL

Description

The ALLOC_LRECL option is a dynamic allocation option that specifies the logical record length used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_lrecl *length*					√

Values

length is equivalent to the first positional parameter of the JCL DD statement's LRECL parameter.

Valid values for *length* are any number (length of the record).

[Default is *1024*.]

References

Refer to the IBM JCL Reference manual for complete details.

3.12 ALLOC_MGMTCLAS

Description

The ALLOC_MGMTCLAS option is a dynamic allocation option that specifies the SMS management class used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_mgmtclas *class*					√

Values

class is equivalent to the first positional parameter of the JCL DD statement's MGMTCLAS parameter.

Valid values for *class* are any SMS management classes defined in the local environment.

[There is no default.]

References

Refer to the IBM JCL Reference manual for complete details.

3.13 ALLOC_NORMAL_DISP

Description

The ALLOC_NORMAL_DISP option is a dynamic allocation option that specifies the disposition of data set when normal ending occurs.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_normal_disp *disposition*					√

Values

disposition is equivalent to the second positional parameter of the JCL DD statement's DISP parameter.

Valid values for *disposition* are:

- **keep**
  Keep *the data set.*
- **delete**
  Delete the data set.
- **catlg**
  Catalog the data set.
- **uncatlg**
  Un-catalog the data set.

[Default is catlg.]

References

Refer to the IBM JCL Reference manual for complete details.

3.14 ALLOC_OUTPUT_STATUS

Description

The ALLOC_OUTPUT_STATUS option is a dynamic allocation option that specifies the status of existing data sets being allocated for output.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_output_status *status*					√

Values

status is equivalent to the first positional parameter of the JCL DD statement's DISP parameter.

Valid values for *status* are:

- **new**
  Create new data set.
- **shr**
  Allocate the data set non-exclusively.
- **old**
  Allocate the data set exclusively.
- **mod**
  Either create a new data set, for exclusive use, or allocate a sequential data set exclusively and add records to the end of it.

[Default is old.]

References

Refer to the IBM JCL Reference manual for complete details.

3.15 ALLOC_PRIM_SPACE

Description

The ALLOC_PRIM_SPACE option is a dynamic allocation option that specifies the primary space allocation used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_prim_space *space*					√

Values

space is equivalent to the first sub-parameter of the second sub-parameter of the JCL DD statement's SPACE parameter.

Valid values for space are any number (number of space units to allocate).

[Default is 15.]

References

Refer to the IBM JCL Reference manual for complete details.

3.16 ALLOC_RECFM

Description

The ALLOC_RECFM option is a dynamic allocation option that specifies the record format used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_recfm *format*					√

Values

format is equivalent to the JCL DD statement's RECFM parameter.

Valid values for *format* are dependent on the data set organization and access method used. The following values are valid for both partitioned and sequential data sets:

- **F**[**B**][**A**|**M**]
  Fixed, optionally blocked, and optionally either ANSI or Machine control characters.
- **V**[**B**][**A**|**M**|**S**]
  Variable, optionally blocked, and optionally either ANSI or Machine control characters, or spanned.

[Default is VB.]

References

Refer to the IBM JCL Reference manual for complete details.

3.17 ALLOC_SEC_SPACE

Description

The ALLOC_SEC_SPACE option is a dynamic allocation option that specifies the secondary space allocation used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_sec_space *space*					√

Values

space is equivalent to the second sub-parameter of the second sub-parameter of the JCL DD statement's SPACE parameter.

Valid values for *space* are any number (number of space units to allocate).

[Default is 15.]

References

Refer to the IBM JCL Reference manual for complete details.

3.18 ALLOC_SPACE_UNIT

Description

The ALLOC_SPACE_UNIT option is a dynamic allocation option that specifies the space unit in which space is allocated for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_space_unit *space*					√

Values

space is equivalent to the first sub-parameter of the JCL DD statement's SPACE parameter.

Valid values for *space* are:

- *number*
  Block length or record length
- **cyl**
  Cylinder allocation
- **trk**
  Track allocation

[Default is trk.]

References

Refer to the IBM JCL Reference manual for complete details.

3.19 ALLOC_STORCLAS

Description

The ALLOC_STORCLAS option is a dynamic allocation option that specifies the SMS storage class used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_storclas *class*					√

Values

class is equivalent to the JCL DD statement's STORCLAS parameter.

Valid values for *class* are any SMS storage classes defined in the local environment.

[There is no default.]

References

Refer to the IBM JCL Reference manual for complete details.

3.20 ALLOC_UNIT

Description

The ALLOC_UNIT option is a dynamic allocation option that specifies the unit used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_unit *unit*					√

Values

unit is equivalent to the JCL DD statement's UNIT parameter.

Valid values for *unit* are:

- *number*
  Device number
- *name*
  Unit generic or group name

[Default is SYSALLDA.]

References

Refer to the IBM JCL Reference manual for complete details.

3.21 ALLOC_VOLSER

Description

The ALLOC_VOLSER option is a dynamic allocation option that specifies the volume serial number used for newly allocated data sets.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	alloc_volser *number*					√

Values

number is equivalent to the sub-parameter SER of the JCL DD statement's VOL parameter.

Valid values for *number* are any Volume serial number.

[There is no default.]

References

Refer to the IBM JCL Reference manual for complete details.

3.22 CODE_PAGE

Description

The CODE_PAGE option specifies the character code page that is used to translate text data received and transmitted over the network.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	codepage *codepage*	√		√	√	√

Values

codepage can be any character code page.

Note: UTF-8 is not a supported codepage value for the CODE_PAGE option. UTF-8 codepage is valid only for text file data translation. Consequently, it can be specified only on the UDM open script statement.

(Table 7.13 Character Code Pages identifies the character code pages provided for UDM by Stonebranch Inc.)

Default

The default code page is different for different operating systems:

OS/400

- IBM037

UNIX

- ISO8859-1 (8-bit ASCII) ASCII-based operating systems
- IBM1047 (EBCDIC) Non-OS/400, EBCDIC-based operating system

z/OS

- ISO8859-1 (8-bit ASCII) ASCII-based operating systems
- IBM1047 (EBCDIC) Non-OS/400, EBCDIC-based operating system

UTT Files

Universal Translate Table (UTT) files are used to translate between Unicode and the local single-byte code page.

Platform	Location
OS/400	UNVNLS file in the Universal Products installation library (default is UNVPRD320). The value *codepage* is the base file name of the UTT file. UTT files are stored as members without the UTT extension.
z/OS	Members of the PDS allocated to the Broker ddname UNVNLS. The value *codepage* is a member name in the library.
UNIX	Directory specified by NLS_DIRECTORY (default is **`/opt/universal/nls`**). The value *codepage* is the base file name of the UTT file. All UTT files end with an extension of **`.utt`**.

Table 3.2 UTT File Locations

3.23 CODEPAGE_TO_CCSID_MAP

Description

CAUTION: This option is intended only for use by OS/400 specialists who fully understand code pages, CCSIDs, how the two relate to each other, and how OS/400 uses CCSIDs for data translation between data streams and files. If a code page and CCSID are not correctly matched, data corruption will occur.

The CODEPAGE_TO_CCSID_MAP option specifies whether to use the internal table or external table for code page to CCSID mapping.

An internal table provides code page to CCSID mapping for the code page specified by the open command. The mapping only occurs if the CCSID is required for text file mapping. For the LIB file system, this includes mapping text to source physical files or to data files with an associated DDS file. All files in the root and QOpenSys file systems have associated CCSIDs.

This CCSID is not the same as the CCSID attribute associated with the attrib command; the UDM CCSID attribute determines the CCSID of the target file if the file does not exist. This CCSID in the mapping table is used as the CCSID associated with the attrib command when the default value, CODEPAGE, is specified; it also identifies the data stream CCSID, allowing the operating system to translate the code page translated data stream to the file. However, data written to or read from a file, record, or field with a CCSID of 65535 (or 'HEX') will not be translated.

The code page (and the mapped CCSID) from the open command is used for data mapping between the two parties involved in a data transfer. For OS/400, the code page also is used for mapping the data stream to or from the OS/400 file, whether a LIB or HFS transfer.

Under normal circumstances, the external mapping table will not be needed. This external table replaces the internal table, so all potentially needed code page to CCSID mappings must be provided in the external table.

Universal Products for OS/400 provides an example external table in file **`CP2CCSID_X`**, in product library **`UNVPRD320`**. For UDM to use the external table, create a single member file with the name **`CDPG2CCSID`** in the installation library. The example file may be used as a template by copying it to **`CDPG2CCSID`** in the same library.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	codepage_to_ccsid_map_opt *table*	✔				

Value

table is the specification for which table to use:

- **error**
  Use the external table; if it is not found, report an error.
- **quiet**
  Use the external table; if it is not found, use the internal table. No message is issued.
- **internal**
  Use the internal table.

[Default is internal.]

3.24 DATA_COMPRESSION

Description

The DATA_COMPRESSION option specifies whether or not data standard I/O file transmissions across the network should be compressed. Optionally, it also can specify the compression method to use.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	compress *option*[,*method*]	√		√	√	√

Values

option is either of the following values:

- **yes**
  Data compression is required. All data in standard I/O file transmissions is compressed regardless of the UDM Manager DATA_COMPRESSION option value.
- **no**
  Data compression is not required. However, data compression still can be requested via the UDM Manager DATA_COMPRESSION option.

[Default is no.]

method is either of the following values:

- **zlib**
  Data is compressed using ZLIB compression algorithm. This method usually results in a very high compression rate, but tends to be somewhat CPU-intensive. It is recommended in environments where controlling a process's CPU usage is not necessarily a priority.
- **hasp**
  Data is compressed using the HASP compression algorithm. This method is less CPU-intensive than the ZLIB method. It is recommended in environments where controlling CPU usage is a priority. With HASP, the compression rate, while still very good, tends to be a little less than what is possible with the ZLIB.

[Default is zlib.]

3.25 DATA_SSL_CIPHER_LIST

Description

The DATA_SSL_CIPHER_LIST option specifies the acceptable and preferred SSL cipher suites to use for the data session on which file data is transferred between UDM primary and secondary servers.

The SSL protocol uses the cipher suites to specify which encryption and message authentication (or message digest) algorithms to use.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	data_ssl_cipher_list *list*	√		√	√	√

Values

list is a comma-separated list of SSL cipher suites. The cipher suites should be listed with the most preferred cipher suite first and the least preferred cipher suite last.

Table 7.15 SSL Cipher Suites for UDM identifies the list of SSL cipher suites provided for UDM by Stonebranch Inc.

Note: In order to establish a transfer session without using SSL for the data session, the **NULL-NULL** cipher must be specified in the cipher list for any UDM Server involved in the session and in the encrypt option of the open command.

Default

The default list of SSL cipher suites is:

RC4-SHA,RC4-MD5,AES256-SHA,AES128-SHA,DES-CBC3-SHA, DES-CBC-SHA,NULL-SHA,NULL-MD5

3.26 EVENT_GENERATION

Description

The EVENT_GENERATION option specifies which events are to be generated and processed as persistent events.

A persistent event record is saved in a Universal Enterprise Controller (UEC) database for long-term storage.

(For a list of all event types for all Universal Products components, see the Universal Event Subsystem 3.2.0 Event Definitions Guide.)

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	event_generation *types*	√		√	√	√

Values

type specifies a comma-separated list of event types. It allows for all or a subset of all potential event message types to be selected.

Event type ranges can be specified by separating the lower and upper range values with a dash (**-**) character.

Event types can be selected for inclusion or exclusion:

- Inclusion operator is an asterisk (*****).
- Exclusion operator is **X** or **x**.

Examples

- 100,101,102
  Generate event types 100, 101, and 102.
- 100-102
  Generate event types 100 through 102.
- 100-102,200
  Generate event types 100 through 102 and 200.
- *
  Generate all event types.
- *,X100
  Generate all event types except for 100.
- x*
  Generate no event types.
- *,X200-250,X300
  Generate all event types except for 200 through 250 and 300.

[Default is *X (no event types).]**

3.27 FRAME_INTERVAL

Description

The FRAME_INTERVAL options sets the number of UDM transfer blocks transferred before a frame-sync message is sent when UDM is operating with network fault tolerance on (see Section 2.43 NETWORK_FAULT_TOLERANT).

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	frame_interval *number*	√		√	√	√

Values

number can be any number.

[Default is *100*.]

Note: This value should not be changed without direction from Stonebranch, Inc. Customer Support. Changing this value could degrade UDM performance.

3.28 INSTALLATION_DIRECTORY

Description

The INSTALLATION_DIRECTORY option specifies the directory in which Universal Data Mover is installed.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	installation_directory *directory*			√	√	

Values

directory is any directory.

Defaults

UNIX

[Default is `/opt/universal/udmsrv`.]

Windows

[Default is the UDM Server installation file: `c:\Program Files\Universal\udmsrv`.]

3.29 LOGON_METHOD

Description

The LOGON_METHOD option specifies the user's log on method.

If the UCMD Server is configured for user security (see the USER_SECURITY option), the log on method determines how the user is logged onto the Windows system.

If security is inactive, LOGON_METHOD is ignored.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	logon *option*				√	
Manager Override	n/a					

Values

option is the user's log on method.

Valid values for *option* are:

- **batch**
  Windows log on type is **`batch`**. A batch log on prevents the user account from interacting with the desktop. The user ID logging on as a batch user requires the Windows User Right "Log on as a batch job." If the user does not have this right, the user authentication will fail.
- **interactive**
  Windows log on type is **`interactive`**. An interactive log on permits the user account to interact with the desktop. A user account must have the "Allow log on locally" privilege granted to it to successfully authenticate with an interactive logon.

[Default is interactive.]

3.30 MESSAGE_LEVEL

Description

The MESSAGE_LEVEL option specifies the level of messages to write.

It also specifies, optionally, whether or not to write a date and time stamp with each message.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	message_level *level*[,*time*]	√		√	√	√

Values

level indicates either of the following level of messages:

- **trace**
  Activates tracing and generates a trace file to which UDM writes trace messages used for debugging.

  Note: Use **trace** only as directed by Stonebranch, Inc. Customer Support.
- **audit**
  Writes audit, informational, warning, and error messages.
- **info**
  Writes informational, warning, and error messages.
- **warn**
  Writes warning and error messages.
- **error**
  Writes error messages only.

[Default is info.]

time specifies either of the following:

- **time**
  Include a time and date stamp on each message.
- **notime**
  Do not include a time and date stamp on each message.

[Default is time.]

Trace Files

UNIX

Trace file name is **udmsrv-N-S.trc** where:

- N = Component ID assigned to this instance of the Server,
- S = Sequence number.

The trace file is created in the trace directory, as specified by the TRACE_DIRECTORY option (default is **/var/opt/universal/trace**).

z/OS

There are two possible destinations of the trace data:

1. If ddname **UNVTRMDL** is defined in the UBROKER started task procedure, a sequential data set is created using the data set allocated to **UNVTRMDL** as a model. The dynamically allocated trace data set name is **#HLQ.UDM.Dyymmdd.Thhmmss.Cnnnnnnn** where:
   - **#HLQ** is the data set name allocated on the **UNVTRMDL** ddname
   - **yymmdd** is the year, month, day
   - **hhmmss** is the hour, minute, second the data set was allocated
   - **nnnnnnn** is the last seven digits of the Server's component ID in hexadecimal format
2. If ddname **UNVTRMDL** is not defined in the UBROKER started task procedure, member name **Dnnnnnns** is created in the PDS or PDS/E allocated to the **UNVTRACE** ddname, where, **nnnnnn** is the last six digits of the Server's component ID in hexadecimal format, and **s** is the component ID's sequence number (from 0 - F). Each time a server is restarted, its sequence number is incremented. If a server is restarted more than 15 times, tracing is disabled.

Depending on the error condition being diagnosed, it is possible that the member name of the **UNVTRACE** PDS or PDS/E is not created. If this occurs, the **UNVTRMDL** ddname must be used to create a sequential data set name.

The records written to PDS and PDS/E members cannot be wrapped, so the TRACE_FILE_LINES limit has no effect on the maximum number of trace records written to the member.

OS/400

Trace file name is ***CURLIB/UNVTRCUDMS(Txxxyyyyyy)** where:

- **xxx** = first three digits of the process ID, in hexadecimal, assigned to this instance of the Server by OS/400.
- **yyyyyy** = last six digits of the component ID, in hexadecimal, assigned to this instance of the Server by the Broker.

3. These values are used to generate a unique identifier for each invocation of the UDMSRV program.
4. The default library for TRACE is the current library (***CURLIB**) of the UBROKER process, not the user process. The UBROKER current library is the temporary library designated during the Universal Products installation process; by default, the UBROKER current library is **UNVTMP320**.

3.31 NETWORK_DELAY

Description

The NETWORK_DELAY option sets the expected network latency (in seconds).

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	network_delay *number*	√		√	√	√

Values

number is any number.

[Default is *120*.]

3.32 NLS_DIRECTORY

Description

The NLS_DIRECTORY option specifies the directory name where the UDM Server can find its message catalog and code page tables.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	nls_directory *directory*			√	√	

Values

directory can be any directory.

Full path names are recommended.

Relative path names are relative to the **`universal`** installation directory.

UNIX

[Default is **`/opt/universal/nls`**.]

Windows

[Default is **`..\nls`**.]

3.33 OUTBOUND_IP

Description

The OUTBOUND_IP option sets the host or IP address that UDM binds to when initiating outgoing connections to another UDM server.

By default, this configuration option is not set.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	outbound_ip *host*	√		√	√	√

Values

host is the required IP address.

[There is no default.]

3.34 RECONNECT_RETRY_COUNT

Description

The RECONNECT_RETRY_COUNT option sets the number of attempts that the primary transfer server will make in order to re-establish a transfer session with the secondary transfer server in a three-party transfer session when a network fault occurs.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	reconnect_retry_count *number*	√		√	√	√

Values

number is any number.

[Default is *20*.]

3.35 RECONNECT_RETRY_INTERVAL

Description

The RECONNECT_RETRY_INTERVAL option sets the number of seconds that UDM will wait between each successive attempt to reestablish a transfer session when a network fault occurs.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	reconnect_retry_interval *number*	√		√	√	√

Values

number is any number (of seconds).

[Default is *60*.]

3.36 RECV_BUFFER_SIZE

Description

The RECV_BUFFER_SIZE option sets the size (in bytes) of the TCP receive buffer for UDM.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	recv_buffer_size *size*	√		√	√	√

Values

size is the number of bytes.

[Default is *0*.]

(The default, **0,** instructs UDM to use the operating system default.)

Note: The size of the TCP receive buffer should be changed only when performance tweaking is necessary. Changing this value could affect performance adversely.

3.37 SEND_BUFFER_SIZE

Description

The SEND_BUFFER_SIZE option sets the size (in bytes) of the TCP send buffer for UDM.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	send_buffer_size *size*	√		√	√	√

Values

size is the number of bytes.

[Default is *0*.]

(The default, **0,** instructs UDM to use the operating system default.)

Note: The size of the TCP send buffer should be changed only when performance tweaking is necessary. Changing this value could affect performance adversely.

3.38 SIZE_ATTRIB

Description

The SIZE_ATTRIB option sets the default size (number of records) for file creation of physical files for both data and source file types.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	size_attrib *size*	√				

Values

size is the number of records.

Other than the leading * for *NOMAX, any valid value can be entered for file size.

The values for Initial Number of Records, Increment Number of Records, and Maximum increments are comma-separated, not space-separated.

Examples:

- 1000,100,100
- 10000,499
- 50000

To specify *NOMAX, enter 'NOMAX'.

UDM uses this value as the default file SIZE attribute. (See Table 14.2 OS/400-Specific LIB File Attributes for Creating New Files in the Universal Data Mover 3.2.0 User Guide for details.)

[Default is *empty string*.]

3.39 TCP_NO_DELAY

Description

The TCP_NO_DELAY option specifies whether or not to use TCP packet coalescing.

The packet coalescing algorithm, which can delay the sending of small amounts of data over the network, is designed to improve network congestion. However, since it can have a significantly negative effect on the performance of UDM, TCP_NO_DELAY specifies – by default – not to use TCP packet coalescing.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	tcp_no_delay *option*			√	√	√

Values

option is the specification for whether or not to use TCP packet coalescing.

Valid values for *option* are:

- **yes**
  Do not use TCP packet coalescing.
- **no**
  Use TCP packet coalescing.

[Default is yes.]

3.40 TMP_DIRECTORY

Description

The TMP_DIRECTORY option specifies the name of the directory that the UDM Server uses for temporary files.

OS/400

Do not include this directory in any system or backup that requires an exclusive lock on the directory while UDM is running.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	tmp_directory *directory*	√		√	√	√

Values

directory is the name of the directory.

It should specify a fully qualified path name.

Defaults

UNIX

Default is */var/opt/universal/tmp*.

Windows

[Default is *..\tmp*.

z/OS

Default is */tmp*.

OS/400

Default is */tmp*.

3.41 TRACE_DIRECTORY

Description

The TRACE_DIRECTORY option specifies the directory name that the UDM Server uses for its trace files.

Relative path names are relative to the UDM Server installation directory. Full path names are recommended.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	trace_directory *directory*			√	√	

Values

directory is the name of the directory.

UNIX

[Default is `/var/opt/universal/trace`.]

Windows

[Default is `c:\program\files\universal\udmsrv`.]

3.42 TRACE_FILE_LINES

Description

The TRACE_FILE_LINES option specifies the maximum number of lines to write to the trace file.

When the maximum number of lines has been reached, the trace file will wrap around and start writing trace entries after the trace header lines.

Tracing is activated, and a trace file is generated, when the MESSAGE_LEVEL option is set to TRACE.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	trace_file_lines *number*	√		√	√	√

Values

z/OS

The average size of a trace file line is 50 characters.

The trace file wrapping is only supported with sequential data sets that have a fixed record format. Partitioned data sets or variable record formats are not supported.

[Default is *200,000*.]

UNIX

The average size of a trace file line is 50 characters.

The default is a very large value of `200,000`. If space is limited in the trace file directory (specified with the TRACE_DIRECTORY option), set this to a smaller value.

OS/400

The current record length is 384 characters.

As allocated on the AS/400 (i5), the maximum number of records is 509,000. Increasing TRACE_FILE_LINES beyond this value requires manual adjustment of the file size via the CHGPF command.

Note:

1. The file is not created until the first time Universal Data Mover Server trace is used.
2. Deleting the trace file will reset the maximum number of records to 509,000.
3. To clear the trace file without resetting the maximum number of records the file may contain, use the RMVM FILE(UNVTMP320/UNVTRCUDM) MBR(*ALL) command. Substitute the default library designated during product installation if different from UNVTMP320.
4. WARNING: Setting this number of records larger than the maximum number of records allocated to the trace file will result in a prompt sent to QSYSOPR. Unless a default response is set for this message (CPA5305), the PROCESS WILL HANG waiting for the QSYSOPR response.

[Default is *200,000*.]

If space is limited in the ASP where the trace file is located, set this to a smaller value. To avoid trace file wrapping, set this to a larger value. Please read the **`trace_file_lines`** information in the UNVCONF(UDMS) file member before increasing this value.

3.43 TRACE_TABLE

Description

The TRACE_TABLE option specifies the size of a wrap-around trace table maintained in memory.

The trace table is written to a file / data set when the program ends under the conditions specified by value *cond*. Tracing is activated, and a trace file is generated, when the MESSAGE_LEVEL option is set to TRACE.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	trace_table *size,condition*	√		√	√	√

Values

size is the size (in bytes) of the table.

The size can be suffixed with either of the following characters:

- **M** indicates that the size is specified in megabytes
- **K** indicates that the size is specified in kilobytes

For example, **50M** indicates that 50 X 1,048,576 bytes of memory is allocated for the trace table.

Note: A value of **0** indicates that the trace table is not used.

condition is the condition under which the trace table is printed.

Possible values for *condition* are:

- **error**
  Write the trace table if the program ends with a non-zero exit code.
- **always**
  Write the trace table when the program ends regardless of the exit code.
- **never**
  Never write the trace table.

3.44 UMASK

Description

The UMASK option specifies the file mode creation mask. It affects the file permission bits of newly created files.

All files are created with a permission mode of 666, which is read-write permission for the owner, group, and other permission categories. UDM uses UMASK to turn off selected permission bits by subtracting the value of UMASK from mode 666.

Note: UMASK is supported only for the Hierarchical File System (HFS).

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	umask *number*	√		√		√

Values

number can be any number, 001 to 666.

[Default is *026*.]

The default value (**026**) results in file permission 640 (666 - 026 = 640), which is:

- read-write for the owner
- read for the group
- none for others

References

Refer to the UNIX man page umask(1) for complete details.

3.45 USER_SECURITY

Description

The USER_SECURITY option specifies the user security option.

If user security is activated, the UDM Server logs the user onto the system, and the command is run with the user's identity. If user security is not activated, the command runs with the same identity as the Server.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Configuration File Keyword	security *option*	√		√	√	√

Values

option is either of the following values:

z/OS

- **default**
  Use z/OS SAF user authentication. User ID must have OMVS segment.
- **none**
  No user security

UNIX

- **default**
  Use UNIX default user authentication method, **`/etc/passwd`**. **[default]**
- **none**
  No user security.
- **pam**
  Use the pluggable Authentication Modules (PAM) interface.
- **trusted**
  User HP Trust Security authentication.

OS/400

- **default**
  Use OS/400 security.
- **none**
  No user security

Chapter 4
Universal Data Mover Component Definition Options

4.1 Overview

This chapter provides detailed information about the options that comprise Universal Data Mover (UDM) component definitions.

The options are listed alphabetically, without regard to any specific operating system.

Information on how component definitions are used is documented in the Universal Data Mover 3.2.0 User Guide.

Section 4.2 Component Definition Information provides a guideline for understanding the information presented for each component definition option.

4.2 Component Definition Information

For each component definition option, this chapter provides the following information.

Description

Describes the option and how it is used.

Usage

Provides a table of the following information:

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Component Definition Keyword	<Format / Value>					

Method

Identifies the method used for specifying a Universal Data Mover component definition option:

- Component Definition Keyword

Syntax

Identifies the syntax of the method used to specify the option:

- Format Specific characters that identify the option.
- Value Type of value(s) to be supplied for this method.

(Operating System)

Identifies (with a ✔) the operating systems for which the method of specifying the option is valid:

- OS/400
- HP NonStop
- UNIX
- Windows
- z/OS

Values

Identifies all possible values for the specified value type.

Defaults are identified in **[bracketed bold type]**.

4.3 Component Definition Options List

Table 4.1 identifies all of the options that can comprise a Universal Data Mover component definition.

Component	Description	Page
AUTOMATICALLY_START	Specification for whether or not UDM Server starts automatically when Universal Broker is started.	164
COMPONENT_NAME	Name by which the clients know the UDM Server.	165
CONFIGURATION_FILE *	Name of the UDM Server configuration file.	166
RUNNING_MAXIMUM	Maximum number of UDM Servers that can run simultaneously.	167
START_COMMAND *	Program name of the UDM Server.	168
WORKING_DIRECTORY *	Directory used as the working directory of the UDM Server.	169
* These options are required in all component definitions.		

Table 4.1 Universal Data Mover Component Definition Options

4.4 AUTOMATICALLY_START

Description

The AUTOMATICALLY_START option specifies whether or not the UDM Server starts automatically when Universal Broker is started.

Note: AUTOMATICALLY_START is optional in a component definition.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Component Definition Keyword	auto_start *option*	√	√	√	√	√

Values

option is the specification for how the UDM Server is started.

The only valid value for *option* is:

- **no**
  UDM Server is not started automatically when Universal Broker is started. It is started only on demand.

4.5 COMPONENT_NAME

Description

The COMPONENT_NAME option specifies the name of the UDM Server.

Component start requests refer to UDM Server by this name.

Note: COMPONENT_NAME is optional in a component definition. If it is not specified, the file name is used as the component name.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Component Definition Keyword	component_name *name*	√	√	√	√	√

Values

name is the name of the UDM Server.

There is only one valid value for *name*: **udm**. (This is the name of the UDM Server component definitions file / member.)

Note: This name should not be changed.

4.6 CONFIGURATION_FILE

Description

The CONFIGURATION_FILE option specifies the name of the UDM Server configuration file.

Note: CONFIGURATION_FILE is required in a component definition.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Component Definition Keyword	configuration_file *member* **or** configuration_file *filename*	√	√	√	√	√

Values

member / *filename* is the name of the configuration member / file.

OS/400

Configuration file name can be any valid file name. It can be edited manually with SEU, EDTF, or any other installed source file editor. The default file name is **UNVPRD320/UNVCONF(UDMS)**.

HP NonStop

Full path name of the configuration file. The file name can be any valid file name. The installation default is **$SYSTEM.UNVCONF.UDMSCFG**.

UNIX

Full path name of the configuration file. The file name can be any valid file name. The installation default is **/etc/universal/udms.conf**.

Windows

Full path name of the configuration file. The file name can be any valid file name. The installation default is **c:\Documents and Settings\All Users\Application Data\Universal\conf\udms.conf**.

z/OS

Member name of the component configuration file in the **UNVCONF** library allocated to the Universal Broker ddname **UNVCONF**. The installation default is **UDSCFG00**.

4.7 RUNNING_MAXIMUM

Description

The RUNNING_MAXIMUM option specifies the maximum number of UDM Servers that can run simultaneously.

If this maximum number is reached, any command received to start the component is rejected.

Note: RUNNING_MAXIMUM is optional in a component definition.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Component Definition Keyword	running_max *max*	√	√	√	√	√

Values

max is the maximum number of UDM Servers that can run simultaneously.

[Default is *100*.]

4.8 START_COMMAND

Description

The START_COMMAND option specifies the full path name (member name for z/OS) of the Universal Data Mover Server program.

Optionally, START_COMMAND also can specify command line options.

Note: START_COMMAND is required in a component definition.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Component Definition Keyword	start_command *member* **or** start_command *name[options]*	√	√	√	√	√

Values

member / *name* is the program name of UDM Server.

options is the optional list of command line options.

z/OS

name is the program object of the UDM Server. The program object must be in the Universal Broker's search order for loading program objects. The default location is the **SUNVLOAD** library allocated to the Universal Broker's **STEPLIB** ddname.

HP NonStop and UNIX

name is the full path name of the UDM Server program.

Windows

name is the full path name of the UDM Server program.

OS/400

name is the fully qualified name of the UDM Server program. The default is ***LIBL/UDMSRV**.

4.9 WORKING_DIRECTORY

Description

The WORKING_DIRECTORY option specifies the full path name of the directory used as the working directory of UDM Server.

Note: WORKING_DIRECTORY is required in a component definition.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
Component Definition Keyword	working_directory *directory*	√	√	√	√	√

Values

directory is the full path name of the working directory.

[Default is (.).

HP NonStop, UNIX, Windows

directory is the full path name of the directory UDM Server uses as its working directory.

z/OS

directory is the HFS directory name that the UDM Server uses as its working directory.

OS/400

working_directory serves as a required placeholder only.

Note: Do not change this directory.

Chapter 5
Universal Data Mover UACL Entries

5.1 Overview

This chapter provides detailed information on the Universal Access Control List (UACL) entries available for use with Universal Data Mover.

The UACL entries are listed alphabetically, without regard to any specific operating system.

Information on how these UACL entries are used is documented in the Universal Data Mover 3.2.0 User Guide.

Section 5.2 UACL Entries Information provides a guideline for understanding the information presented for each UACL entry.

5.2 UACL Entries Information

For each UACL entry, this chapter provides the following information.

Description

Describes the UACL entry and how it is used.

Usage

Provides a table of the following information:

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
UACL File Keyword	<Type / Rule>					

Method

Identifies the method used for specifying a UACL entry:

- UACL FIle Keyword

Syntax

Identifies the syntax of the method used for a UACL entry:

- Type Universal Products component to which the rule applies.
- Rule Client's identity, request to which the entry pertains, and security attributes that the entry enforces.

(Operating System)

Identifies (with a ✔) the operating systems for which the method of specifying the UACL entry is valid:

- OS/400
- HP NonStop
- UNIX
- Windows
- z/OS

Values

Identifies all possible values for the fields in a UACL entry rule.

Defaults are identified in **[bracketed bold type]**.

5.3 UACL Entries List

Table 5.1 identifies all Universal Data Mover UACL entries.

UACL Entry	Description	Page
UDM_ACCESS	Allows or denies access to Universal Data Mover Server services. There are two forms to this entry: • **udm_access** • **udm_cert_access**	174
UDM_MGR_ACCESS	Allows or denies access based on the host name and/or user of the Manager trying to initiate a UDM session.	176

Table 5.1 Universal Data Mover UACL Entries

5.4 UDM_ACCESS

Description

A UDM_ACCESS UACL entry either allows or denies access to Universal Data Mover Server services.

If access is permitted, UDM_ACCESS also specifies whether or not user authentication is required.

There are two forms of the UDM_ACCESS entry based on the client identification method:

- **`udm_access`** form is for IP-based client identification.
- **`udm_cert_access`** is for X.509 certificate-based client identification.

A **`udm_access`** UACL entry is matched if all of the following occur:

- Request comes from an IP address identified by *host*.
- Remote end is executing as user *remote_user*.
- Remote user is requesting to execute a command as local user *local_user*.

A **`udm_cert_access`** UACL entry is matched if both of the following occur:

- Request comes from a client with a certificate identifier of *certid*.
- Remote user is requesting to execute a command as local user *local_user*.

The first matching rule is used to control access.

See Section 2.8.2 UACL Entries in the Universal Data Mover 3.2.0 User Guide for details on *host*, *remote_user*, and *local_user* specification syntax.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
UACL File Keyword	udm_access *host,remote_user,local_user,access, auth* udm_cert_access * *certid,local_user,access,auth*	√		√	√	√
* udm_cert_access is not a valid form of UDM_ACCESS for OS/400.						

Values

Valid values for *access* are:

- **deny**
  Service is denied. A message is returned to the remote end. The connection is closed.
- **allow**
  Service is accepted and processed.

Valid values for *auth* are:

- **auth**
  Local user account must be authenticated. The Manager must provide a proper password for the account.
- **noauth**
  Local user account does not require user authentication.

  Note: **noauth** should be used with care. Turning off user authentication may violate your local security policies on the Server system.

5.5 UDM_MGR_ACCESS

Description

A UDM_MGR_ACCESS UACL entry either allows or denies access to Universal Data Mover Server services based on the host name and/or user of the Manager trying to initiate a UDM session.

If access is permitted, UDM_MGR_ACCESS also specifies whether or not user authentication is required.

A **`udm_mgr_access`** UACL entry is matched if all of the following occur:

- Request comes from a Manager initiated on host name identified by host. This is the machine host name, which may or may not be equivalent to the host DNS name.
- Manager is executing as user *manager_user*.
- Manager is requesting to execute a command as local user *local_user*.

The first matching rule is used to control access.

See Section 2.8.2 UACL Entries in the Universal Data Mover 3.2.0 User Guide for details on *host*, *manager_user*, and *local_user* specification syntax.

Usage

Method	Syntax	OS/400	NonStop	UNIX	Windows	z/OS
UACL File Keyword	udm_mgr_access *manager_host,manager_user, local_user,access,auth*	√		√	√	√

Values

Valid values for *access* are:

- **deny**
  Service is denied. A message is returned to the remote end. The connection is closed.
- **allow**
  Service is accepted and processed.

Valid values for *auth* are:

- **auth**
  Local user account must be authenticated. The Manager must provide a proper password for the account.
- **noauth**
  Local user account does not require user authentication.

  Note: **noauth** should be used with care. Turning off user authentication may violate your local security policies on the Server system.

Chapter 6
UDM Commands

6.1 Overview

This chapter provides detailed information on UDM commands.

6.1.1 UDM Commands List

Table 6.1 identifies all of the UDM commands.

Detailed information on each command is provided on the identified page.

Name	Description	Page
appenddata	Appends a line of text to the end of an existing data element, or creates a new data element containing that line of text.	181
attrib	Sets the file system attributes that govern the transfer operations on the host with the specified logical name.	182
break	Stops iterating through a **`forfiles`** loop and picks up execution at the script line immediately following the end statement marking the end of the **`forfiles`** loop.	187
call	Loads and executes a command script.	188
cd	Changes the working directory (on UNIX, Windows, OS/400, and file system HFS) or if z/OS, the current data set qualifiers (for DSN and DD file systems) on the specified logical machine to the specified path.	190
close	Closes the current transfer session.	191
closelog	Closes the open log file.	192
compare	Compares two strings of data	193
copy	Initiates a copy operation.	194
copydir	Initiates a copy operation that recurses into subdirectories.	196

Name	Description	Page
data	Defines an in-stream data element that can be passed as input for other commands.	198
debug	Turns debug information on and off.	199
delete	Deletes a file (or series of files if file-spec contains any wildcards) from the transfer server with the corresponding logical name.	200
deletestring	Removes a substring from an existing string.	201
echo	Sends text to standard out (stdout).	203
echolog	Sends text to an open log file.	204
exec	Executes system commands on remote machines.	206
execsap	Executes SAP events.	210
exit	Exits the UDM Manager (same as the **`quit`** command).	213
filesys	Sets the file system with which the server with the specified logical name is working.	214
filetype	Set a series of masks and corresponding transfer mode types.	216
find	Finds a specific occurrence of a substring in an existing string or list element.	218
format	Creates a formatted string.	220
insertstring	Inserts a substring into an existing string.	222
loaddata	Loads the contents of a data element from a file, instead of setting them in a script via the data command.	224
logdata	Writes the content of a data element to the open log file.	225
lower	Forces all alpha characters in a given variable or list element to lower case.	226
mode	Sets the current transfer mode.	227
move	Initiates a move operation.	228
open	Opens a UDM session.	230
openlog	Opens a log file on disk for writing custom log information.	235
pad	Takes a string in an existing variable or list element and pads it to make it the given length.	237
parse	Parses a string, placing the components of the string into variables.	239
print	Prints a message in the UDM manager's transaction output.	241
query	Prints out the UDM Manager version.	242
quit	Exits the UDM Manager (same as the **`exit`** command).	243
rename	Renames a file.	244
replace	Replaces one or more instances of a sequence with another sequence.	245
report	Sets UDM's reporting options.	247
resetattribs	Resets the attributes for all UDM file systems on the transfer server with the specified logical name.	248
return	Stops executing the current script immediately and returns execution to the calling script immediately after the **`call`** command used to invoke the current script.	249
reverse	Reverses the order of all characters in the string of a specified existing variable or element.	250
savedata	Writes each line of a data element to a file on disk.	251

Name	Description	Page
set	Sets the UDM Manager's built-in and global variable values.	253
status	Displays the current connection status.	254
strip	Strips occurrences of a sequence from a string.	255
substring	Finds a substring in an existing string and stores it in a variable.	257
truncate	Truncates a string to a specific length.	259
upper	Forces all alpha characters in a given variable or list element to upper case.	260

Table 6.1 UDM Commands

6.2 appenddata

Syntax

appenddata *data_element_name* [*value_1*]... [*value_n*]

Description

The **appenddata** command appends a line of text to the end of an existing data element or, if that data element does not exist, creates a data element containing that line of text.

The *data_element_name* parameter specifies the name of a data element.

The *value_* parameters, which specify the text to be appended, are concatenated.

Variable references and expressions in these parameters are resolved, as with any other command, before assembling the line (with no spaces between each value) and appending it to the data element.

Parameters

Parameter	Description
data_element_name	Name of a data element.
value_1	First value in the line of text to be appended.
value_n	Last value in the line of text to be appended.

Table 6.2 **appenddata** Command Parameters

Examples

To append **The answer to 1 + 1 is 2** to the data element **mydata**:

```
appenddata mydata "The answer to 1 + 1 is" <1 + 1>
```

To append **SingleSystemImage** to the data element **mydata**:

```
appenddata mydata Single System Image
```

6.3 attrib

Syntax

```
attrib   logical-name[={dd|dsn|hfs|lib}]
         [attribute-name=[attribute_value]]...
```

Description

The **attrib** command sets the file system attributes that govern the transfer operations on the host with the specified logical name.

If only a logical name is specified in the **attrib** command, the current set of attributes for the specified host is displayed. For systems that support multiple file systems, such as z/OS and OS/400, the logical name can be followed by an file system name that indicates the files system to which the attribute applies:

- z/OS hfs, dsn, dd
- OS/400 hfs, lib

If no file system name is specified, the attributes will be applied for the currently selected file system. If no attributes are specified, the transfer server returns its current set of attributes and their values.

Parameters

Parameter	Description
logical-name	Logical name of the transfer server for which to set the attributes (or from which to retrieve the attributes).
dd|**dsn**|**hfs**|**lib**	File system for which the attribute is to be set: • Values **dd** and **dsn** are valid only on z/OS file systems. • Value **hfs** is valid only on z/OS and OS/400 file systems. • Value **lib** is valid only on OS/400file systems. If no file system is specified, the attribute is set for the current file system on the specified server.
attribute-name	Name of an attribute.
attribute-value	Value to be set for the attribute.

Table 6.3 **attrib** Command Parameters

Common File System Attributes

The following attributes are common to UDM on most platforms:

Attribute Name	Values	Description
createop	**append**, **new**, or **replace**	Specification for how the file is to be created: • If value is **append**, the transferred data is appended to any data already in an existing destination file. If the destination file does not exist, it is created. • If value is **new**, the UDM copies the file only if the destination file does not already exist. If the destination file does exist at the time the copy operation is initiated, the operation returns with an error. • If value is **replace**, UDM overwrites an existing destination file. Otherwise, UDM creates the file. [Default is **new**.]
defext	Any sequence of characters valid for the destination file system.	Sequence of characters appended to the end of the filename used to write the destination file, if the source filename is being used implicitly as the destination filename. This occurs after the file extension has been truncated (if **truncext** is set to **yes**). [By default, no default extension is defined.] Note: The sequence of characters is appended verbatim. UDM does not add a dot character before the sequence, so if one is desired, it must be specified explicitly.
eol	Any sequence of valid text data	End-of-line sequence used in text transfers. For the source side of a copy operation, excepting those from the z/OS **dd** and **dsn** file systems and the OS/400 **lib** file system, the end-of-line sequence is used to determine the end of each line of data. When the specified sequence occurs in the data, UDM considers all data read up to that point (starting from the previous line) as a single line. Each line is transferred without the end-of-line sequence. On the destination side of a text transfer, the end-of-line sequence is appended to the end of each line before it is written. Two special character sequences can be used in any end of line sequence: • • **\r** sequence indicates a carriage return. • • **\n** sequence indicates a line feed. [Default depends on the platform and the file system: • Under Windows, the default is **\r\n**. • For UNIX platforms and the HFS file system under USS, the default value is \n. • Under z/OS (for the **dd** and **dsn** file systems) and OS/400 (for the **lib** file system), the **eol** attribute is undefined. • Under OS/400 (for the **hfs** file system), the default is **FILE**, which makes end-of-line terminator consistent with file **ccsid**.]

Attribute Name	Values	Description
linelen	A positive integer	Maximum length of each line of data (record under the z/OS **dd** and **dsn** file systems) written. It applies only to the destination side of a transfer and is used in conjunction with the **lineop** and **padline** attributes. [Default is 0.] • Under Windows, UNIX, and the **hfs** file system, this means no line operation takes place. • Under z/OS (for the **dd** and **dsn** file systems), the value **linelen** will be set equal to the logical record size used in allocating the destination file.
lineop	**none**, **stream**, **wrap**, or **trunc**	Line operation for transferred lines (records under the z/OS **dd** and **dsn** file systems). For the line operation to be in play in the transfer, the value of **linelen** must not be zero (**linelen** is set automatically for the z/OS **dd** and **dsn** file systems to the logical record size if it is zero). • If value is **none**, each source line or record or data is written as it is received as a complete line or record. If the length of the source line is greater than that specified by **linelen**, UDM issues an error and the transfer operation is aborted. • If value is **stream**, the source data is treated as one long, single line of data. The source data is broken into multiple lines (records), each with a length of that specified by **linelen**. • If value is **trunc**, each source line longer than the value specified by **linelen** is truncated to be exactly **linelen** characters long. • If value is **wrap**, each source line longer than the value specified by **linelen** is broken up into multiple lines, each no longer than **linelen** characters long. Each segment is written out as a separate line (record). Note: If an end of line sequence is specified, the length of the sequence is not considered by UDM when determining the length of a line on the destination side. UDM only looks at the raw data that is transferred. [By default, the **lineop** attribute is not defined.]
mode	Set of three numbers (0-7) or nothing	Note: This attribute is used only for UDM for UNIX. Specification for the mode (in UNIX parlance), or file permissions, of a file created by UDM in a copy operation. Existing files do not have their modes modified by UDM. They retain the file mode that they had before the copy operation was initiated. Each number in the set corresponds to one or more individuals for whom access is granted for the file: • First number Owner of the file. • Second number Users in the group assigned to the file. • Third number Everyone else. The value of each number is the sum of values representing file permissions: • 0 No permissions. • 1 Permission to execute the file. • 2 Permission to write to the file. • 4 Permission to read from the file. [By default, the mode attribute is not set. The default mode of a newly created file by UDM is dependent upon the user's umask or the mode of the source file in a UDM transfer.]

Attribute Name	Values	Description
padline	**none**, **null**, or **space**	Specification for whether or not data is padded. (Used in conjunction with **linelen**, when **linelen** is not zero.) • If value is **none**, each line (record) of data written is not padded. • If value is **null**, each line of data is padded with null characters (hex value 0) at the end until the line is **linelen** characters in length. • If value is **space**, each line of data is padded with spaces at the end until the line is **linelen** characters in length. [Default is **none**.]
srccreatetime	**yes** or **no**	Note: This attribute is used only for UDM for Windows. Specification for whether or not the creation timestamp of the destination file in a copy operation matches the creation timestamp of the source file. • If value is **yes**, the creation timestamp of the destination file matches the creation timestamp of the source file. • If the value is **no**, the creation timestamp of the destination file does not match the creation timestamp of the source file. [Default is **no**.]
srcmodtime	**yes** or **no**	Note: This attribute is used only for UDM for UNIX and Windows. Specification for whether or not the modification timestamp of the destination file in a copy operation matches the modification timestamp of the source file. • If value is **yes**, the modification timestamp of the destination file matches the modification timestamp of the source file. • If the value is **no**, the modification timestamp of the destination file does not match the modification timestamp of the source file. [Default is **no**.]
srcaccesstime	**yes** or **no**	Note: This attribute is used only for UDM for UNIX and Windows. Specification for whether or not the last access timestamp of the destination file in a copy operation matches the last access timestamp of the source file prior to the copy operation. • If value is **yes**, the last access timestamp of the destination file matches the last access timestamp of the source file prior to the copy operation. • If the value is **no**, the last access timestamp of the destination file does not match the last access timestamp of the source file prior to the copy operation. [Default is **no**.]
trans	**yes** or **no**	Specification for whether or not a transactional file copy is to be performed: • If value is **yes**, the file is copied to a file with a temporary name that is renamed to the destination filename once the file has been successfully transferred. • If value is **no**, the file is copied directly to the specified destination filename. [Default is **no**.] Note: For z/OS and OS/400, **`trans`** is valid only under the **hfs** file system.

Attribute Name	Values	Description
truncext	**yes** or **no**	Specification for whether or not the source filename's extension should be truncated if it is being used as the destination filename (no filename was explicitly specified on the destination side of the transfer operation). • If the value of this attribute is **yes**, the extension is truncated. • If the value is **no**, the filename is left untouched. [Default is **no**.] Note: UDM considers a file extension to be the sequence of characters following the last dot (**.**) character in the filename. When an extension is truncated, the dot marks the beginning of the extension and is truncated as well. UDM will not consider the a dot character as the first character in a filename as indicating a file extension.
umask	Three-digit octal value	File permissions mask used to create the destination file. Refer to the UNIX man page umask(1) for complete details. [By default, **umask** is not defined.] Note: For UNIX and z/OS (under the **hfs** file system), **umask** is valid only on the destination side of the transfer.
usefqn	**yes** or **no**	Specification, when copying a data set under the **dsn** file system, whether or not the fully qualified data set name is sent over as the source file name to be used by the destination if an explicit destination filename is not given. • If set to **yes**, the fully qualified data set name is sent. • If set to **no**, only the part of the data set name matching the source mask in the copy operation is used as the destination filename. [Default is **no**.] Note: For z/OS, **usefqn** is valid only on the source side of the transfer.

Table 6.4 Common File System Attributes

Example

To set the line length, line operation, and line padding sequence:

```
attrib ntmachine linelen=80 lineop=wrap padline=none
```

6.4 break

Syntax

```
break
```

Description

The **break** command stops iterating through a **forfiles** loop and picks up execution at the script line immediately following the **end** statement marking the end of the **forfiles** loop.

Example

To iterate through the files on a machine and attempt to delete each file (if the delete operation fails, UDM breaks out of the loop):

```
forfiles local=*
   delete $(_path)
   if $(_lastrc) NE 0
      print msg="Failed to delete $(_path)"
      break
   end
end
```

6.5 call

Syntax

```
call script-file [parameter-name=parameter-value]...
```

Description

The **call** command loads and executes a command script.

Scripts are interpreted line by line, with parameter substitution made just before each line is executed. Parameter substitution is indicated with a $(PARAMETER_NAME) sequence in the called script. Parameters are replaced by the value corresponding with the parameter's name (name=value format) in the **call** command itself. Each parameter of the **call** command must have a corresponding value.

call commands can be nested in called scripts up to ten levels deep.

If a parameter with the same name appears more than once in **call**, the first instance of the parameter with that name is used. If a parameter is referenced in the script, but was not passed in via the **call** command, an error is issued when the line with the reference is parsed.

Parameters

Parameter	Description
script-file	Filename of the script to process.
parameter-name	Name of a parameter to pass to the script.
parameter-value	Value to be set for the parameter.

Table 6.5 **call** Command Parameters

Examples

To invoke a script called **script.udm**.

- Parameter **file** has a value of *.
- Parameter **src** has a value of **c:\source**.
- Parameter **dst** has the value **/etc/dest**.

```
call script.udm file=* src=c:\source dst=/etc/dest
```

To invoke a script under OS/400, the member name is required and can be ***FILE**:

```
call mylib/myfile(myscript)
```

Specifying ***FILE** invokes the normal default OS/400 file search order.

To invoke a script under OS/400 included as an inline file in a database job, the call must specify ***FIRST** as the database member name.

The following example illustrates both:

- Invocation of an inline script, **CALLME**, using the STRUDM command from a database job.
- Invocation of an inline script, **CALL1**, using the CALL command from a database job.

```
//BCHJOB JOB(testcall) ENDSEV(10) OUTQ(mytest/UDMOUTQ) LOGCLPGM(*YES)
LOG(2 20 *SECLVL) MSGQ(*USRPRF)
//DATA FILE(CALL1) ENDCHAR(ENDDATAFILE)
print msg="I made it to call1 - an inline file"
ENDDATAFILE
//DATA FILE(CALLME) ENDCHAR(ENDDATAFILE)
OPEN S=AS400V5 USER=qatest PWD=***** PORT=4311
CALL CALL1(*FIRST)
CLOSE
ENDDATAFILE
STRUDM SCRFILE(CALLME)
//ENDBCHJOB
```

6.6 cd

Syntax

cd *logical-name*[*=directory*]

Description

The **cd** command changes the working directory (on UNIX, Windows, OS/400, and file system HFS; on z/OS, the current data set qualifiers for DSN and DD file systems) on the specified logical machine (*logical-name*) to the specified path.

If no directory is specified, the current working directory (or qualifier) is printed to the UDM Manager.

Parameters

Parameter	Description
logical-name	Logical name of the transfer server to execute the **cd** command.
directory	Directory to be changed to the working directory on the server.

Table 6.6 **cd** Command Parameters

Example

To change the current directory on the machine with the logical name **ntmachine** to **c:\src**:

```
cd ntmachine=c:\src
```

6.7 close

Syntax

```
close
```

Description

The **close** command closes the current transfer session.

When a close message is issued, the UDM Manager lets the primary server know to close down the session. The primary server in turn notifies the secondary server.

Once a session has been closed, a new transfer session can be established with the **open** command. If a session is not open, an error message is printed.

6.8 closelog

Syntax

```
closelog
```

Description

The **close** command closes the open log file.

If the **close** command is issued when a log file is not open, an error is produced.

6.9 compare

Syntax

compare *STRING_1 STRING_2* [case=**yes**|**no**] [length=*length*]

Description

The **compare** command compares two strings.

The first two parameters (*STRING_1* and *STRING_2*) are the strings to be compared.

The optional **case** parameter specifies whether the comparison is case-sensitive (**yes**) or case-insensitive (**no**). [Default is **no**.]

If the **length** parameter is set, only the first **n** characters are compared.

The **_lastrc.result** variable receives the result of the comparison:

- If the strings match, 0 is stored in the result.
- If the strings do not match, the index of the point at which the comparison failed is stored as the result.

The **_lastrc.message** built-in variable contains MATCH if the strings are equal; it contains NO_MATCH if the strings are not equal.

Parameters

Parameter	Description
STRING_1	String to be compared to *STRING_2*.
STRING_2	String to be compared to *STRING_1*.
case=**yes** | **no**	Specification for whether or not the comparison is case-sensitive: • If set to **yes**, comparison is case-sensitive. • If set to **no**, comparison is not case-sensitive. Note: If **case** is not used, the comparison is case insensitive.
length=*length*	First **n** characters to be compared.

Table 6.7 **compare** Command Parameters

6.10 copy

Syntax

copy *source-logical-name=source-file-specification*
[*destination-logical-name=destination-file-specification*]

Description

The **copy** command initiates a copy operation.

source-logical-name specifies the logical name of the source server (logical name of either the primary or secondary transfer server specified in the open command).

source-file-specification specifies the complete path or single file name of the file or files to be copied.

Optionally, *destination-logical-name* and *destination-file-specification* can be used to specify the logical name of the destination server and the complete path or single file name for the destination file.

If *destination-file-specification* identifies only a file name, the current directory (or high-level qualifier) is used for the destination server. If *destination-file-specification* identifies only a directory, the file name specified in *source-file-specification* is used.

If *destination-logical-name* and *destination-file-specification* are not specified, the other server in the transfer session (that is, the server not specified in *source-logical- name)* is assumed to be the destination server and *source-file-specification* is used for the destination file name.

By default, the destination file has timestamps matching its creation date, last modification date, and access date. However, via attributes set in the attrib command, you can set the destination file to have timestamps matching the source file.

Note: If an error is encountered, a copy operation will halt, and control will be returned to the script.

Parameters

Parameter	Description
source-logical-name	Logical name of the server acting as the source of the transfer operation.
source-file-specification	Complete path or single file name of the file or files to be copied. The file name (or file name portion of the path) can contain wildcard characters: • Wildcard ***** represents zero or more characters. • Wildcard **?** indicates a single character.
destination-logical-name	Logical name of the destination server in the transfer operation.
destination-file-specification	Complete path or single file name for the destination file.

Table 6.8 **copy** Command Parameters

Examples

To copy file **`test.txt`** - from a machine with logical name **`src`** to a machine with logical name **`dst`** - as **`test.bak`**:

```
copy src=test.txt dst=test.bak
```

To copy all files in the current directory from a machine with logical name **`src`** to the other machine in the transfer session:

```
copy src=*
```

6.11 copydir

Syntax

copydir *source-logical-name=file-specification*
[*destination-local-name=file-specification*]

Description

The **copydir** command initiates a copy operation that recurses into subdirectories.

If the source of the copy operation has subdirectories beneath the location given by the file-spec, UDM will create those directories on the destination side of the transfer and copy their contents as well. If any of the directories already exist on the destination side (relative to the destination file specification), the copy operation will fail if the **creatop** attribute on the destination side is not set to replace.

The source file specification is given as the value for source host's logical name (which should be either the primary or secondary logical name specified in the **open** command. An optional destination file-spec may be given as well. If none is given, the current directory (or high-level qualifier) for the destination machine in the transfer session is used.

If no destination server is given in the command (the command contains only the source's logical name and file specification), the other server in the transfer operation is assumed to be the destination and the source filename is used for the destination filename.

If a destination file specification is not given, or contains only a directory sequence, but no filename, the filename of the source file will be used.

The **copydir** command is available only on UNIX, Windows, and the **hfs** file systems for z/OS and OS/400.

Parameters

Parameter	Description
source-logical-name	Server acting as the source of the transfer operation.
file-specification	File or files to be copied. It also can specify a directory whose contents should be copied. The file specification can be a single filename, directory name, or a complete path to a file or directory. The filename (or filename portion of the path) can contain wildcard characters: • Wildcard ***** represents stands for zero or more characters. • Wildcard **?** indicates a single character.
destination-logical-name	Logical name of the destination server in the transfer.
file-specification	Complete path or filename for the destination file.

Table 6.9 **copydir** Command Parameters

Examples

To copy all files in a directory, recursing through all subdirectories:

```
copydir local=/mydir/*
```

To copy all files in a directory (same as above) by specifying the directory name only in the source (no wildcards or filename portion is needed):

```
copydir local=/mydir
```

To copy the files in **mydir** and all of its subdirectories into another existing directory on the destination side:

```
copydir local=/mydir dest=/yourdir
```

To copy an entire directory structure underneath a subdirectories and any files ending in **.txt**:

```
copydir local=/mydir/*.txt
```

6.12 data

Syntax

```
data [name|print=name] [resolve={all|defined|no}] data-element
     [end=end-sequence]
```

Description

The **data** command defines an in-stream data element that can be passed as input for other commands.

Parameters

Parameter	Description
name | **print**=*name*	Specifies either the **name** of the in-stream data element being defined (*name*) or a request to **print** the lines of that data element.
resolve={**all** | **defined** | **no**}	Variable resolution method: • **all** Resolve all variable references in the data. • **defined** Resolve only defined variable references in the data. • **no** Do not resolve any variable references in the data.
data-element	Contents of the in-stream data element being defined.
end=*end-sequence*	Sequence indicating the end of the data.

Table 6.10 **data** Command Parameters

6.13 debug

Syntax

debug [EXPRESSION_SHOW_POSTFIX=**yes**|**no**]
[EXPRESSION_SHOW_EVALUATION=**yes**|**no**]
[COMMAND_SHOW_STRUCTURE=**yes**|**no**]

Description

The **debug** command turns debug information on and off.

Each parameter identifies a debug feature, as described in Table 6.11, and specifies whether that feature is turned on (**yes**) or off (**no**).

Parameters

Parameter	Description
EXPRESSION_SHOW_POSTFIX=**yes** | **no**	Postfix version of an expression after it has been converted from infix notation.
EXPRESSION_SHOW_EVALUATION=**yes** | **no**	Evaluation of an expression.
COMMAND_SHOW_STRUCTURE=**yes** | **no**	Different elements of a command after it has been parsed.

Table 6.11 **debug** Command Parameters

6.14 delete

Syntax

delete *logical-name=file-specification*

Description

The **delete** command deletes a file (or series of files if *file-specification* contains any wildcards) from the transfer server with the corresponding logical name.

Parameters

Parameter	Description
logical-name	Server from which you want to delete the file(s).
file-specification	File or files to be deleted; it can be a single filename or a complete path to a file or directory. The filename (or filename portion of the path) can contain wildcard characters: • Wildcard ***** represents zero or more characters. • Wildcard **?** indicates a single character. Note: In z/OS, wildcards can be used on sequential data sets.

Table 6.12 **delete** Command Parameters

Examples

To delete all members of a PDS:

```
delete local='my.pds.name(*)'
```

To delete all files in a single directory level:

```
delete local=/mydir/*
```

To delete a single file:

```
delete local=myfile.txt
```

6.15 deletestring

Syntax

```
deletestring  variable_name pos=position |{startseq=sequence
              [startseqnum=number]} length=length |{endseq=sequence
              [endseqnum=number]} [case=yes|no]
```

Description

The **deletestring** command removes a substring from an existing string.

The first parameter, **VARIABLE_NAME**, is the name of an existing variable.

The beginning of the sequence to be deleted is indicated by either its starting position (one-based index, using the **pos** parameter) or as immediately following a particular occurrence of a character sequence (**startseq** specifies the sequence and the optional **startseqnum** specifies the occurrence number).

The end of the substring is determined by either specifying the length of the substring using the **length** parameter or giving a sequence that indicates the end of the substring (**endseq** specifies the ending sequence and the optional **endseqnum** specifies the occurrence number).

The optional **case** parameter specifies whether the comparisons of the sequences are case-sensitive (**yes**) or case-insensitive (**no**). [Default is **no**.]

The **_lastrc.message** built-in variable will contain:

- NO_MATCH if start or end sequences were specified and could not be matched
- INVALID_VALUE if the starting position or length are out of range
- SUCCESS if a sub-string was successfully deleted

Parameters

Parameter	Description
variable_name	Name of an existing variable.
pos=*position*	Starting position of sequence to be deleted (one based index).
startseq=*sequence*	Starting position of sequence to be deleted (following a specific character sequence).
startseqnum=*number*	Occurrence number of starting position of sequence to be deleted (following a specific character sequence).
length=*length*	Length of the substring to be deleted.
endseq=*sequence*	Ending position of sequence to be deleted (preceding a specific character sequence).
endseqnum=*number*	Occurrence number of ending position of sequence to be deleted (preceding a specific character sequence).
case=**yes** | **no**	Indicates whether or not the comparisons of the sequence are case-insensitive. [Default is **no**.]

Table 6.13 **deletestring** Command Parameters

Examples

The following examples illustrate how **`deletestring`** is used starting with a sample string called **`mystring`** with an initial value of **`This is not just some sample text`**:

```
set mystring="This is not just some sample text"
deletestring mystring pos=8 length=4
echo "$(mystring)"
This is just some sample text

set mystring="This is not just some sample text"
deletestring mystring startseq=" " startseqnum=2 endseq=" "
echo "$(mystring)"
This is just some sample text

set mystring="This is not just some sample text"
deletestring mystring startseq=" " startseqnum=2 length=21
echo "$(mystring)"
This is text
```

6.16 echo

Syntax

echo [*parm_1*]...[*parm_n*]

Description

The **echo** command writes text to standard out.

Parameters

Parameter	Description
parm_1	First parameter to echo.
parm_n	Other parameter(s) to echo.

Table 6.14 **echo** Command Parameters

Examples

To write:

This is my message:

Execute:

```
echo "This is my message"
```

6.17 echolog

Syntax

echolog [*value_1*]... [*value_n*]

Description

The **echolog** command writes text to the open log file.

If a log file is not open, **echolog** issues an error.

Parameters

Parameter	Description
value_1	First parameter to echo.
value_n	Other parameter(s) to echo.

Table 6.15 **echolog** Command Parameters

Examples

To write:

```
Tobeornottobe!Thatisthequestion.
```

Execute:

```
echolog To be or not to be! That is the question.
```

To write:

```
To be or not to be! That is the question.
```

Execute:

```
echolog "To be or not to be! That is the question."
```

To write:

```
1 + 1 = 2.
```

Execute:

```
echo "1 + 1 = " <1 + 1>
```

6.18 exec

Syntax

```
exec {logical-name | host-name} {cmd=command | cmdref=command-ref |
     stc=started-task} [user=userid] [pwd=password] [port=port]
     [codepage=codepage] [file=filename | xfile=filename [key=key]]
     [option=option] [mergelog=yes|no] [trace=yes|no]
     [input=data-element] [svropt=server-options]
     [stdout=data-element] [stderr=data-element]
```

Description

The **exec** command executes system commands on remote machines if you have a licensed version of the Universal Command (UCMD) Manager, version 3.1.1 or later, on the same system with the UDM Manager.

The first parameter of the **exec** command is either:

- Logical name (**logical-name**) of a transfer server (valid only if a transfer session has been established)
- Host name (**host-name**) of the machine on which you want to execute the command.

Note: You must have the UCMD Server and Universal Broker installed on the machine on which the command is to be executed.

The second parameter is the command type, which is either:

- **cmd** (command)
- **cmdref** (command reference)
- **stc** (started task)

For any of these three types, the value (*command*) is the remote command to be executed. (See the Universal Command 3.2.0 User Guide for more information about command types.)

UDM must authenticate a user on the remote machine in order to execute a command.

- If a logical name is specified in the first parameter, the **user** and **pwd** values are inherited from the same options specified in the open command for that logical name. These inherited values can be overridden by specifying them explicitly in the **exec** command.
- If a host name is specified in the first parameter, the **user** and **pwd** values must be specified explicitly in the **exec** command.

The **port** and **codepage** values are inherited from the UDM Manager's configuration file unless overridden explicitly in the call to the **exec** command.

- **port** specifies which port the Universal Broker is listening on for the remote machine.
- **codepage** specifies to which codepage the output of the remote command is translated.

The **user**, **pwd**, **port**, and **codepage** parameters can be stored in an external file instead of being specified explicitly in the **exec** command.

- If a plain text file is used, use the **file** parameter to specify the name of this file.
- If the file was encrypted with Universal Encrypt, use the **xfile** parameter to specify the name of this file.

If an encryption key other than the Universal Encrypt default was used, specify that key with the **key** parameter.

These parameters, and the format of the file containing these parameters, work exactly like the corresponding option in the **open** command.

The **option** parameter is used to pass options to the UCMD Server (see the SCRIPT_OPTIONS option for UCMD Manager in the Universal Command 3.2.0 User Guide for more details).

Two streams of data come back from the remote command. By default, output from standard out and standard error of the remote command are written to standard out and standard error by the UDM Manager (SYSPRINT and SYSOUT, respectively, under z/OS). The **mergelog** option can be set to yes if you want both output streams written to the UDM transaction log (standard out under UNIX, Windows, and OS/400; SYSPRINT under z/OS).

By default, if the UDM Manager is invoked with tracing turned on, tracing will be turned on in the Universal Command Manager when UDM invokes it via the **exec** command. Likewise, if **trace** is turned off in the UDM Manager, the Universal Command Manager is invoked with tracing turned off. You can override this behavior for the UCMD Manager invocation by setting the **trace** option in the call to the **exec** command.

There are some commands that require input from standard input. To provide this input, you must create a data element with the data command containing the input. Specifying the name of the data element with the **input** parameter will cause the information in the data element to be sent over as standard input to the remote command.

The **svropt** parameter can be used to override UCMD Server options.

Note: UDM does not require a space before the server options, as does Universal Command.

The **stdout** and **stderr** parameters specify data elements to contain standard out and standard error, respectively, from the remote command. If the data elements do not exist, they are created. If the data elements do exist, they are overwritten with the output from the remote command. If the value portion refers to an existing non-data element variable or the name of a built-in variable (that is, any variable beginning with an underscore), an error is issued.

The **exec** command output will still be written to UDM stdout (the transaction log) and UDM stderr, where appropriate, even with the presence of the **stdout** and/or **stderr**.

Parameters

Parameter	Description
logical-name	Logical name of a transfer server as specified in the **open** statement (valid only if a transfer session has been established) of the machine on which you want to execute the command. Note: You must have the Universal Command Server and Universal Broker installed on the machine on which the command is to be executed.
host-name	Host name of the machine you wish to execute the command on. Note: You must have the Universal Command Server and Universal Broker installed on the machine on which the command is to be executed.
cmd=*command*	Command type **cmd** (command) will be executed on the remote server with its value being the command to be executed. (See the Universal Command User Guide for more information about available command types for each platform.)
cmdref=*command-ref*	Command type **cmdref** (command reference) will be executed on the remote server with its value being the command reference to be executed. (See the Universal Command User Guide for more information about available command types for each platform.)
stc=*started-task*	Command type **stc** (started task) will be executed on the remote server with its value being the started task to be executed. (See the Universal Command User Guide for more information about available command types for each platform.)
user=*user-id*	User ID (**user**) and password (**pwd**) are inherited from the same parameters specified in the **open** command for that logical name. These values can be overridden by specifying them explicitly in the **exec** command. Note: If a host name is used (instead of a logical name), a user ID and password must be specified explicitly in the **exec** command.
pwd=*password*	User ID (**user**) and password (**pwd**) options are inherited from the same options specified in the **open** command for that logical name. These values can be overridden by specifying them explicitly in the **exec** command. Note: If a host name is given (instead of a logical name), a user ID and password must be specified explicitly in the **exec** command.
port=*port*	Port that the Universal Broker is listening on for the remote machine. The **port** is inherited from the UDM Manager's configuration file unless explicitly overridden in the call to the **exec** command.
codepage=*codepage*	Codepage to which the remote command output is translated. The **codepage** is inherited from the UDM Manager's configuration file unless overridden explicitly in the call to the **exec** command.
file=*filename*	Plain text file containing the values for the remote execution server: **port**, **user**, **pwd**, and **codepage**. Note: These values override any corresponding values specified for the transfer server in the **open** command.
xfile=*filename*	Universal Encrypted text file containing the values for the remote execution server: **port**, **user**, **pwd**, and **codepage**. Note: These values override any corresponding values specified for the transfer server in the **open** command.

Parameter	Description
key=*key*	Key used to decrypt the file specified by the **`xfile`** parameter. If this parameter is not present, the default key for Universal Encrypt is used.
option=*option*	Passes options to the Universal Command Server.
mergelog=**yes** | **no**	Specification for whether or not to merge the two streams of data that come back from the remote command to the UDM transaction log (SYSOUT under UNIX, Windows, and OS/400; SYSPRINT under z/OS). By default, output from standard out and standard error of the remote command are written to standard out and standard error by the UDM Manager (SYSPRINT and SYSOUT, respectively, under z/OS). [Default is **no**.]
trace=**yes** | **no**	Overrides the trace behavior of the Universal Command Manager when invoked via the **`exec`** command. [Default is the trace value of the UDM Manager.]
input=*data-element*	Data element to be used as standard input to the remote command.
svropt=*server-options*	Overrides Universal Command Server options.
stdout=*data-element*	Data element to be used as standard out from the remote command.
stderr=*data-element*	Data element to be used as standard error from the remote command.

Table 6.16 **exec** Command Parameters

6.19 execsap

Syntax

```
execsap host={host-name|destination} type={event|generic}
        [eventid=event-id] [parm=event-parm] [client=client]
        [user=userid] [pwd=password] [codepage=codepage]
        [file=filename | xfile=filename [key=key]] [mergelog=yes|no]
        [trace=yes|no]
```

Description

The **execsap** command executes SAP events if you have a licensed version of the Universal Connector (version 3.1.1 or later) on the same system with the UDM Manager.

Note: UDM does not support the **execsap** command for OS/400 and Windows.

The first parameter of the **execsap** command is either:

- Host parameter with an SAP destination entry
- Name of a destination in your SAP RFC file

The **type** parameter specifies the type of action being performed. A specified type of **event** requires that an SAP event ID be specified with the **eventid** parameter.

Note: For version 3.2.0, the only valid type is **event**, which triggers an SAP event.

An event parameter can be passed to the SAP event using the **parm** parameter. The **client** parameter specifies the SAP client.

UDM must authenticate a user SAP in order to execute an SAP action. The user id and password can be specified with the **user** and **pwd** parameters, respectively.

The **codepage** value is inherited from the UDM Manager's configuration file unless overridden explicitly in the call to the **execsap** command. **codepage** specifies to which codepage the output of the remote command is translated.

The **user**, **pwd**, and **codepage** parameters can be stored in an external file instead of being specified explicitly in the **execsap** command syntax.

- If a plain text file is used, the **file** parameter specifies the name of this file.
- If the file was encrypted with Universal Encrypt, the **xfile** parameter specifies the name of this file.

If an encryption key other than Universal Encrypt's default was used, that key can be specified with the **key** parameter.

These options and the format of the file containing these options work exactly like the corresponding option in the **open** command.

Two streams of data come back from the SAP execution. By default, output from standard out and standard error is written to standard out and standard error by the UDM Manager (SYSPRINT and SYSOUT, respectively, under z/OS). The **mergelog** parameter can be set to **yes** if you want both output streams written to the UDM transaction log (standard out under UNIX, Windows, and OS/400; SYSPRINT under z/OS).

By default, if the UDM Manager is invoked with tracing turned on, tracing will be turned on in the Universal SAP Connector when UDM invokes it via the **execsap** command. Likewise, if **trace** is turned off in the UDM Manager, the Universal SAP Connector is invoked with tracing turned off. You can override this behavior for the Universal SAP Connector invocation by setting the **trace** option in the call to the **execsap** command.

Parameters

Parameter	Description
host=*hostname* | *destination*	SAP destination entry or, if host=name format is used, the name of a destination in you SAP RFC file.
type=**event** | **generic**	Type of action (**event** is the only type supported under UDM 3.2.0).
eventid=*event-id*	ID of the SAP event to trigger.
parm=*event-parm*	Passes an event parameter to the SAP event.
client=*client*	SAP client for which to perform the action.
user=*user-id*	User ID (**user**) and password (**pwd**) are inherited from the same parameters specified in the **open** command for that logical name. These values can be overridden by specifying them explicitly in the **execsap** command. Note: If a host name is used (instead of a logical name), a user ID and password must be specified explicitly in the **execsap** command.
pwd=*password*	User ID (**user**) and password (**pwd**) are inherited from the same parameters specified in the **open** command for that logical name. These values can be overridden by specifying them explicitly in the **execsap** command. Note: If a host name is used (instead of a logical name), a user ID and password must be specified explicitly in the **execsap** command.
codepage=*codepage*	Codepage to which the remote command output is translated. The **codepage** is inherited from the UDM Manager's configuration file unless overridden explicitly in the call to the **execsap** command.

Parameter	Description
file=*filename*	Plain text file containing the values for the remote execution server: **user**, **pwd**, and **codepage** These values override any corresponding values specified for the transfer server in the **open** command.
xfile=*filename*	Universal Encrypted text file containing the values for the remote execution server: **port**, **user**, **pwd**, and **codepage**. These values override any corresponding values specified for the transfer server in the **open** command.
key=*key*	Key used to decrypt the file specified by the **xfile** parameter. If this parameter is not present, the default key for Universal Encrypt is used.
mergelog=**yes** | **no**	Specification for whether or not to merge the two streams of data that come back from the remote command to the UDM transaction log (SYSOUT under UNIX, Windows, and OS/400; SYSPRINT under z/OS). By default, output from standard out and standard error of the remote command are written to standard out and standard error by the UDM Manager (SYSPRINT and SYSOUT, respectively, under z/OS). [Default is **no**.]
trace=**yes** | **no**	Overrides the trace behavior of the Universal Command Manager when invoked via the **execsap** command. [Default is trace value of the UDM Manager.]

Table 6.17 **execsap** Command Parameters

6.20 exit

Syntax

```
exit
```

Description

The **exit** command behaves the same as the **quit** command (see Section 6.37 quit).

6.21 filesys

Syntax

filesys *logical-name*[={**dd**|**dsn**|**hfs**|**lib**}]

Description

The **filesys** command sets the file system with which the server specified by **logical-name** is working.

UNIX and Windows

For UNIX and Windows systems, there is only one file system; specifying a **filesys** value will result in a warning.

z/OS

For z/OS systems, this value can be either:

- dd (ddnames defined with JCL DD statements)
- dsn (data set name)
- hfs (UNIX System Services file system).

Note: dd is available only on z/OS Manager for two-party transfer.

OS/400

For OS/400 systems, this value can be either:

- lib (Library file system)
- hfs (IFS file system: root or QOpenSys)

Parameters

Parameter	Description
logical-name	Logical name of the transfer server on which to change the file system.
dd | **dsn** | **hfs** | **lib**	File system with which the specified server is working: • If the value is **dd**, the DD name file system is used on the specified server (valid only for z/OS). • If the value is **dsn**, the data set name file system is used (valid only for z/OS). • If the value is **hfs**, the HFS file system is used. (Valid only for z/OS and OS/400. For z/OS, USS is used; for OS/400, IFS (root or QOpenSys) is used.) • If the value is **lib**, the **LIB** file system is used (valid only for OS/400). If no value is given, UDM will return the current file system for the specified transfer server. [Default is **dsn**.] Note: UDM supports multiple file systems only under z/OS and OS/400. An attempt to change the file system for a server running under Windows or UNIX will result in an error.

Table 6.18 **filesys** Command Parameters

Example

To set the file system on the server with the logical name **mvsmachine** to the **dsn** file system:

```
filesys mvsmachine=dsn
```

6.22 filetype

Syntax

```
filetype    [binary|text=file-mask-1]...[binary|text=file-mask-n]
            [remove=file-mask-1]...[remove=file-mask-n] [resetall]
```

Description

The **filetype** command sets a series of masks and corresponding transfer mode types.

For any file whose source name matches a specific mask, the transfer mode type corresponding to that mask (text or binary) is used in transferring that file. If the source file name in a transfer does not match any registered masks, the default transfer mode type (set using the **mode** command) is used.

Issuing the **filetype** command by itself, with no parameters, dumps all of the file masks and their corresponding mode types that have been set.

Parameters

Parameter	Description
binary | **text** = *filemask_1*	Sets the specified file mask (*filemask_1*) so that source file names matching this mask will be transferred as binary files.
binary | **text** = *filemask_n*	Sets the specified file mask (*filemask_n*) so that source file names matching this mask will be transferred as binary files.
remove=*filemask_1*	Removes an entry that matches the specified file mask (*filemask_1*).
remove=*filemask_n*	Removes an entry that matches the specified file mask (*filemask_n*).
resetall	Removes all file mask entries.

Table 6.19 **filetype** Command Parameters

Examples

To set a single file mask and its corresponding transfer mode type:

```
filetype binary=*.exe
```

To set a series of file masks in a single call, instead of making multiple calls:

```
filetype text=*.txt text=*.c binary=*.exe binary=*.dat
```

To remove a file mask:

```
filetype remove=*.txt
```

6.23 find

Syntax

```
find string seq=sequence [pos=index] [case=yes|no] [num=number|last]
```

Description

The **find** command finds a specific occurrence of a substring in an existing string or list element.

The first parameter, *string*, is the string in which to search for the sequence. It can be a variable reference or a literal. The **seq** parameter specifies the sequence for which the search is being made.

The optional **pos** parameter specifies the one-based index of the string where the find operation begins.

The optional **case** parameter indicates whether the search is case-sensitive (**yes**) or case-insensitive (**no**). [Default is **no**.]

The optional **num** parameter specifies the instance of the sequence for which the search is being made. If the value of **num** is **last**, the find function gives back the index of the last occurrence of the sequence in the string.

If the sequence is found, the index (one-based) of the starting point of the requested occurrence is place in the **_lastrc.result** variable. If the sequence was not found in the string, a value of -1 is placed in the **_lastrc.result** variable.

If the sequence is found, **_lastrc.message** contains a value of MATCH. If the sequence could not be found, **_lastrc.message** contains NO_MATCH.

Parameters

Parameter	Description
string	String in which to search for the sequence.
seq=*sequence*	Sequence for which the search is being made.
pos=*index*	One-based index of the string where the **find** operation begins.
case=**yes** | **no**	Specification of whether the search is case-sensitive (**yes**) or case-insensitive (**no**). [Default is **no**.]
num=*number* | **last**	Instance of the sequence for which the search is being made.

Table 6.20 **find** Command Parameters

Examples

The following examples demonstrate the **find** command:

```
find "This is a test" seq=is
echo $(_lastrc.result)
3

find "This is a test" seq=" " num=2
echo $(_lastrc.result)
8

set mystring="I hate examples"
find "$(mystring)" seq=EXAMPLES case=yes
echo $(_lastrc.result)
-1

find "This is a test" seq=" " pos=6
echo $(_lastrc.result)
8
```

6.24 format

Syntax

```
format variable_name [{string_1|expression_1}
       [align={center|left|right|justify}] [pad=sequence]
       [trunc=yes|no] [length=length]]...[{string_n|expression_n}
       [align=center|left|right|justify] [pad=sequence]
       [trunc=yes|no] [length=length]]
```

Description

The **format** command creates a formatted string.

The first parameter, *variable_name*, is the variable or list element in which the newly formatted string is stored.

The second parameter, *string_1* (or *expression_1*), is the first section of the string. Each additional, optional parameter specifies formatting that affect that string.

All following parameters, *string_n* (or *expression_n*), are the next sections of the string. Each additional, optional parameter specifies formatting that affect that string.

Parameters

Parameter	Description
variable_name	Variable or list element in which the newly formatted string is stored
string_1 | *expression_1*	First section of the string
align=**center** | **left** | **right** | **justify**	Method of how the value is aligned within the given space. [Default alignment is left.]
pad=*sequence*	Character(s) used to pad out the formatted value. [Default is space character.]
trunc=**yes** | **no**	Truncates the value to the length given by **length** if it is longer than that value. [Default is to not truncate.]
length=*length*	Length of the formatted field. [Default is length of the value specified.]

Table 6.21 **format** Command Parameters

Examples

The following examples demonstrate the **format** command.

```
set firstname="Stonebranch"
set lastname="Incorporated"
format mystring "First Name: " "$(firstname)"  " Last Name: " +
"$(lastname)" length=3 trunc=yes
echo "$(mystring)"

Output:
First Name: Stonebranch Last Name: Inc


format mystring "The " "Value " "Is: " 4 length=8 align=right pad=0
echo "$(mystring)"
The Value Is: 00000004
```

6.25 insertstring

Syntax

```
insertstring  variable_name sequence {pos=position | startseq=sequence
startseqnum=number} [case=yes|no]
```

Description

The **insertstring** command inserts a substring into an existing string.

The first parameter, *variable_name*, is the name of the existing variable or list element into which to insert the sequence.

The second parameter, *sequence*, specifies the sequence to be inserted.

The insertion point of the sequence is either:

- By position (specified by the **pos** parameter)
- At the character immediately following the instance of a given sequence (**startseq** specifies the sequence and **startseqnum** gives the instance).

The optional **case** parameter specifies whether the comparison used to find the start sequence is case-sensitive (**yes**) or case-insensitive (**no**). [Default is **no**.]

If the start sequence is given and could not be found, **_lastrc.message** will contain NO_MATCH. If a position is given and invalid, **_lastrc.message** will contain INVALID_VALUE.

Parameters

Parameter	Description
variable_name	Name of the existing variable or list element into which to insert the sequence.
sequence	Sequence to be inserted.
pos=*position*	Position at which sequence is inserted.
startseq=*sequence*	Sequence after which specified sequence is inserted.
startseqnum=*number*	Instance of sequence after which specified sequence is inserted.
case=**yes** | **no**	Specification of whether or not the comparison used to find the start sequence is case-insensitive. [Default is **no**.]

Table 6.22 **insertstring** Command Parameters

Example

The following examples show how the **`insertstring`** command can be used on a preexisting variable called **`mystring`** with a starting value of **`This is a string`**:

```
insertstring mystring " sample" pos=10
echo "$(mystring)"
This is a sample string
```

```
insertstring mystring "sample " startseq=" " startseqnum=3
echo "$(mystring)"
This is a sample string
```

6.26 loaddata

Syntax

loaddata *data-element-name=file-path*

Description

The **loaddata** command loads the contents of a data element from a file, instead of setting the contents in a script using the data command.

Parameters

Parameter	Description
data-element-name	Data element into which the file contents is loaded.
file-path	File from which contents is loaded into the data element.

Table 6.23 **loaddata** Command Parameters

Examples

If a file called **commands.txt** has the following contents:

```
cd /
ls -al
exit
```

Issuing the following commands would load the contents of **commands.txt** into a data element called **mydata** and print them out:

```
> loaddata mydata=commands.txt
> data print=mydata
cd /
ls -al
exit
```

6.27 logdata

Syntax

logdata *data_element_name*

Description

The **logdata** command writes the contents of a data element to the open log file.

If a log file is not open, or the named data element does not exist, an error is issued. This is effectively the same as constructing a **fordata** loop and writing out each line of the data element with the echolog command.

Any variable references appearing in the contents of the data element will follow the same resolution rules as if the operation were performed using the aforementioned **fordata**/echolog loop.

Parameters

Parameter	Description
data_element_name	Name of the data element to write to the open log.

Table 6.24 **logdata** Command Parameters

Example

To load some information into a data element, execute:

```
loaddata mydata=secretcodes.txt
```

To write that data element to the open log file, execute:

```
logdata mydata
```

6.28 lower

Syntax

```
lower variable_name
```

Description

The **lower** command forces all alphabetic characters in a given variable or list element to lower case.

Parameters

Parameter	Description
variable_name	Variable or list element in which to force alphabetic characters to lower case.

Table 6.25 **lower** Command Parameters

6.29 mode

Syntax

```
mode [type={text|binary}] [trim={yes|no}]
```

Description

The **mode** command sets the current transfer mode.

The **type** parameter specifies the current mode type:

- If **text** is the current mode type, file transfers are treated as text and codepage translation takes place.
- If **binary** is the current mode type, files are transferred as binary.

The **trim** parameter specifies whether or not trailing spaces are trimmed (removed) from each record / line in a text transfer.

Issuing the **mode** command without parameters displays the current transfer mode.

Parameters

Parameter	Description
type=**text** | **binary**	Sets the type of transfer. • If the transfer type is **text**, UDM performs text translation on the data between the codepages for the primary and secondary servers. • If the transfer type is **binary**, UDM sends the source data over as is, with no translation. [Default is **binary** when a new session is opened.]
trim=**yes** | **no**	Sets whether or not to trim trailing spaces during a text transfer. • If the value is **yes**, spaces at the end of each line (record) are removed from the data when it is written. • If the value is **no**, spaces at the end of each line (record) are not removed. [Default is **no** when a transfer session is initiated.]

Table 6.26 **mode** Command Parameters

Example

To set the current transfer type to text transfers:

```
mode type=text
```

6.30 move

Syntax

move *source-logical-name=source-file-specification*
[*destination-logical-name=destination-file-specification*]

Description

The **move** command initiates a move operation.

A move operation is similar to a copy operation. The only difference between a **move** command and a copy command is that after you move a file, it is deleted from the source server from which it was moved.

If the **move** command cannot delete the source file after the move, the move operation has failed; **move** deletes the destination file.

z/OS

If the source side of the move is in the DD file system, or if the source is a GDG, an error will be issued specifying that the move command is not supported.

source-logical-name specifies the logical name of the source server (logical name of either the primary or secondary transfer server specified in the open command).

source-file-specification specifies the complete path or single file name of the file or files to be copied.

Optionally, *destination-logical-name* and *destination-file-specification* can be used to specify the logical name of the destination server and the complete path or single file name for the destination file.

If *destination-file-specification* identifies only a file name, the current directory (or high-level qualifier) is used for the destination server. If *destination-file-specification* identifies only a directory, the file name specified in *source-file-specification* is used.

If *destination-logical-name* and *destination-file-specification* are not specified, the other server in the transfer session (that is, the server not specified in *source-logical- name)* is assumed to be the destination server and *source-file-specification* is used for the destination file name.

Note: As with a copy command, if an error is encountered, a move operation will halt, and control will be retuned to the script.

Parameters

Parameter	Description
source-logical-name	Logical name of the server acting as the source of the move operation.
file-specification	File or files to be copied. It can be a single filename or a complete path. The filename (or filename portion of the path) can contain wildcard characters: • Wildcard ***** represents for zero or more characters. • Wildcard **?** indicates a single character.
destination-logical-name	Logical name of the destination server in the move operation.
file-specification	Complete path or file name for the destination file.

Table 6.27 **move** Command Parameters

Examples

To move file **test.txt** - from a machine with logical name **src** to a machine with logical name **dst** - as **test.bak**:

```
move src=test.txt dst=test.bak
```

To move all files in the current directory from a machine with logical name **src** to the other machine in the transfer session:

```
move src=*
```

6.31 open

Syntax

```
open [primary={*|local|host-name}
        [port=broker-port] [user=username [pwd=password]]
        [codepage=codepage] [{file=filename | xfile=filename
        [key=key]}]
     secondary=host-name [port=broker-port] [user=username
        [pwd=password]] [codepage=codepage] [{file=filename |
        [xfile=filename [key=key]}]
     [encrypt={yes|no|cipher}]
     [compress=compression-method]
     [nft=yes|no]
     [comment=text]
```

Description

The **open** command opens a UDM transfer session.

Each transfer session has a primary transfer server and a secondary transfer server. These servers are given logical names (*primary* and *secondary*) by the user.

Each logical name is set to the host name or IP address (*host-name*) of the transfer server. For a two-party transfer session, the UDM Manager acts as the primary server, and its logical name (*primary*) must be set to * or **local** as the host name.

Optionally, only a *secondary* server may be specified, which implies a two-party transfer. In this case, the primary server automatically is assigned **local** as the logical name.

Each transfer server parameter (*primary* and *secondary*) can be followed by one or more of the following parameters that further define the transfer set-up (each parameter applies to the transfer server which it immediately follows):

- **port** specifies the port that the broker is accepting requests on to start a UDM server.
- **user** specifies the user (local to the host on which the server will be running) under which the transfer operation is being carried out
- **pwd** is the password for the user.
- **codepage** specifies the codepage that will be used for text translation of transferred data.

Note: The **user** and **pwd** parameters are not required for the local side (primary server) of a two-party transfer, as the UDM Manager will be running as the user that invoked it.

Each transfer server parameter also can be followed by the **file** or **xfile** parameter. These parameters specify a file (plain text or Universal Encrypted text, respectively) that contain **port**, **user**, **pwd**, and/or **codepage** in the format of a UDM command file (see file / xfile Format in Section Examples). If a file is specified, its values apply to the transfer server that the **file** or **xfile** parameter follows. These file values override any values specified by the **port**, **user**, **pwd**, and /or **codepage** parameters following that transfer server.

The optional **key** parameter specifies the encryption key used to decrypt the encrypted file specified by **xfile**.

The **encrypt** parameter specifies either:

- **yes**
  An agreed-upon cipher will be negotiated based on the components **data_ssl_cipher_list** configuration value.
- **no**
  NULL-MD5 is used as the encryption method.
- *cipher*
  Specific cipher to use as encryption method: RC4-SHA, RC4-MD5, AES256-SHA, AES128-SHA, DES-CBC3-SHA, DES-CBC-SHA, NULL-SHA, or NULL-MD5. Specifying NULL-NULL as the cipher completely disables SSL when NULL-NULL also is specified in the UDM Server Data Cipher Lists associated with a transfer.

Note: If **encrypt** is not used, a default value of **no** is used.

The **compress** parameter can have either of the following values:

- **yes**
  Compression option in the UDM Manager's configuration file is used.
- **no**
  No compression is used when transferring data.
- **zlib**
  Forces the transfer servers to use ZLIB compression when transferring files.
- **hasp**
  Forces the transfer servers to use HASP compression.

If **compress** is not specified, a default value of **no** is used.

The **nft** parameter specifies whether or not the UDM sessions will be network fault tolerant.

The **comment** parameter specifies a comment for a single session (or overrides a comment specified by the COMMENT option.)

For example: **open src=* dst=zos14 comment="Data transfer for account 94882"**

Parameters

Parameter	Values	Description
primary	**[{*|local|***host name***}]**	Logical name of the primary transfer server. If the value is ***** or **local**, a two-party transfer is initiated with the UDM Manager acting as the primary server. Otherwise, a three-party transfer session is initiated with the primary transfer server running on the machine specified by *host name*. If no primary transfer server is specified, a two-party transfer session is implied, with **local** as the UDM Manager / primary server's logical name.
secondary	*host name*	Logical name of the secondary transfer server. Its value is the host name or IP address of the machine on which the server will be running. Note: The host name of the secondary transfer server must be given from the perspective of the primary transfer server, not the UDM Manager.
port *	TCP port number or service name	Port on which the broker that will initiate the transfer server is listening. If this parameter is not used, the port number is assumed to be **7887**. Note: The option is not valid for the primary server in a two-party transfer.
user *	Valid username on the system the transfer server will be running on	User name to authenticate with on the transfer server. The user name must be valid on the system. Once authenticated, the default directory on the transfer server is set to the user's home directory under UNIX and HFS. Under Windows, the default directory will be a directory created for the user underneath where the Universal product suite is installed. For z/OS under the **dsn** file system, the user name will be the high level qualifier. Note: This option is not valid for the primary server in a two-party transfer.
pwd *	Password of the user to authenticate.	Password, for the specified user name, for authenticating the user on the transfer server. Note: This option is not valid for the primary server in a two party transfer.
codepage *	Valid codepage	Codepage used for text translation on the transfer server. If no codepage is specified, the codepage listed in UDM's configuration will be used.
file *	Valid filename	Plain text file containing the values for the transfer server: **port**, **user**, **pwd**, and/or **codepage** (see file / xfile Format in Section Examples). These values override any values specified by the **port**, **user**, **pwd**, and /or **codepage** parameters for the specified transfer server.
xfile *	Valid filename	Universal Encrypted text file containing the values for the transfer server: **port**, **user**, **pwd**, and/or **codepage** (see file / xfile Format in Section Examples). These values override any values specified by the **port**, **user**, **pwd**, and /or **codepage** parameters for the specified transfer server.
key *	Key used to decrypt the file specified by **xfile**	Key used to decrypt the file specified by the **xfile** parameter. If this parameter is not used, the default key for Universal Encrypt is used.

Parameter	Values	Description
encrypt	**yes**, **no**, or *cipher*	Encryption method for the transfer session. • If the value is **yes**, an agreed-upon cipher will be negotiated based on the components **`data_ssl_cipher_list`** configuration value. • If the value is **no**, the **NULL-MD5** cipher is used. Otherwise, a valid cipher must be specified: RC4-SHA, RC4-MD5, AES256-SHA, AES128-SHA, DES-CBC3-SHA, DES-CBC-SHA, NULL-SHA, or NULL-MD5. Specifying NULL-NULL as the cipher completely disables SSL when NULL-NULL is also specified in the UDM server Data Cipher Lists associated with a transfer.
compress	**yes**, **no**, **hasp**, or **zlib**	Compression method for the transfer session: • If the value is **yes**, the compression method specified in the UDM Manager's configuration is used. • If the value is **no**, no compression is used. • If the value is **hasp**, **HASP** compression is used. • If the value is **zlib**, **ZLIB** (ZIP) compression is used.
nft	**yes** or **no**	Specification for whether or not the session is network fault tolerant: • **yes** specifies that the session will be network fault tolerant. • **no** specifies the session will not be network fault tolerant.
* These parameters apply to the transfer server (primary or secondary) that they follow in the **open** command.		

Table 6.28 **open** Command Parameters

Examples

To open a two-party transfer session between two machines, explicitly specifying the primary server:

```
open primary=* secondary=mvsmachine
```

To open a two-party transfer session between two machines, not specifying the primary server:

```
open secondary=mvsmachine
```

To open a three-party transfer session between two machines:

```
open primary=ntmachine secondary=mvsmachine
```

To open a three-party transfer session between two machines with the primary server's options coming from an encrypted configuration file and the secondary server having an authenticate user and changing the codepage for its side of the transfer:

```
open primary=ntmachine xfile=enc.dat dst=test.bak user=me pwd=mypwd
```

```
codepage=IBM277
```

file / xfile Format

The parameters in a **`file`** or **`xfile`** are in the same format as the parameters in a UDM command file, as shown in Table 6.29 file / xfile Format, below.

Note: **`file`** and **`xfile`** can be shared with Universal Command.

open Command Parameter Format	file / xfile Parameter Format	Description
port=*broker-port*	-port *broker-port*	Port that the broker is accepting requests on to start a UDM server
user=*user-name*	user *user-name*	User (local to the host on which the server will be running) under which the transfer operation is being carried out
pwd=*password*	-pwd *password*	Password for the user
codepage=codepage	-codepage *codepage*	Codepage that will be used for text translation of transferred data

Table 6.29 **`file`** / **`xfile`** Format

6.32 openlog

Syntax

```
openlog  log_file_path[append=yes|no]
```

Description

The **openlog** command opens a log file on disk for writing custom log information.

Windows

The log file is open shared so that other processes can read it from while it is still open by UDM.

z/OS

The log file must be allocated as part of the job. The log file name takes the format of **dd:ddname**.

Only one log file can be open at any one time in UDM. If the user issues an **openlog** command while a log file is open, the user receives an error. The user can explicitly close the open log file by issuing a closelog command; otherwise the log file will be closed automatically when the manager ends.

UNIX, Windows

An optional **append** parameter indicates whether or not an existing log file should be appended to.

- If **append** is set to **YES** and the log file exists, any log statements written are appended to the end of that file.
- If **append** is set to **NO** and the log file exists, the file is truncated to zero length as part of the open operation.
- If **append** is set to either **YES** or **NO**, and the log file does not exist, it will be created.
- If **append** is not included in an issued openlog command, the log file is opened as if **append** were set to **NO**.

Parameters

Parameter	Description
log_file_path	Pathname of the log file to open.
append=**yes** | **no**	Specification for whether or not to append log statements to an existing log file.

Table 6.30 **openlog** Command Parameters

Examples

To open a log file under z/OS, execute:

```
openlog dd:mylog
```

To open a log file with an absolute path, execute:

```
openlog "c:\document and settings\user\logs\mylog.txt"
```

To open a log file with a relative path, execute:

```
openlog logs/mylog.txt
```

To open a log file so that new entries are appended to the dn of the existing data in the file, execute:

```
openlog mylog.txt append=yes
```

6.33 pad

Syntax

pad *variable_name* length=*length* [seq=*sequence*]

Description

The **pad** command takes a string in an existing variable or list element and pads it to make it the given length. If the string already is longer than the specified length, it is not padded.

The first parameter, *variable_name*, is the name of an existing variable or list element to pad. The `length` parameter specifies the desired length of the padded string.

The optional `seq` parameter specifies the sequence that is used for padding the string; if `seq` is not specified, a space is used.

Parameters

Parameter	Description
variable_name	Name of an existing variable or list element to pad
length=*length*	Desired length of the padded string.
seq=*sequence*	Sequence used for padding the string.

Table 6.31 **pad** Command Parameters

Examples

The following examples use a predefined string, **mystring**, with a starting value of **This is a test**.

```
pad mystring length=20
echo "$(mystring) TheEnd"
This is a test      TheEnd

pad mystring length=20 seq="-"
echo "$(mystring)"
This is a test------
```

6.34 parse

Syntax

```
parse string_name var_1 [seq_1 var_2]... [seq_n var_n]
```

Description

The **parse** command parses a string, placing components of the string into variables.

The first parameter, *string_name*, specifies the string to parse.

The *var_1* parameter specifies the first component of the parsed string. (A string can be parsed into a single component.)

The *seq_1* parameter specifies the sequence of the string that serves as a delimiter between *var_1* and the *var_2* parameter, which specifies the second component of the parsed string.

Each subsequent pair of *seq_n* and *var_n* parameters specify the delimiter (*seq_n*) between the preceding component and the next component (*var_n*).

The last *var_n* parameter in the command identifies the last component of the string.

Parameters

Parameter	Description
string_name	String to parse.
var_1	First component of the parsed string placed into a variable.
seq_1	Sequence of the string serving as a delimiter between *var_1* and *var_2*.
var_2	Second component of the parsed string placed into a variable.
seq_n	Sequence of the string serving as a delimiter between the preceding *var_n* and the following *var_n*.
var_n	Next component of the string after the preceding *seq_n*.

Table 6.32 **parse** Command Parameters

Examples

The following examples illustrate how **parse** can be used.

```
parse "This is my string" var1
echo "$(var1)"
This is my string

#comment: seq_1 equals " "
parse "This is a test" var1 " " var2 " " var3 " " var4
echo "$(var1)" "$(var2)" "$(var3)" "$(var4)"
Thisisatest


#comment: seq_1 equals "."
parse "KLH1A.DEV.UDM.TEST" var1 . var2 . var3 . var4
echo "$(var4)"
TEST

set string="This is a test"
parse $(string) var1 " is a" var2
echo "$(var1)" "$(var2)"
This test
```

6.35 print

Syntax

```
print msg=message
```

Description

The **print** command prints a message in the UDM Manager's transaction output.

If the message contains spaces, it should be surrounded by double (**"**) quotation marks.

Parameters

Parameter	Description
msg=*message*	Message to be printed by the UDM Manager in its transaction output.

Table 6.33 **print** Command Parameter

Examples

To print a simple message:

```
print msg="This is a simple message."
```

To print a message containing the value of a variable:

```
print msg="The current value of the UDM return code is: $(_rc)"
```

6.36 query

Syntax

```
query
```

Description

The **query** command prints the UDM Manager version.

If a session is established, the primary and secondary transfer servers display their host name and UDM version, as in this sample:

```
machine1: Connected to UDM host endymion version 1.1.0 Level 0
machine2: Connected to UDM host hyperion version 1.1.0 Level 0
```

Each line is prefixed with the logical name of a transfer server and contains its host name and version information for the UDM server instance.

6.37 quit

Syntax

```
quit
```

Description

The **quit** command exits the UDM Manager. If a session is open, the UDM Manager issues a **close** command before exiting to end the transfer session. UDM returns the value of the internal variable **_rc** when it exits (see set command).

Note: The UDM Manager will issue a **quit** command automatically if the return value of any command is greater than the value of the internal variable, **_halton** (if the value of **_halton** is greater than zero).

6.38 rename

Syntax

rename *logical-name old-filename new-filename*

Description

The **rename** command renames a file.

Parameters

Parameter	Description
logical-name	Logical name of the transfer server on which you want to rename a file.
old-filename	Current name (filename, path, or absolute path) for the file that you want to rename.
new-filename	New name (filename, path, or absolute path) to which the file is to be renamed.

Table 6.34 **rename** Command Parameter

6.39 replace

Syntax

```
replace variable_name oldsequence newsequence
        [num=index] [all=yes|no] [case=yes|no]
```

Description

The **replace** command replaces one or more instances of a sequence with another sequence.

The first parameter, *variable_name*, is the name of an existing variable or list element on which to perform the replace. The *oldsequence* parameter specifies the sequence to be replaced. The *newsequence* parameter specifies the sequence replacing *oldsequence*.

The optional num parameter specifies a one-based sequence number that identifies the occurrence of the sequence that is replaced.

The optional all parameter indicates whether all instances of the old sequence are replaced (**yes**) or just the first instance encountered (**no**). If all is not specified, it is assumed that only the first instance is replaced.

If all equals **yes**, and num also is specified, num is ignored.

The optional case parameter specifies whether the comparison to match the old sequence is case sensitive (**yes**) or case insensitive (**no**). [Default is **no**.]

Parameters

Parameter	Description
variable_name	Name of the existing variable or list element on which to perform the replace.
oldsequence	Sequence to be replaced.
newsequence	Sequence replacing *oldsequence*.
num=*index*	One-based sequence number that identifies the occurrence of *oldsequence*.
all=**yes** | **no**	Specification for whether all instances (**yes**) of the sequence are replaced or only the first instance (**no**) of the sequence.
case=**yes** | **no**	Specification for whether the comparison to match the old sequence is case sensitive (**yes**) or case insensitive (**no**). [Default is **no**.]

Table 6.35 **replace** Command Parameters

Examples

The following examples use a predefined variable, **mystring**, with a value of **abcabcabc**.

```
replace mystring ab " " all=yes
echo "$(mystring)"
 c c c

replace mystring "abc" "123" num=1
echo "$(mystring)"
123abcabc
```

6.40 report

Syntax

```
report progress=yes|no
```

Description

The **report** command sets reporting options for UDM.

Parameters

Parameter	Description
progress=**yes** | **no**	Turns on or off the writing of periodic transfer progress messages by the UDM Manager during a file transfer operation.

Table 6.36 **report** Command Parameters

6.41 resetattribs

Syntax

resetattribs *logical-name*

Description

The **resetattribs** command resets the attributes for all UDM file systems on the transfer server with the specified logical name.

Parameters

Parameter	Description
logical-name	Logical name of the transfer server whose attributes are to be reset to their default values.

Table 6.37 **resetattribs** Command Parameters

6.42 return

Syntax

```
return [return-value]
```

Description

The **return** command, generally used in conjunction with an **if** statement, performs the following:

1. Stops executing the current script immediately.
2. Returns execution to the calling script immediately after the **call** command used to invoke the current script.

If the current script was called by invoking UDM with the SCRIPT_FILE option, UDM will exit.

Optionally, the **return** command can specify a numeric return value. UDM will set the **_rc** variable (UDM's return code) to this value.

Parameters

Parameter	Description
return-value	Return value to which the **_rc** variable is set, specifying when to return.

Table 6.38 **return** Command Parameters

Examples

To return from the currently executing script using the current value for **_rc**:

```
return
```

To return from the currently executing script using a specific numeric return value to set **_rc** to:

```
return 4
```

6.43 reverse

Syntax

```
reverse [variable_name]
```

Description

The **reverse** command reverses the order of all characters in the string of an existing variable or element (*variable_name*).

Parameters

Parameter	Description
variable_name	String in which all characters will be reversed.

Table 6.39 **reverse** Command Parameters

6.44 savedata

Syntax

savedata [*data_element_name=file_spec*]

Description

The **savedata** command writes each line of a data element to a file on disk.

If the data element was created (via the data command) with a **resolve** value of **all** or **defined**, variable references within each line will be resolved, where appropriate. In such cases, if a variable reference cannot be resolved when called for, an error will be issued.

savedata issues an error if it cannot:

- Open the specified file.
- Write the entire contents of the data element to that file.
- Close the file.

It also issues an error if the named data element does not exist.

Parameters

Parameter	Description
data_element_name	Name of the data element.
file_spec	Name of the file on which to write each line of the data element.

Table 6.40 **savedata** Command Parameters

z/OS

When issuing **savedata**, *file_spec* must be a DD name in the format of dd:ddname, as is the case with the loaddata command. Also as with loaddata, **savedata** only deals with files on the file system local to the manager. You cannot use **savedata** to write files on other systems.

Examples

To save a data element called **mydata** to a DD name under z/OS called **export**, execute:

```
savedata mydata=dd:export
```

To save the data element to the current directory, execute:

```
savedata mydata=mydata.txt
```

To save the data element using an absolute path, execute:

```
savedata mydata="C:\documents and settings\playground\stuff.txt"
```

6.45 set

Syntax

set [built-in_name=*built-in_value_1*]…[built-in_name=*built-in_value_n*]
[global_name=*global_value_1*]…[global_name=*global_value_n*]

Description

The **set** command sets values for the UDM Manager's built-in (pre-defined) variables and global variables. (Each **set** command parameter represents a variable.)

For detailed information on these variables, see Chapter 11 UDM Scripting Language in the Universal Data Mover 3.2.0 User Guide.

Issuing the set Command

1. If the **set** command is issued without any parameters (variables), all of the global variables and their current values are displayed.
2. If the **set** command is issued with variable names but no following equal signs (**=**), the values to which the variables resolve are displayed.
3. If the **set** command is issued with variable names followed by an equal signs (**=**) but no values, the values are set to an empty string.

Parameters

Parameter	Description
built-in_name=*built-n_value_1*	Value of a built-in (pre-defined) variable
built-in_name=*built-n_value_n*	Value of a built-in (pre-defined) variable
global_name=*global_value_1*	Value of a global (user-defined) variable
global_name=*global_value_n*	Value of a global (user-defined) variable

Table 6.41 **set** Command Parameters

Example

To set UDM to echo lines and print line numbers on command errors:

```
set _echo=yes _lines=yes
```

6.46 status

Syntax

```
status
```

Description

The status command displays the current connection status.

If a session is open, the UDM Manager displays the following information for each server:

- Host name
- UDM version
- Logical name

If a session is not open, the UDM Manager displays a status message indicating that there currently is no session established.

Example

Sample output from the command under a three-party transfer session:

```
Three-party session established with ntmachine(1069560889) and
mvsmachine(1069560900)
```

The numbers in parentheses after each logical name are the component identifiers of the UDM server running on that machine.

6.47 strip

Syntax

```
strip variable_name sequence [num=index] [all=yes|no] [case=yes|no]
```

Description

The **strip** command strips occurrences of a sequence from a string.

The first parameter, *variable_name*, is the name of an existing variable or list element on which to perform the strip. The *sequence* parameter specifies the sequence to be stripped.

The optional **num** parameter indicates the occurrence number of the sequence that is to stripped.

The optional **all** parameter indicates whether all instances (yes) or only the first instance (no) of the sequence are stripped. (If **all** is not specified, it is assumed that only the first instance is stripped).

The optional **case** parameter specifies whether the comparison to find the sequence is case sensitive (**yes**) or case insensitive (**no**). **[Default is no.]**

Parameters

Parameter	Description
variable_name	Name of the existing variable or list element on which to perform the strip.
sequence	Sequence to be stripped
num=*index*	One-based sequence number that identifies the occurrence of the sequence to be stripped.
all=**yes** | **no**	Specification for whether all instances (**yes**) of the sequence are replaced or only the first instance (**no**) of the sequence.
case= **yes** | **no**	Specification for whether the comparison to match the old sequence is case sensitive (**yes**) or case insensitive (**no**). [Default is **no**.]

Table 6.42 **strip** Command Parameters

Examples

The following **strip** command examples use a predefined variable, **mystring**, with a value of **abcabcabc**.

```
strip mystring ab all=yes
echo "$(mystring)"
ccc

strip mystring abc num=1
echo "$(mystring)"
abcabc
```

6.48 substring

Syntax

```
substring  variable_name string {pos=position | startseq=sequence
           [startseqnum=number]} {length=length | endseq=sequence
           [endseqnum=number]} [case=yes|no]
```

Description

The **substring** command finds a substring in an existing string and stores it in a variable.

The first parameter, *variable_name*, specifies the name of the variable into which the substring is placed. The *string* parameter specifies the string from which the substring is taken.

The beginning of the **substring** is marked either by a position or a start sequence and optional sequence occurrence number (similar to deletestring). The end of the substring is determined by specifying the length of the substring or an ending sequence and optional sequence occurrence number, also similar to deletestring.

The **case** parameter specifies whether comparisons for start and end sequences are case sensitive (**yes**) or case insensitive (**no**). [Default is **no**.]

_lastrc.message is set to NO_MATCH if a start or end sequence was specified and could not be found. If the position or length are specified and contain invalid values, **_lastrc.message** will contain INVALID_VALUE.

Parameters

Parameter	Description
variable_name	Name of the existing variable into which the substring is placed.
string	String from which the substring is taken.
pos=position	Starting position of string to be taken (one based index)
startseq=*sequence*	Starting position of string to be taken (following a specific character sequence)
startseqnum=*number*	Occurrence number of starting position of string to be taken (following a specific character sequence)
length=*length*	Length of the string to be taken
endseq=*sequence*	Ending position of string to be taken (following a specific character sequence)
endseqnum=*number*	Occurrence number of ending position of string to be taken (following a specific character sequence
case=**yes** | **no**	Specification for whether or not the comparisons of the start and end sequences are case sensitive (**yes**) or case insensitive (**no**). [Default is **no**.]

Table 6.43 **substring** Command Parameters

Examples

The following examples illustrate the **substring** command:

```
substring res "This is fun" pos=6 length=2
echo "$(res)"
is

substring res "This is less fun" startseq=" " endseq=" "
echo "$(res)"
is

substring res "This This is fun fun" startseq=" " endseq=" " +
         endseqnum=3
echo "$(res)"
This is fun
```

6.49 truncate

Syntax

```
truncate variable_name length=length
```

Description

The **truncate** command truncates a string to a specific length.

The first parameter, *variable_name*, is the name of an existing variable or list element to truncate. The *length* parameter specifies the length to which the string is truncated.

Parameters

Parameter	Description
variable_name	Name of the existing variable or list element to truncate.
length=*length*	Length to which the string truncates.

Table 6.44 **truncate** Command Parameters

Examples

The following examples assume that a variable, **mystring**, exists with a beginning value of **12345789**.

```
truncate mystring length=4
echo "$(mystring)"
1234
```

```
truncate mystring length=8
echo "$(mystring)"
12345678
```

6.50 upper

Syntax

```
upper variable_name
```

Description

The **upper** command forces all alphabetic characters in a specified variable or list element (*variable_name*) to upper case.

Parameters

Parameter	Description
variable_name	Variable or list element in which to force alphabetic characters to upper case.

Table 6.45 **upper** Command Parameters

Example

The following example assumes that a variable, **myvar**, exists with a beginning value of **abcXYZ**:

```
set myvar=abcXYZ
upper myvar
echo $(myvar)
```

The output would be:

ABCXYZ

Chapter 7
Additional Information

7.1 Overview

This chapter identifies additional information relative to Universal Data Mover (UDM).

Table 7.1 identifies this information and provides a link to its location in this chapter.

Information	Description	Page
Common File System Attributes	Attributes common to UDM on most operating systems.	264
z/OS Dynamic Allocation Attributes	z/OS dynamic allocation attributes that can be specified via the attrib (Attribute) command.	265
z/OS Attributes for Allocating Temporary Data Sets when Copying Load Modules	z/OS attributes that can be used for allocating temporary data sets when copying load modules via the copy command.	268
OS/400-Specific File System Attributes	Attributes that are unique to the LIB file system for OS/400.	269
Built-in Variables	All UDM built-in variables.	272
Global Variable Attributes	Attributes that can be used with the following types of UDM global variables: • All variables (user-defined or built-in) • Specific built-in variables • Logical Name built-in variables	275
Statements	Statements that can be used in the UDM scripting language.	277
`if` Statement Comparators	Comparators that can be used in an **`if`** statement to determine the type of comparison to be made between the left-hand and right-hand values.	278
Command Expression Operators	Operators for UDM command expressions.	279
Character Code Pages	Character code pages available for use with UDM.	281
SSL Cipher Suites	SSL Cipher Suites provided for use with UDM.	283
DD Statements	DD statements used in the UDM Manager Batch JCL.	284

Table 7.1 Universal Data Mover - Additional Information

7.2 Common File System Attributes

Table 7.2 provides information on file system attributes that are common to UDM on most operating systems.

Attribute Name	Values	Description
createop	**append**, **new**, or **replace**	Determines how the file is to be created. If the value is **append**, the transferred data is appended to the data already in the destination file, if it exists. If the destination file does not exist, it will be created. If the value is **new**, the UDM will copy the file only if the destination file does not already exist. If the destination file does exist at the time the copy operation is initiated, the operation will return with an error. If the value is **replace**, UDM will overwrite the destination file it already exists. Otherwise UDM will create the file. The default value is **new**.
defext	Any sequence of characters valid for the destination file system.	If the source filename is being implicitly used as the destination filename, the sequence of characters specified by this attribute is appended to the end of the filename used to write the destination file. This occurs after the file extension has been truncated (if **truncext** is set to **yes**). By default, no default extension is defined. **Note:** The sequence of characters is appended verbatim. UDM does not add a dot character before the sequence, so if one is desired, it must be explicitly specified.
eol	Any sequence of valid text data	Specifies the end of line sequence used in text transfers. For the source side of a copy operation, excepting those from the z/OS **dd** and **dsn** file systems and the OS/400 LIB file system, the end of line sequence is used to determine the end of each line of data. When the specified the sequence occurs in the data, UDM considers all data read up to that point (starting from the previous line) as a single line. Each line is transferred without the end of line sequence. On the destination side of a text transfer, the end of line sequence is appended to the end of each line before it is written. Two special character sequences can be used in any end of line sequence: • **\r** sequence indicates a carriage return. • **\n** sequence indicates a line feed. The default value of **eol**depends on the platform and the file system: • Under Windows, the default value is **\r\n**. • For UNIX platforms and the HFS file system under USS, the default value is \n. • Under z/OS for the **dd** and **dsn** file systems, and under OS/400 for the LIB file system, the **eol** attribute is undefined. • Under OS/400 for the **hfs** file system, the default is **`FILE`**, which makes end-of-line terminator consistent with file **`ccsid`**.

Attribute Name	Values	Description
linelen	A positive integer	Determines the maximum length of each line of data (record under the z/OS **dd** and **dsn** file systems) written. It applies only to the destination side of a transfer and is used in conjunction with the **lineop** and **padline** attributes. By default, **linelen** has a value of zero. Under Windows, UNIX, and the **hfs** file system, this means no line operation takes place. Under the z/OS **dd** and **dsn** file systems, the value **linelen** will be set equal to the logical record size used in allocating the destination file.
lineop	**none**, **stream**, **wrap**, or **trunc**	Sets the line operation for transferred lines (records under the z/OS **dd** and **dsn** file systems). For the line operation to be in play in the transfer, the value of **linelen** must not be zero (**linelen** is set automatically for the z/OS **dd** and **dsn** file systems to the logical record size if it is zero). If the value is **none**, each source line or record or data is written as it is received as a complete line or record. If the length of the source line is greater than that specified by **linelen**, UDM issues an error and the transfer operation is aborted. If the value is **stream**, the source data is treated as one long, single line of data. The source data is broken into multiple lines (records), each with a length of that specified by **linelen**. If the value is **trunc**, each source line longer than the value specified by **linelen** is truncated to be exactly **linelen** characters long. If the value is **wrap**, each source line longer than the value specified by **linelen** is broken up into multiple lines, each no longer than **linelen** characters long. Each segment is written out as a separate line (record). **Note:** If an end of line sequence is specified, the length of the sequence is not considered by UDM when determining the length of a line on the destination side. UDM only looks at the raw data that is transferred. By default the **lineop** attribute is not defined.
mode	Set of three numbers (0-7) or nothing	Note: This attribute is used only for UDM for UNIX. Specification for the mode (in UNIX parlance), or file permissions, of a file created by UDM in a copy operation. Existing files do not have their modes modified by UDM. They retain the file mode that they had before the copy operation was initiated. Each number in the set corresponds to one or more individuals for whom access is granted for the file: • First number Owner of the file. • Second number Users in the group assigned to the file. • Third number Everyone else. The value of each number is the sum of values representing file permissions: • 0 No permissions. • 1 Permission to execute the file. • 2 Permission to write to the file. • 4 Permission to read from the file. [By default, the mode attribute is not set. The default mode of a newly created file by UDM is dependent upon the user's umask or the mode of the source file in a UDM transfer.]

Attribute Name	Values	Description
padline	**none**, **null**, or **space**	Used in conjunction with **linelen**, when **linelen** is not zero. If the value is **none**, each line (record) of data written is not padded. If the value is **null**, each line of data is padded with null characters (hex value 0) at the end until the line is **linelen** characters in length. If the value is **space**, each line of data is padded with spaces at the end until the line is **linelen** characters in length. The default value is **none**.
trans	**yes** or **no**	Indicates whether or not a transactional file copy is to be performed. If the value is **yes**, the file is copied to a file with a temporary name that is renamed to the destination filename once the file has been successfully transferred. If the value is **no**, the file is copied directly to the specified destination filename. The default value is **no**. For z/OS and OS/400, **`trans`** is valid only under the **hfs** file system.
truncext	**yes** or **no**	Indicates whether or not the source filename's extension should be truncated if it is being used as the destination filename (no filename was explicitly specified on the destination side of the transfer operation). If the value of this attribute is **yes**, the extension is truncated. If the value is **no**, the filename is left untouched. The default value is **no**. **Note:** UDM considers a file extension to be the sequence of characters following the last dot (.) character in the filename. When an extension is truncated, the dot marks the beginning of the extension and is truncated as well. UDM will not consider the a dot character as the first character in a filename as indicating a file extension.
umask	A three-digit octal value	Valid only on the destination side of the transfer for the UNIX platform and the z/OS HFS file system (**hfs**). Specifies the file permissions mask used to create the destination file. Refer to the UNIX man page umask(1) for complete details. By default, the **umask** attribute is not defined. Note: In a UNIX-to-UNIX transfer, if the destination file does not exist, UDM will set the file permissions to match that of the source file once transfer has completed.
usefqn	**yes** or **no**	Valid only on the source side of the transfer for the z/OS platform. Specifies whether when copying a data set under the DSN file system, the fully qualified data set name is sent over as the source file name to be used by the destination if an explicit destination filename is not given. If set to no, only the part of the data set name matching the source mask in the copy operation is used as the destination filename. The default value is **no.**

Table 7.2 Common File System Attributes

For an explanation of how these attributes are used, see Chapter 12 UDM Transfer Operations of the Universal Data Mover 3.2.0 User Guide.

7.3 z/OS Dynamic Allocation Attributes

Table 7.3, below, lists the z/OS dynamic allocation attributes that can be specified via the attrib (Attribute) command.

(For complete details on an allocation attribute, refer to the IBM JCL Reference.)

Attribute Name	Description
abnormaldisp	• Disposition of a data set after the job ends abnormally. • Equivalent to the third position sub-parameter of the JCL DISP parameter: DISP=(status,normaldisp,abnormaldisp). • Default is DELETE: it can be set with the UDM configuration option ALLOC_ABNORMAL_DISP.
avgrec	• Indication that the unit of allocation space specified with the **spaceunit** attribute is records and that the primary space and secondary space values are in units of 1's, K's, or M's. • Equivalent to the JCL AVGREC parameter: AVGREC=size. • No default or configuration option.
blksize	• Block size with which the data set is allocated. • Equivalent to the JCL BLKSIZE parameter: BLKSIZE=size. • Default is 27998: it can be set with the UDM configuration option ALLOC_BLKSIZE.
blkszlim	• Block size limit when there is not block size specified from any source. • Equivalent to the JCL BLKSZLIM parameter: BLKSZLIM=size. • No default or UDM configuration option.
dataclas	• SMS data class name. • Equivalent to the JCL DATACLAS parameter: DATACLAS=name. • No default: it can be set with the UDM configuration option ALLOC_DATACLAS.
datasetseq	• Data set sequence number that specifies the relative position of a tape data set on the volume. • Equivalent to the data set sequence sub-parameter of the JCL LABEL parameter: LABEL=(datasetseq,,,,). • No default or UDM configuration option.
ddndcbref	• DCB reference to a ddname. • Equivalent to the ddname sub-parameter of the JCL DCB parameter: DCB=ddname. • No default or UDM configuration option.
den	• Tape density to use. • Equivalent to the DEN sub-parameter of the JCL DCB parameter: DCB=DEN=density. • No default or UDM configuration option.
dirblocks	• Number of directory blocks to allocate for a partitioned data set. • Equivalent to the third positional parameter of the second positional parameter of the JCL SPACE parameter: SPACE=(, (,, dirblocks), . . .). • Default is 20: it can be set with the UDM configuration option ALLOC_DIR_BLOCKS.
dsndcbref	• DCB reference to a cataloged data set name. • Equivalent to the data set name sub-parameter of the JCL DCB parameter: DCB=dsn. • No default or UDM configuration option.

Attribute Name	Description
dsntype	• Type of SMS data set to allocate. • Equivalent to the JCL DSNTYPE parameter: DSNTYPE=type. • No default or UDM configuration option.
dsorg	• Data set organization with which the data set is allocated. • Equivalent to the JCL DSORG parameter: DSORG=org. • Default is PS: it can be set with the UDM configuration option ALLOC_DSORG.
expdt	• Expiration date of the data set. • Equivalent to the JCL EXPDT parameter: EXPDT=date. • No default or UDM configuration option.
label	• Data set label type used for mostly tape data sets. • Equivalent to the label sub-parameter of the JCL LABEL parameter: LABEL=(,label,,,). • No default or configuration option.
like	• SMS data set name from which to model data set attributes. • Equivalent to the JCL LIKE parameter: LIKE=dsname. • No default or UDM configuration option.
lrecl	• Logical record length with which the data set is allocated. • Equivalent to the JCL LRECL parameter: LRECL=len. • Default is 1024: it can be set with the UDM configuration option ALLOC_LRECL.
mgmtclas	• SMS management class name. • Equivalent to the JCL MGMTCLAS parameter: MGMTCLAS=name. • No default: it can be set with the UDM configuration option ALLOC_MGMTCLAS.
normaldisp	• Disposition of a data set after the job ends. • Equivalent to the JCL DISP parameter: DISP=(status,normaldisp,abnormaldisp). • Default is CATLG: it can be set with the UDM configuration option ALLOC_NORMAL_DISP.
password	• Password for password protected data sets. • No JCL equivalent.
primspace	• Primary amount of space to allocate for the data set. • Equivalent to the first positional parameter of the second positional parameter of the JCL SPACE parameter: SPACE=(, (primspace,), . . .). • Default is 15: it can be set with the UDM configuration option ALLOC_PRIM_SPACE.
pwdprotect	• Specification for whether or not the data set is password protected. • Equivalent to the PASSWORD or NOPWREAD sub-parameters of the JCL LABEL parameter: LABEL=(,,{PASSWORD|NOPWREAD},,). • Value must be either PASSWORD or NOPWREAD.
recfm	• Record format with which the data set is allocated. • Equivalent to the JCL RECFM parameter: RECFM=fmt. • Default is VB: it can be set with the UDM configuration option ALLOC_RECFM.
refdd	• ddname from which to copy SMS data set attributes. • Equivalent to the JCL REFDD parameter: REFDD=ddname. • No default or configuration option.
retpd	• Retention period of the data set. • Equivalent to the JCL RETPD parameter: RETPD=date. • No default or configuration option.

Attribute Name	Description
rlse	• Specification for whether or not to release unused space when the data set is unallocated. • Equivalent to the sub-parameter RLSE of the JCL SPACE parameter: SPACE=(,(,,),RLSE). • Default is no. There is no UDM configuration option. • Setting the attribute value to *yes* turns on the attribute.
secspace	• Secondary amount of space to allocate for the data set. • Equivalent to the second positional parameter of the second positional parameter of the JCL SPACE parameter: SPACE=(, (, secspace), . . .). • Default is 15: it can be set with the UDM configuration option ALLOC_SEC_SPACE.
spaceunit	• Allocation unit used to specify the space to allocate for the data set. • Equivalent to the first positional parameter of the JCL SPACE parameter: SPACE=(unit, . . .). • Default is TRK: it can be set with the UDM configuration option ALLOC_SPACE_UNIT.
status	• Status of the data set to be allocated. • Equivalent to the first positional parameter of the JCL DISP parameter: DISP=(status,normaldisp,abnormaldisp). • Default is: OLD for input and output data sets that exist, NEW for output data sets that don't exist. • Default input status can be set with UDM configuration option ALLOC_INPUT_STATUS. • Default output status can be set with UDM configuration option ALLOC_OUTPUT_STATUS.
storclas	• SMS storage class name. • Equivalent to the JCL STORCLAS parameter: STORCLAS=name. • No default: default value can be set with UDM configuration option ALLOC_STORCLAS.
unit	• Unit on which the data set is allocated. • Equivalent to the JCL UNIT parameter: UNIT=unit. • Default is SYSALLDA: it can be set with UDM configuration option ALLOC_UNIT.
unitcnt	• Number of units to allocate for a multi-volume data set. • Equivalent to the unit count sub-parameter JCL UNIT parameter: UNIT=(,unitcnt,). • No default or UDM configuration option.
volcnt	• Number of volumes to allocate for a multi-volume data set. • Equivalent to the volume count sub-parameter JCL VOL parameter: VOL=(,,,volcnt,). • No default or UDM configuration option.
volseq	• Volume sequence number on which a multi-volume data set starts. • Equivalent to the volume sequence number sub-parameter JCL VOL parameter: VOL=(,,volseq,,). • No default or UDM configuration option.
volser	• Volume serial number on which the data set is allocated. • Equivalent to the SER sub-parameter of the JCL VOL parameter: VOL=SER=volser. • No default: default value can be set with UDM configuration option ALLOC_VOLSER.

Table 7.3 z/OS attrib Command - Dynamic Allocation Attributes

7.4 z/OS Attributes for Allocating Temporary Data Sets when Copying Load Modules

Table 7.4, below, lists the z/OS attributes that can be used for allocating temporary data sets when copying load modules via the copy (Copy) command.

Attribute Name	Description
TMPVOLSER	Sets the volume on which temporary files will be allocated: • On the source side: specifies the location of the temporary sequential data set that will be transferred. • On the destination side: specifies the volume for the temporary transfer file as well as the volume used by the temporary staging PDS/E.
TMPPRIMSPACE	Specifies the amount of primary space used when allocating the temporary files. • On the source side: Affects the temporary sequential data set that will be transferred. • On the destination side: Used in allocating the temporary transfer file and the temporary staging PDS/E.
TMPSECSPACE	Specifies theamount of secondary space used when allocating the temporary files. • On the source side: Affects the temporary sequential data set that will be transferred. • On the destination side: Used in allocating the temporary transfer file and the temporary staging PDS/E.
TMPSPACEUNIT	Specifies the space unit used when allocating the temporary files. • On the source side: Affects the temporary sequential data set that will be transferred. • On the destination side: Used in allocating the temporary transfer file and the temporary staging PDS/E.
TMPDIRBLOCKS	Specifies the number of directory blocks used. • On the source side: n/a. • On the destination side: specifies the number of directory blocks used by the staging PDS/E.

Table 7.4 z/OS Attributes for Allocating Temporary Data Sets when Copying Load Modules

7.5 OS/400-Specific File System Attributes

7.5.1 LIB File System Attributes

Table 7.5 identifies attributes that are unique to the LIB file system for OS/400.

Name	Description	Source	Value	UDM Default	System Default	File Type
ACCPTH	Access path type		ARRIVAL, KEYED	NULL	ARRIVAL	SP
ALWDLT	Allow delete operation	√	YES, NO	NULL	YES	PF, SP
ALWUPD	Allow update operation	√	YES, NO	NULL	YES	PF, SP
ASPDEV	ASP device		ASP, ASPGRPPRI, SYSTEM, device name	NULL	ASP	LIB
ASPNUM	ASP number		LIBASP, 1-32, ASPDEV	NULL	LIBASP for SAVF, 1 for LIB	LIB, SAVF
AUT	Authority		LIBCRTAUT, ALL, CHANGE, EXCLUDE, USE [2]	NULL	LIBCRTAUT for LIB,PF,SP EXCLUDE for SAVF	LIB, PF, SP, SAVF
CCSID *	CCSID of the file (source physical files only). For data physical files, the DDS (if one is given) determines its CCSID; if no DDS is given, the value is 65535.	√	EBCDIC CCSIDs	CODEPAGE		SP
CRTAUT	Create authority		SYSVAL, ALL, CHANGE, EXCLUDE, USE, authority name	NULL		LIB
DDSLIB	Library of the DDS used to describe the file	√		empty string		PF
DDSFILE	File of the DDS used to describe the file	√		empty string		PF
DDSMBR	Member of the DDS used to describe the file	√		empty string		PF
DLTPCT	Maximum percentage of deleted records allowed		1-100, NONE	NULL	NONE	PF
EXPDATE	Expiration date for member	√	date, NONE	NULL	NONE	PF, SP
FILETYPE	Type of file to create when creating a new file	√	DATA, SRC, SAVF	DATA		PF, SP, SAVF

Name	Description	Source	Value	UDM Default	System Default	File Type
FRCRATIO	Records to a force write		integer, NONE	NULL	NONE	PF, SP
GENLVL	Generation severity level		0-30	NULL	20	PF
LIBTYPE	Type of library created when creating a library	√	PROD, TEST	PROD		LIB
LVLCHK	Record format level check	√	YES, NO	NULL	YES	PF
MAXMBRS	Maximum number of members	√	integer, NOMAX	NULL	1 for PF, NOMAX for SP	PF, SP
MAXRCDS	Maximum number of records		1-2146762800, NOMAX	NULL		SAVF
OPTION	Source listing options		SRC, NOSRC, SOURCE, NOSOURCE, LIST, NOLIST, SECLVL, NOSECLVL, EVENTF, NOEVENTF (up to four repetitions)	empty string		PF
RCDLEN	Record length if no DDS is used	√	integer	92		PF, SP
REUSEDLT	Reuse deleted records	√	YES, NO	NULL	NO	PF
SEQSTART	Beginning sequence number used when writing to a source physical file	√	0000.01 – 9999.99	1.00		SP
SEQINCR	Amount to increment sequence number by when writing a record to a source physical file	√	00.01 – 99..99	1.00		SP
SHARE	Share open data path	√	YES, NO	NULL	NO	PF, SP, SAVF
SIZE	Member size	√	• Single values: NOMAX • Other values: Comma-separated element list • Element 1: Initial number of records 1-2147483646, • Element 2: Increment number of records Integer, • Element 3: Maximum increments Integer (EX: 10000,1000,3)	size_attrib configuration file entry if provided; otherwise empty string	10000,1000, 3 for PF, 10000,1000, 499 for SP	PF, SP

Name	Description	Source	Value	UDM Default	System Default	File Type
USESRCSEQ	Sequence number and modification date information: • On Source side: retain this information when copying a source physical file • On Destination side: Record data includes this information	√	YES, NO	NO		SP
WAITFILE	Maximum file wait time		integer, IMMED, CLS	NULL	30 for PF IMMED for SP, SAVF	PF, SP, SAVF
WAITRCD	Maximum record wait time		integer, IMMED, NOMAX	NULL	60	PF, SP
* With CCSID set to CODEPAGE, when the UDM CCSID attribute is not set either explicitly or implicitly via an OS/400 to OS/400 file transfer, the CCSID associated with the code page via the code page to CCSID mapping tables gets used as the CCSID attribute value. One implication is that, by default, files may be created with the CCSID associated with the codepage option.						

Table 7.5 OS/400-Specific LIB File Attributes for Creating New Files

7.5.2 HFS Attributes

Table 7.6 identifies attributes that are unique to the HFS file system for OS/400. (Currently, there is only one HFS unique attribute, CCSID.)

Name	Description	Source	Value	UDM Default	System Default	File Type
CCSID	CCSID of the file	√	EBCDIC and ASCII CCSIDs	CODEPAGE		stream

Table 7.6 OS/400-Specific HFS File Attributes for Creating New Files

7.6 Built-In Variables

Table 7.7 lists all of the UDM built-in variables.

Variable	Description
_date	Displays the current date in the format appropriate for the system's locale.
_echo	Specification for whether or not a command is echoed prior to processing.
_execrc	Holds the value of the process executed by the last exec command issued.
_file	Name of the file for the current iteration in a **`forfiles`** loop.
_halton	Return code value that causes UDM to terminate if it is greater than 0 and is equalled or exceeded by the return code value in the _rc variable.
_keepalive	Interval at which keep-alive messages are sent form the UDM Manager to transfer servers.
_lastmsg	Contains all of the messages written in the transaction log for the last network- or file-oriented command issued.
_lastrc	Holds the return code of the last command issued and, optionally, an indication of what happened with the last executed statement.
_lines	Specification for whether or not the line number is printed with the error if a command cannot be parsed or is malformed.
_path	Absolute path of the file for the current iteration in a **`forfiles`** loop.
_rc	Current UDM return code.
_time	Current time.
_uuid	Generates a UUID.

Table 7.7 Built-In Variables

For an explanation of how these variables are used, see Section 11.6 UDM Variables in the Universal Data Mover 3.2.0 User Guide.

7.7 _file Built-in Variable – Special Attributes

Table 7.8 lists all of the special attributes for the _file built-in variable.

Attribute Name	Description
accessdate	Date on which the file was last accessed. Format (ISO 8601) is yyyy-mm-dd.
accesstime	Time when the file was last accessed. Format (ISO 860) is hh:mm:ss.
accesstimestamp	Combination of **accessdate** and **accesstime**: yyyy-mm-dd hh:mm:ss. If the file does not have an access time, but does have a access date, 00:00:00 is used for the time portion.
createdate	Date on which the file was created. Format (ISO 8601) format of yyyy-mm-dd.
createtime	Time when the file was created. Format (ISO 8601) is hh:mm:ss.
createtimestamp	Combination of **createdate** and **createtime**: yyyy-mm-dd hh:mm:ss. If the file does not have a creation time, but does have a creation date, 00:00:00 is used for the time portion.
moddate	Date on which the file was last modified (referenced for z/OS). Format (ISO 8601) is yyyy-mm-dd.
modtime	Time when the file was last modified (referenced for z/OS). Format (ISO 8601) is hh:mm:ss.
modtimestamp	Combination of **moddate** and **modtime**: yyyy-mm-dd hh:mm:ss. If the file does not have a modification time, but does have a modification date, 00:00:00 is used for the time portion.
name	Name of the file (same as referencing **_file** itself without any attributes).
size	Size of the file (in bytes).
type	Type of file. Values are: • file • directory (also used for PDSs under z/OS) • unknown **type** has meaning in a **forfiles** statement under OS/400 in the LIB file system: • If the value of _file.type is directory, the file type is a Physical file. • If the value of _file.type is file, the file type is a Save file.

Table 7.8 _file Built-in Variable – Special Attributes

7.8 Global Variable Attributes

Table 7.9 lists the attributes that can be used with the following types of UDM global variables:

- All variables (user-defined or built-in)
- Specific built-in variables
- Logical Name built-in variables

Variable	Attribute	Attribute Description
(all)	exists	Expands to **yes** if a variable with that name exists at any scope; it expands to **no** if no variable with that name exists.
(all)	length	Expands to the length of the variable's value.
_date	day	Resolves to the day of the week.
_date	month	Resolves to the current month.
_date	dd	Resolves to a two-digit day of the month.
_date	ddd	Resolves to the Julian day.
_date	mm	Resolves to the two-digit month of the year.
_date	yy	Prints the two-digit year.
_date	ww	Resolves to the two-digit current week of the year. (The value of ww is zero-based, not one based. That is, the first week of the year is 0, the second week is 1, the third week is 2, and so on.)
_date	yyyy	Resolves to the four-digit year.
_file	type	Type of file contained in the **`file`** variable: file, directory (also used for PDSs under z/OS), or unknown. **`type`** also has meaning in a **`forfiles`** statement under OS/400 in the LIB file system: • If the value of **`_file.type`** is **`directory`**, the file type is a Physical file. • If the value of **`_file.type`** is **`file`**, the file type is a Save file.
_lastrc	message	Human-readable string indicating what happened with the last executed statement. • If a command could not be executed or had improper values, the value of **`_lastrc.message`** is ERROR. • If a command successfully executed, the value of **`_lastrc.message`** is either SUCCESS or some other message (depending upon the command).
_lastrc	result	Integer value that indicates the result of the last command executed. The meaning of this value depends on the command. Unless otherwise stated: • -1 indicates failure. • 0 or a positive value indicates success.
_time	hh	Resolves to the two-digit hour (24-hour time).
_time	mm	Resolves to the two-digit minute.
_time	ss	Number of seconds that have elapsed since the current minute.

Variable	Attribute	Attribute Description
_time	hs	Resolves to the number of hundredths of a second that have elapsed since the last second.
(logical name)	host	Contains the host name of the transfer server.
(logical name)	port	Holds the port used to connect to the transfer server over.
(logical name)	user	Contains the userid used to sign into the transfer server.

Table 7.9 UDM Global Variable Attributes

For an explanation of how these attributes are used in the variables, see Section 11.6 UDM Variables in Chapter 11 UDM Scripting Language of the Universal Data Mover 3.2.0 User Guide.

7.9 UDM Statements

Table 7.10 lists all of the statements that can be used in the Universal Data Mover scripting language.

Statement	Description
if	Adds conditional branching of UDM commands. An **if** statement consists of: 1. Comparison operation. 2. Series of UDM commands that are carried out if the comparison operation evaluates to true. 3. **end** statement that indicates the end of the **if** statement. A comparison consists of three parts: 1. Left-hand value 2. Comparator (see Section 7.10 if Statement Comparators) 3. Right-hand value The left-hand and right-hand values can be either: • Variable reference • Variable attribute • Constant The syntax is: **if comparison** … **UDM commands** … **end** If the comparison does not evaluate to true, UDM picks up execution from the line after the **end** statement.
else	Provides an alternate path, when used as part of an **if** statement, if the comparison evaluates to false. The syntax is: **if expression** … **[else** **…]** **end** In this **if** statement, the parameter for **if** is an expression. If the expression evaluates to a value that is not equal to zero, the positive branch is taken; otherwise, the negative (**else**) branch is taken.

Statement	Description
while	Implements a simple **while** loop. The syntax is: **while expression** ... **end** In this case, the loop iterates (executing the commands between the while and end statements) as long as the expression evaluates to a value that is not zero. If the expression evaluates to a value of zero, code execution picks up at the point immediately following the end of the **while** loop.
fordata	Iterates through a data element, once for each line. For each iteration, a variable provided by the user is set to hold the contents of the line in the data element corresponding to the current iteration. The syntax is: **fordata variable-name=data-element** ... **end**
forfiles	Iterates through a series of statements for each file found that matches a given file specification. The syntax is: **forfiles logical_name=file_spec** **[sortby=attribute-name[,ascending|descending]]** ... **UDM commands** ... **end**
subroutine	Names a subroutine and defines the script code that becomes associated with that subroutine name. The syntax is: **subroutine name** **[script line 1}** ... **{script line 2]** **endsub**
callsub	Carries out the work of lines of script associated with a subroutine. The syntax is: **callsub name**

Table 7.10 UDM Statements

For an explanation of how these statements are used, see Section Chapter 11 UDM Scripting Language in the Universal Data Mover 3.2.0 User Guide.

7.10 **if** Statement Comparators

Table 7.11 lists all of the comparators that can be used in an `if` statement to determine the type of comparison to be made between the left-hand and right-hand values.

Comparator	Description
EQ (Equal)	Evaluates to true if the left-hand and right-hand values are equal. If one or more of the values contains alpha characters (non-numeric), the comparison is case insensitive.
NE (Not Equal)	Evaluates to true if the left-hand value is not equal to the right-hand value. If one or more of the values contains alpha characters (non-numeric), the comparison is case insensitive.
LT (Less Than)	Evaluates to true if the left-hand value is less than the right-hand value. LT performs a numeric comparison.
GT (Greater Than)	Evaluates to true if the left-hand value is greater than the right-hand value. GT performs a numeric comparison.
LE (Less Than or Equal)	Evaluates to true if the left-hand value is less than or equal to the right-hand value. LE performs a numeric comparison.
GE (Greater Than or Equal)	Evaluates to true if the left-hand value is greater than or equal to the right-hand value. GE performs a numeric comparison.

Table 7.11 if Statement Comparators

For an explanation of how these comparators are used, see Section 11.7.1 Comparison Operations in Section 11.7 if Statement of the Universal Data Mover 3.2.0 User Guide.

7.11 UDM Command Expression Operators

Table 7.12 identifies and describes all of the operators for UDM command expressions.

Operator	Description
EQ	Compares the value on the left to the value on the right. If the two are equal, the result is 1, otherwise it is 0. Both the left an right values can be strings or numbers. If they are strings, the comparison is case insensitive.
NE	Works like the equal operator, except that it results in 1 if the left and right value are not equal and 0 if they are.
LT	Results in a value of 1 if the left value is less than the right value, otherwise it results in 0. This is a numeric operator.
GT	Results in a value of 1 if the left value is greater than the right value, otherwise it results in 0. This is a numeric operator.
LE	Results in a value of 1 if the left value is less than or equal to the right value, otherwise it results in 0. This is a numeric operator.
GE	Results in a value of 1 if the left value is greater than or equal to the right value, otherwise it results in 0. This is a numeric operator.
AND	Results in a value of 1 if both the left value and right value are not 0, otherwise it results in 0.
OR	Results in a value of 1 if either the left value or the right value are not 0, otherwise it results in 0.
XOR	Results in a value of 1 if either the left or the right values are not 0, but not both. If both the left and right values are 0, then the result of the XOR operator is 0.
NOT	Unlike all of the operators, the NOT operator has only one operand that appears to the right of the NOT operator. This operation evaluates to one if the operand is zero and zero if the operand is non-zero.
+	Result is the sum of the left and right values.
-	Result is subtracting the right value from the left value.
*	Result is the product of the left and right values.
/	Result is the left value divided by the right value. Assuming integer-only math, the remainder is discarded.
%	Result is the remainder of the left value divided by the right value.

Table 7.12 UDM Command Expressions - Operators

7.12 Character Code Pages

Table 7.13 identifies the character code pages provided by Stonebranch Inc. for use with Universal Products on each supported operating system.

Code Page	CCSID	z/OS	UNIX	Windows	OS/400		HP NonStop
					HFS	LIB	
IBM037	037	√			√	√	
IBM273	273	√			√	√	
IBM277	277	√			√	√	
IBM278	278	√			√	√	
IBM280	280	√			√	√	
IBM284	284	√			√	√	
IBM500	500	√			√	√	
IBM875	875	√					
IBM1047							
IBM1140	1140	√			√	√	
IBM1141	1141	√			√	√	
IBM1142	1142	√			√	√	
IBM1143	1143	√			√	√	
IBM1144	1144	√			√	√	
IBM1145	1145	√			√	√	
IBM1146	1146	√			√	√	
IBM1147	1147	√			√	√	
IBM1148	1148	√			√	√	
IBM4971	4971	√					
ISO8859-1	819		√	√	√		√
ISO8859-2	912		√	√	√		√
ISO8859-3	913		√	√	√		√
ISO8859-4	914		√	√	√		√
ISO8859-5	915		√	√	√		√
ISO8859-6	1089		√	√	√		√
ISO8859-7	813		√	√	√		√
ISO8859-8	916		√	√	√		√
ISO8859-9	920		√	√	√		√
ISO8859-10			√	√	√		√
ISO8859-13	921		√	√	√		√
ISO8859-14			√	√	√		√
ISO8859-15	923		√	√	√		√
PC437	437			√	√		

Code Page	CCSID	z/OS	UNIX	Windows	OS/400		HP NonStop
					HFS	LIB	
PC737	737			√	√		
PC775	775			√	√		
PC850	850			√	√		
PC852	852			√	√		
PC855	855			√	√		
PC857	857			√	√		
PC860	860			√	√		
PC861	861			√	√		
PC862	862			√	√		
PC863	863			√	√		
PC864	864			√	√		
PC865	865			√	√		
PC866	866			√	√		
PC869	869			√	√		
PC874	874			√	√		
WIN1250	1250			√	√		
WIN1251	1251			√	√		
WIN1252	1252			√	√		
WIN1253	1253			√	√		
WIN1254	1254			√	√		
WIN1255	1255			√	√		
WIN1256	1256			√	√		
WIN1257	1257			√	√		
WIN1258	1258			√	√		

Table 7.13 Character Code Pages

7.13 UTT Files

Table 7.14 identifies the Universal Translate Table (UTT) files that are used to translate between Unicode and the local single-byte code page.

Operating System	UTT File Location
OS/400	UTT files are located in the source physical file **UNIVERSAL/UNVNLS**. *codepage* is the member name of the UTT file.
z/OS	UTT files are located in the library allocated to the **UNVNLS** ddname. *codepage* is the member name of the UTT file.
UNIX	UTT files are located in the **nls** subdirectory of the installation directory. *codepage* is the base file name of the UTT file. All UTT files end with an extension of **.utt**.
Windows	UTT files are located in the NLS subdirectory of the installation directory. *codepage* is the base file name of the UTT file. All UTT files end with an extension of **.utt**.

Table 7.14 UTT File Locations

7.14 SSL Cipher Suites

Table 7.15 identifies all of SSL cipher suites provided by Stonebranch Inc. for use with UDM.

Cipher Suite	Description
RC4-SHA	128-bit RC4 encryption and SHA-1 message digest
RC4-MD5	128-bit RC4 encryption and MD5 message digest
AES256-SHA	256-bit AES encryption and SHA-1 message digest
AES128-SHA	128-bit AES encryption and SHA-1 message digest
DES-CBC3-SHA	128-bit Triple-DES encryption and SHA-1 message digest
DES-CBC-SHA	128-bit DES encryption and SHA-1 message digest
NULL-SHA	No encryption and SHA-1 message digest
NULL-MD5	No encryption and MD5 message digest
NULL-NULL *	UDM pseudo cipher used in conjunction with the open command encrypt option to disable SSL.
* UDM Manager ignores the NULL-NULL pseudo cipher.	

Table 7.15 SSL Cipher Suites for UDM

7.15 DD Statements

Table 7.16 describes the DD statements used in the UDM Manager batch JCL.

DD name	DCB Attributes	Mode	Description
STEPLIB	DSORG=PO, RECFM=U	input	Universal Products load library containing the program being executed.
SYSIN	DSORG=PS, RECFM=(F, FB, V, VB)	input	Standard input file for the UDM program. UDM reads its command options from SYSIN.
SYSOUT	DSORG=PS, RECFM=(F, FB, V, VB)	output	Standard error file for the UDM program.
SYSPRINT	DSORG=PS, RECFM=(F, FB, V, VB)	output	The standard output file for the UDM program.
UNVCLIB	DSORG=PO, RECFM=(F, FB, V, VB)	input	UDM call library: UDM searches for script files specified on the `call` command in this library.
UNVCONF	DSORG=PS, RECFM=(F, FB, V, VB)	input	Universal Data Mover configuration member.
UNVNLS	DSORG=PO, RECFM=(F, FB, V, VB)	input	Universal Products national language support library. Contains message catalogs and code page translation tables.
UNVSCR	DSORG=PS, RECFM=(F, FB, V, VB)	input	UDM command script: UDM executes the script allocated to this ddname.
UNVTRACE	DSORG=PO, RECFM=(F, FB, V, VB), LRECL=256 or above	output	Universal Products trace PDF. This ddname is used only if UNVTRMDL is not defined.
UNVTRMDL	DSORG=PS, RECFM=(F, FB, V, FB)	input	Data set used as a model for creating UCMD and USAP trace files when they are called by UDM using the exec and execsap commands.
UNVUCCF	DSORG=PS, RECFM=(F, FB, V, VB)	input	Universal Command configuration member.
UNVUSCF	DSORG=PS, RECFM=(F, FB, V, VB)	input	Universal SAP connector configuration member.
UNVUSRC	DSORG=PS, RECFM=(F, FB, V, VB)	input	Universal SAP connector RFC member.
The C runtime library determines the default DCB attributes. Refer to the IBM manual *OS/390 C/C++ Programming Guide* for details on default DCB attributes for stream I/O.			

Table 7.16 Universal Data Mover Batch JCL DD Statements

Appendix A
Customer Support

Stonebranch, Inc. provides customer support, via telephone and e-mail, for Universal Data Mover and all Universal Products.

TELEPHONE

Customer support via telephone is available 24 hours per day, 7 days per week.

North America

(+1) 678 366-7887, extension 6

(+1) 877 366-7887, extension 6 [toll-free]

Europe

+49 (0) 700 5566 7887

E-MAIL

All Locations

support@stonebranch.com

Customer support contact via e-mail also can be made via the Stonebranch website:

www.stonebranch.com

950 North Point Parkway, Suite 200
Alpharetta, Georgia 30005
U.S.A.