Command Line Interface for IBM Aspera products ascli 4.25.0.pre

Laurent MARTIN

2025/10/09

Contents

_	thirduction	0
	1.1 BUGS, FEATURES, CONTRIBUTION	9
	1.2 When to use and when not to use	9
	1.3 Notations, Shell, Examples	9
2	Quick Start	11
	2.1 First use	11
	2.2 Going further	12
	2.2 Going furtile:	12
2	Installation	13
3		
	3.1 Single file executable	13
	3.1.1 Linux: GLIBC version	14
	3.2 Ruby	14
	3.2.1 Windows: Installer	14
	3.2.2 macOS: brew	15
	3.2.3 Linux: Package	15
		16
	3.2.4 Unix-like: RVM: Single user installation (not root)	
	3.2.5 Unix-like: rbenv	16
	3.2.6 Other Unixes (AIX)	16
	3.2.7 JRuby	17
	3.2.8 Optional gems	17
	3.3 Ruby Gem: aspera-cli	18
	3.3.1 Gem installation with signature verification	18
	3.3.2 Beta release of gem	18
	3.4 FASP Protocol: ascp	18
	3.4.1 Installation of ascp through transferd	18
	3.4.2 Installation of ascp through other component	19
	3.5 Installation in air gapped environment	20
	3.5.1 Gem files and dependencies	20
	3.5.2 Unix-like	20
		20
	3.5.3 Windows	
	3.6 Container	21
	3.6.1 Container: Quick start	21
	3.6.2 Container: Details	21
	3.6.3 Container: Sample start script	22
	3.6.4 Container: Offline installation	23
	3.6.5 Container: aspera.conf	23
	3.6.6 Container: Singularity	23
	3.7 SSL library	
		23
	3.8 SSL CA certificate bundle	24
_	No. 11 to the first	~ ~
4	Command Line Interface	26
	4.1 ascp command line	26
	4.2 Command line parsing, Special Characters	27
	4.2.1 Shell parsing for Unix-like systems: Linux, macOS, AIX	27
	4.2.2 Shell parsing for Windows	27
	4.2.3 Shell parsing for Windows: cmd.exe	28
		28
	4.2.4 Shell parsing for Windows: Powershell	28

	4.2.5	Extended Values (JSON, Ruby,)	28
			28
	4.2.0		
	4.2.7		29
	4.2.8		29
	4.2.9		29
	4.2.10	Reading special characters interactively	30
	4211	Command line arguments from a file	30
	1212	Extended value using special characters read from environmental variables or files	30
4.3			31
4.5			
			31
			32
4.4	Interac	ctive Input	33
4.5	Output	3	33
			33
		//	33
	4.5.3		33
	4.5.4	Option: flat_hash : Single level Hash	34
	4.5.5	Enhanced display of special values	34
	4.5.6		35
	4.5.7	Option: display: Verbosity of output	36
	4.5.8	Option: show_secrets : Hide or show secrets in results	36
	4.5.9	· _	36
			37
	4.5.11		37
4.6	Extend	ed Value Syntax	37
4.7	Config	uration and Persistency Folder	39
4.8			10
		rary files	10
		,	‡0 10
4.10	_		
			10
	4.10.2	Special Option Preset: config	11
	4.10.3	Special Option Preset: default	11
	4104		11
4.11			12
			14
	4.11.2	Evaluation order of options	14
	4.11.3	Wizard	15
			15
112			16
4,12			16
		· · · · · · · · · · · · · · · · · · ·	
			17
			17
	4.12.4	· · · · · · · · · · · · · · · · · · ·	17
	4.12.5	Configuration Finder	17
	4.12.6	Securing passwords and secrets	17
4 13			18
7.10			18
		78	18
	4.13.3	openss1 4	18
	4.13.4	Using an application to generate a key pair	19
4.14			19
			19
			+9 19
			50
			51
4.19	HTTP 9	socket parameters	51
4.20	Proxv		52
			52
			52

	4.20.3 FASP proxy (forward) for transfers	. 53
	4.21 FASP configuration	
	4.21.1 Show path of currently used ascp	. 53
	4.21.2 Selection of ascp location for direct agent	. 53
	4.21.3 Installation of Connect Client on command line	. 54
	4.22 Transfer Clients: Agents	
	4.22.1 Agent: Direct	
	4.22.2 Agent: Connect Client	
	4.22.3 Agent: Desktop Client	
	4.22.4 Agent: Node API	. 59
	4.22.5 Agent: HTTP Gateway	. 59
	4.22.6 Agent: Transfer Daemon	
	4.23 Transfer Specification	
	4.24 Transfer Parameters	
	4.24.1 Destination folder for transfers	
	4.24.2 List of files for transfers	
	4.24.3 Source directory structure on destination	. 69
	4.24.4 Multi-session transfer	
	4.24.5 Content protection	
	4.24.6 Transfer Spec Examples	
	4.25 Transfer progress bar	
	4.26 Scheduler	
	4.26.1 Windows Scheduler	. 71
	4.26.2 Unix-like Scheduler	. 71
	4.27 Running as service	
	4.28 Locking for exclusive execution	
	4.29"Provençal"	
	4.30 faux: for testing	
	4.31 Usage	. 74
	4.32 Bulk creation and deletion of resources	. 79
	4.33 Option: query	
	4.33 Option: query	. 79
	4.34 Plugins	. 79 . 80
		. 79 . 80
5	4.34 Plugins 4.35 Plugins vs Transfer Agents	. 79 . 80 . 80
5	4.34 Plugins	. 79 . 80 . 80
5	4.34 Plugins	. 79 . 80 . 80 81 . 81
5	4.34 Plugins 4.35 Plugins vs Transfer Agents	. 79 . 80 . 80 81 . 81 . 82
5	4.34 Plugins 4.35 Plugins vs Transfer Agents	. 79 . 80 . 80 81 . 81 . 82
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details	. 79 . 80 . 80 81 . 81 . 82 . 82
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration	. 79 . 80 . 80 81 . 81 . 82 . 82
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud	. 79 . 80 . 80 81 . 81 . 82 . 82 . 82
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key	. 79 . 80 . 80 81 . 81 . 82 . 82 . 82 . 83
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc : IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration	. 79 . 80 . 80 81 . 81 . 82 . 82 . 82 . 83 . 83
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc : IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT	. 79 . 80 . 80 . 81 . 82 . 82 . 82 . 83 . 83 . 84
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc : IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 83 . 84 . 84
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use	. 79 . 80 . 81 . 81 . 82 . 82 . 82 . 83 . 83 . 84 . 84 . 85 . 85
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc : IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links	. 79 . 80 . 81 . 81 . 82 . 82 . 82 . 83 . 83 . 84 . 84 . 85 . 85
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use 5.3 Calling AoC APIs from command line	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 83 . 84 . 85 . 85
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use 5.3 Calling AoC APIs from command line 5.4 Administration	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 83 . 84 . 85 . 85 . 85
5	4.34 Plugins	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 84 . 85 . 85 . 85
5	4.34 Plugins	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 83 . 84 . 85 . 85 . 85 . 85
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use 5.3 Calling AoC APIs from command line 5.4 Administration 5.4.1 Listing resources 5.4.2 Selecting a resource 5.4.3 Creating a resource	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 83 . 84 . 85 . 85 . 85 . 85 . 86
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use 5.3 Calling AoC APIs from command line 5.4 Administration 5.4.1 Listing resources 5.4.2 Selecting a resource 5.4.3 Creating a resource 5.4.4 Access Key secrets	. 79 . 80 . 81 . 81 . 82 . 82 . 82 . 83 . 84 . 84 . 85 . 85 . 85 . 85 . 86 . 86
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc : IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use 5.3 Calling AoC APIs from command line 5.4 Administration 5.4.1 Listing resources 5.4.2 Selecting a resource 5.4.3 Creating a resource 5.4.4 Access Key secrets 5.4.5 Activity	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 83 . 84 . 85 . 85 . 85 . 85 . 86 . 86 . 87 . 87
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use 5.3 Calling AoC APIs from command line 5.4 Administration 5.4.1 Listing resources 5.4.2 Selecting a resource 5.4.3 Creating a resource 5.4.4 Access Key secrets	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 83 . 84 . 85 . 85 . 85 . 85 . 86 . 86 . 87 . 87
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use 5.3 Calling AoC APIs from command line 5.4 Administration 5.4.1 Listing resource 5.4.2 Selecting a resource 5.4.3 Creating a resource 5.4.4 Access Key secrets 5.4.5 Activity 5.4.6 Transfer: Using specific transfer ports	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 83 . 84 . 85 . 85 . 85 . 85 . 86 . 86 . 87 . 88
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use 5.3 Calling AoC APIs from command line 5.4 Administration 5.4.1 Listing resources 5.4.2 Selecting a resource 5.4.3 Creating a resource 5.4.4 Access Key secrets 5.4.5 Activity 5.4.6 Transfer: Using specific transfer ports 5.4.7 Using ATS	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 83 . 84 . 85 . 85 . 85 . 85 . 86 . 87 . 87 . 88 . 88
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use 5.3 Calling AoC APIs from command line 5.4 Administration 5.4.1 Listing resources 5.4.2 Selecting a resource 5.4.3 Creating a resource 5.4.4 Access Key secrets 5.4.5 Activity 5.4.6 Transfer: Using specific transfer ports 5.4.7 Using ATS 5.4.8 Files with type link	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 84 . 85 . 85 . 85 . 85 . 85 . 85 . 86 . 86 . 87 . 88 . 88 . 88
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use 5.3 Calling AoC APIs from command line 5.4 Administration 5.4.1 Listing resources 5.4.2 Selecting a resource 5.4.3 Creating a resource 5.4.4 Access Key secrets 5.4.5 Activity 5.4.6 Transfer: Using specific transfer ports 5.4.7 Using ATS 5.4.8 Files with type 1ink 5.4.9 Example: Bulk creation of users	. 79 . 80 . 80 . 81 . 82 . 82 . 83 . 83 . 84 . 85 . 85 . 85 . 85 . 85 . 86 . 86 . 87 . 88 . 88 . 88 . 88
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use 5.3 Calling AoC APIs from command line 5.4 Administration 5.4.1 Listing resource 5.4.2 Selecting a resource 5.4.3 Creating a resource 5.4.4 Access Key secrets 5.4.5 Activity 5.4.6 Transfer: Using specific transfer ports 5.4.7 Using ATS 5.4.8 Files with type link 5.4.9 Example: Bulk creation of users 5.4.10 Example: Find with filter and delete	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 83 . 84 . 85 . 85 . 85 . 85 . 85 . 86 . 86 . 87 . 88 . 88 . 88 . 88 . 88
5	4.34 Plugins vs Transfer Agents . Plugin: aoc : IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard . 5.2 AoC configuration: Using manual setup . 5.2.1 AoC manual configuration: Details . 5.2.2 API Client Registration . 5.2.3 Configuration for Aspera on Cloud . 5.2.4 Authentication with private key . 5.2.5 User key registration . 5.2.6 Option Preset modification for JWT . 5.2.7 Public and private links . 5.2.8 AoC: First Use . 5.3 Calling AoC APIs from command line . 5.4.1 Listing resources . 5.4.2 Selecting a resource . 5.4.3 Creating a resource . 5.4.4 Access Key secrets . 5.4.5 Activity . 5.4.6 Transfer: Using specific transfer ports . 5.4.7 Using ATS . 5.4.8 Files with type 1ink . 5.4.9 Example: Bulk creation of users . 5.4.10 Example: Find deactivated users since more than 2 years	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 83 . 84 . 85 . 85 . 85 . 85 . 85 . 85 . 85 . 85
5	4.34 Plugins 4.35 Plugins vs Transfer Agents Plugin: aoc: IBM Aspera on Cloud 5.1 AoC configuration: Using Wizard 5.2 AoC configuration: Using manual setup 5.2.1 AoC manual configuration: Details 5.2.2 API Client Registration 5.2.3 Configuration for Aspera on Cloud 5.2.4 Authentication with private key 5.2.5 User key registration 5.2.6 Option Preset modification for JWT 5.2.7 Public and private links 5.2.8 AoC: First Use 5.3 Calling AoC APIs from command line 5.4 Administration 5.4.1 Listing resource 5.4.2 Selecting a resource 5.4.3 Creating a resource 5.4.4 Access Key secrets 5.4.5 Activity 5.4.6 Transfer: Using specific transfer ports 5.4.7 Using ATS 5.4.8 Files with type link 5.4.9 Example: Bulk creation of users 5.4.10 Example: Find with filter and delete	. 79 . 80 . 81 . 81 . 82 . 82 . 83 . 84 . 85 . 85 . 85 . 85 . 85 . 85 . 86 . 86 . 87 . 87 . 88 . 88 . 88 . 88 . 88

		5.4.14 Example: Display transfer events (ops/transfer)	89
		5.4.15 Example: Display node events (events)	89
		5.4.16 Example: Display members of a workspace	89
		5.4.17 Example: Add all members of a workspace to another workspace	90
		5.4.18 Example: Get users who did not log since a date	90
		5.4.19 Example: List Limited users	90
		5.4.20 Example: Create a group, add to workspace and add user to group	90
		5.4.21 Example: Perform a multi Gbps transfer between two remote shared folders	91
		5.4.22 Example: Create registration key to register a tethered node	91
		5.4.23 Example: Delete all registration keys	91
		5.4.24 Example: Create a Node	92
	5.5		92
		Packages app	92
	0.0	5.6.1 Send a Package	92
		5.6.2 Receive packages	93
		5.6.3 List packages	95
	5.7	· · · · · · · · · · · · · · · · · · ·	95
	5.7	5.7.1 Download Files	96
		5.7.2 Shared folders	96
		5.7.3 Cross Organization transfers	98
		5.7.4 Find Files	99
	E 0	Tested commands for acc	99
	5.6	rested commands for add	77
6	Plus	gin: ats : IBM Aspera Transfer Service	102
•		•	102
			103
			103
		·	103
	0.4	rested communities for the second sec	100
7	Plu	gin: server: IBM Aspera High Speed Transfer Server (SSH)	105
	7.1	Tested commands for server	105
			106
			107
			107
8			108
	8.1	File Operations	
		8.1.1 Browse	
	8.2		109
	8.3		110
	8.4	Central	110
	8.5	Sync	110
	8.6	FASP Stream	110
	8.7		110
	8.8	Out of Transfer File Validation	111
	8.9	Example: SHOD to ATS	111
	8.10	O Node file information	111
	8.12	1 Create access key	112
	8.12	2 Generate and use bearer token	112
		8.12.1 Bearer token: Environment	113
		8.12.2 Bearer token: Preparation	113
			113
			114
	8.13		114
			116
	0.1		
9	Plu	gin: faspex5 : IBM Aspera Faspex v5	
9	Plu 9.1	gin: faspex5: IBM Aspera Faspex v5 Faspex 5 quick start with wizard	117
9	Plug 9.1 9.2	gin: faspex5: IBM Aspera Faspex v5 Faspex 5 quick start with wizard	117 118
9	Plug 9.1 9.2	gin: faspex5: IBM Aspera Faspex v5 Faspex 5 quick start with wizard	117 117 118 119

	9.5 Faspex 5 bootstrap authentication	110
	9.6 Tested commands for faspex5	
	9.7 Faspex 5: Inbox selection	
	9.8 Faspex 5: Send a package	121
	9.9 Faspex 5: Send a package with metadata	122
	9.10 Faspex 5: List packages	
	9.11 Faspex 5: Browsing folder content	
	9.12 Faspex 5: Content of a received Package	
	9.13 Faspex 5: Receive a package	
	9.14 Faspex 5: List all shared inboxes and work groups	124
	9.15 Faspex 5: Create Metadata profile	124
	9.16 Faspex 5: Create a Shared inbox with specific metadata profile	
	9.17 Faspex 5: List content in Shared folder and send package from remote source	
	9.18 Faspex 5: Receive all packages (cargo)	
	7.10 Faspex 5. Receive all packages (caigo)	120
	9.19 Faspex 5: Invitations	125
	9.20 Faspex 5: Cleanup packages	
	9.21 Faspex 5: Admin: Unlock user	125
	9.22 Faspex 5: Faspex 4-style post-processing	126
	9.23 Faspex 5: Faspex 4 Gateway	
	9.24 Faspex 5: Get Bearer token to use API	
	7.24 Laspex 3. det bearer token to use Al I	12/
10	Plugin: faspex : IBM Aspera Faspex v4	128
	10.1 Listing Packages	
	10.1.1 Option box	
	10.1.2 Option recipient	128
	10.1.3 Option query	128
	10.1.4 Example: List packages in dropbox	
	10.2 Receiving a Package	
	10.3 Sending a Package	
	10.4 Email notification on transfer	
	10.5 Operations on dropbox	
	10.6 Remote sources	130
	10.7 Automated package download (cargo)	130
	10.8 Tested commands for faspex	
	200 100000 0011111111111111111111111111	
11	Plugin: shares : IBM Aspera Shares v1	132
	11.1 Tested commands for shares	132
40	Physics and I at TRM Assess Console	424
12	Plugin: console : IBM Aspera Console	134
	12.1 Transfer filter	
	12.2 Tested commands for console	134
13	Plugin: orchestrator :IBM Aspera Orchestrator	135
	13.1 Tested commands for orchestrator	135
14	Plugin: cos : IBM Cloud Object Storage	136
	14.1 Using endpoint, API key and Resource Instance ID (CRN)	136
	14.2 Using service credential file	
	14.3 Operations, transfers	
	14.4 Tested commands for cos	138
1 5	Plugin: httpgw: HTTP Gateway	139
	15.1 Tested commands for httpgw	139
16	Plugin: faspio : Faspio Gateway	140
-0		
	16.1 Tested commands for faspio	140
17	Plugin: alee : Aspera License Entitlement Engine	141
	17.1 Tested commands for alee	141

18		preview																			142
		pera Server	_																		
		ternal tools																			
		.2.1 Image																			
		.2.2 Video:																			
		.2.3 Office: nfiguration																			143 144
		ingulation itions for ge																			
		ecution .																			
		nfiguration																			
		ndidate det																			
		eview File t																			
		pported inp																			
		imemagic																			
		neration: R																			146
	18.1 7 e	sted comm	ands for	previ	iew	 										 				•	146
10	TRM Ac	pera Sync																			148
		arting a syn	nc sessio	n																	
		.1.1 sync																			
		.1.2 sync																			
		nc managei																			
	,,,,				8	 										 				•	
20	Hot fold																				150
		quirements																			
		tup proced																			
		.2.1 ascp																			
		.2.2 Server																			
		.2.3 Sched ample: Upl																			
		ample: Uni																			
		ample: Uni																			151
		•		,																	
21	Health	check and	Nagios																		153
22	SMTP f	or email no	otificatio	ons																	154
	22.1 Ex	ample of co	onfigurat	tion .		 										 	 				154
	22.2 En	nail templat	tes			 										 	 				154
		st																			
	22.4 No	tifications f	for trans	fer stati	us .	 										 					155
22	Tool	session	l																		156
23		mparison o		ices																	156
		nple sessio																			157
		ynchronous																			157
		, ample of la																			157
	23.5 He	lp				 										 	 				157
24	Ruby M	odule: As	pera																		159
25	History																				160
	_																				
26		n problem				 	,														161
		or: "Remot																			161
		or: "can't fi																			161 161
		25519 key uby: net-																			161
		or: "SSL_re																			162
		or: ascp																			162
		or: Cannot																			
				, , ,		 , ,		•	•	-	•	-	-	•	-	 	 -	-	•	-	

gem version 4.24.1 Test passing openssf best practices passing

Chapter 1

Introduction



Figure 1.1: Hootput the Owl

Hootput lives in the terminal, watching over every command with wide, unblinking eyes. Known for concise output and sharp insight, this owl thrives where others get lost in the dark. It doesn't chatter; it hoots—clear, precise, and always on time.

Like ascli, Hootput is built for action: launching transfers, parsing options, and navigating APIs without hesitation. Light on feathers but heavy on wisdom, it turns complexity into simple one-liners. When you hear Hootput's call, you know your data is already in flight.

"Hey, I'm ascli — your data's personal courier. I don't do flashy dashboards; I'm happiest in a terminal window. Hand me a command, and I'll zip your files across the network faster than you thought possible.

Need to automate? I'm script-friendly.

Need to integrate? I've got APIs on speed-dial.

Need to debug? I'll show you what's going on under the hood.

Think of me as Aspera's command-line sidekick: quick, reliable, and a little no-nonsense. You bring the files; I'll bring the horsepower."

Version: 4.25.0.pre Laurent/2016-2025

The aspera-cli Ruby gem offers a powerful command-line interface (CLI, ascli) for IBM Aspera software, facilitating seamless interaction with Aspera APIs and enabling high-performance file transfers. It also serves as an excellent resource for developers seeking to explore and understand the Aspera API ecosystem.

Ruby Gem: https://rubygems.org/gems/aspera-cli

Ruby Doc: https://www.rubydoc.info/gems/aspera-cli

Minimum required Ruby version: >= 3.1.

Warning

The minimum Ruby version will be 3.2 in a future version.

Aspera APIs on IBM developer Link 2

Release notes: see CHANGELOG.md

A PDF version of this documentation is available here: docs/Manual.pdf.

1.1 BUGS, FEATURES, CONTRIBUTION

Refer to BUGS.md and CONTRIBUTING.md.

This documentation does not provide exhaustive details for all commands and options. Most commands correspond directly to REST API calls on Aspera products. For detailed information, consult the official Aspera API documentation. For debugging, use --log-level=debug to view the underlying API calls.

1.2 When to use and when not to use

The ascli tool is designed for command-line interaction with IBM Aspera products, enabling users to execute remote commands and perform file transfers efficiently. It supports both interactive terminal operations (e.g., maintenance tasks on VT100-compatible terminals) and scripting use cases (e.g., batch jobs via shell scripts or cron).

Internally, ascli integrates several components:

- A configuration file (config.yaml) for persistent settings
- Advanced command-line options (see Extended Value Syntax)
- REST API calls, including OAuth (like curl)
- Aspera's ascp for high-speed file transfers

For programmatic integration in languages such as C/C++, Go, Python, NodeJS, and others, it is recommended to use the Aspera APIs directly. These include:

- REST APIs for products like Aspera on Cloud (AoC), Faspex, and Node
- The Transfer Daemon with gRPC interfaces and language-specific stubs (C/C++, Python, .NET/C#, Java, Go, Ruby, Rust, etc.)

Using these APIs is generally more suitable for long-term development and maintenance. Example implementations can be found at: https://github.com/laurent-martin/aspera-api-examples.

For scripting and ad hoc command-line tasks, ascli is ideal. It is developer-friendly and well-suited for quickly testing and learning Aspera APIs (See Logging, Debugging).

Clarifying the CLI landscape: ascp is the low-level command-line utility that implements the FASP protocol and is used for actual data transfers. Every Aspera transfer involves an ascp process on both the client and server sides. While ascp can be used directly, it is limited to basic send/receive operations and lacks features like configuration management, automatic resume, and remote file listing. ascli provides a higher-level interface that encompasses all ascp capabilities and adds significant usability improvements.

1.3 Notations, Shell, Examples

Command line operations examples are shown using a shell such as: bash or zsh.

Command line arguments beginning with my_ in examples, e.g. my_param_value, are user-provided value and not fixed value commands.

ascli is an API Client toward the remote Aspera application Server (Faspex, HSTS, etc...)

Some commands will start an Aspera transfer (e.g. upload). The transfer is not directly implemented in ascli, rather ascli uses one of the external Aspera Transfer Clients called Transfer Agents.

Note

A <u>Transfer Agent</u> is a client for the remote Transfer Server (HSTS/HSTE). It can be local or remote. For example a remote Aspera Transfer Server may be used as a transfer agent (using Node API). i.e. using option --transfer=node

Chapter 2

Quick Start

This section guides you from installation to first use and advanced use.

First, follow section: Installation (Ruby, Gem, FASP) to start using ascli.

Once the gem is installed, ascli shall be accessible:

ascli --version

4.25.0.pre

• Note

All command line examples provided in sections named <u>Tested commands for _plugin_name_</u> are tested during version validation.

2.1 First use

Once installation is completed, you can proceed to the first use with a demo server:

If you want to test with Aspera on Cloud, jump to section: Wizard.

To test with Aspera demo transfer server, set up the environment and then test:

ascli config initdemo

ascli server browse /

zmode	zuid	zgid	size	mtime	name
dr-xr-xr-x	xfer	demousers	4096	2014-11-05 16:01:56 +0100	aspera-test-dir-large
drwxrwxr-x	xfer	demousers	94208	2025-03-31 11:27:33 +0200	Upload
dr-xr-xr-x	xfer	demousers	4096	2014-11-05 16:01:56 +0100	aspera-test-dir-small
dr-xr-xr-x	xfer	demousers	4096	2014-11-05 16:01:56 +0100	aspera-test-dir-tiny

If you want to use ascli with another server, and in order to make further calls more convenient, it is advised to define an Option Preset for the server's authentication options. The following example will:

- Create an Option Preset
- Define it as default for the server plugin
- · List files in a folder
- · Download a file

ascli config preset update myserver --url=ssh://demo.asperasoft.com:33001 --username=asperaweb
---password=my_password_here

Updated: myserver Saving config file.

ascli config preset set default server myserver

Updated: default: server <- myserver
Saving config file.</pre>

ascli server browse /aspera-test-dir-large

zmode	zuid	zgid	size	mtime	name
-rw-rr -rw-rr -rw-rr -rw-rr -rw-rr	xfer xfer xfer xfer xfer xfer	demousers demousers demousers demousers demousers	5368709120 524288000 209715200 1048576000 104857600 10737418240	2014-11-05 16:01:56 +0100 2014-11-05 16:01:56 +0100 2014-11-05 16:01:56 +0100 2014-11-05 16:01:56 +0100 2014-11-05 16:01:56 +0100 2014-11-05 16:01:56 +0100	5GB 500MB 200MB 1GB 100MB 10GB

ascli server download /aspera-test-dir-large/200MB

2.2 Going further

Get familiar with configuration, options, commands: Command Line Interface.

Then, follow the section relative to the product you want to interact with (Aspera on Cloud, Faspex, ...): Application Plugins

Chapter 3

Installation

It is possible to install <u>either</u> directly on the host operating system (Linux, macOS, Windows) or as a <u>container</u> (<u>docker</u> , <u>podman</u> , <u>singularity</u>).

The direct installation is recommended and consists in installing:

- Ruby language
- aspera-cli
- Aspera Transfer Daemon (ascp)

Ruby version: >= 3.1.

A Warning

The minimum Ruby version will be 3.2 in a future version.

The following sections provide information on the various installation methods.

An internet connection is required for the installation. If you don't have internet for the installation, refer to section Installation without internet access.

3.1 Single file executable

Caution

This is a Beta feature.

ascli is available as a single platform-dependent executable. Beta releases can be found here.

Installation:

curl -o ascli https://eudemo.asperademo.com/download/aspera-cli/ascli.4.24.1.osx-arm64 chmod a+x ascli ./ascli conf transferd install

Note

Replace the URL with the one for your platform. Installation of ascp is still required separately. Refer to Install ascp.

3.1.1 Linux: GLIBC version

! Caution

On Linux, the executable requires a minimum GLIBC version, specified in the executable name on download site.

On Linux, check your system's GLIBC version on this site: repology.org, or check your GLIBC version with 1dd:

```
ldd --version | head -n1
```

ldd (GNU libc) 2.34

Check an executable's (e.g. /bin/bash , ascli , ascp) minimum required GLIBC version:

objdump -p /bin/bash | sed -n 's/^.*GLIBC_//p' | sort -V | tail -n1

2.34

1 Note

If objdump is not available, then use strings or grep -z 'GLIBC_'|tr \\0 \\n

The required GLIBC version for ascp can be found in the Release Notes of HSTS or in this page.

3.2 Ruby

A Ruby interpreter is required to run ascli.

Required Ruby version: >= 3.1.

Warning

The minimum Ruby version will be 3.2 in a future version.

Ruby can be installed using any method: rpm, yum, dnf, rvm, rbenv, brew, Windows installer, ...

In priority, refer to the official Ruby documentation:

- Download Ruby
- · Installation Guide

For convenience, you may refer to the following sections for a proposed method for specific operating systems.

Latest version of ascli requires a Ruby version at least under maintenance support. If only an older Ruby version is available due to system constraints, then use an older version of ascli that supports it.

3.2.1 Windows: Installer

Manual installation:

- Navigate to https://rubyinstaller.org/ → Downloads.
- Download the latest Ruby installer "with devkit". (Msys2 is needed to install some native extensions, such as grpc)
- Execute the installer which installs by default in: C:\RubyVV-x64 (VV is the version number)
- At the end of the installation procedure, the Msys2 installer is automatically executed, select option 3 (Msys2 and mingw)
- Then install the aspera-cli gem and Aspera Transfer Daemon (see next sections)

Automated installation, with internet access:

The Ruby installer supports silent installation, to see the options, execute it with /help , or refer to the Ruby Installer FAQ

Download the Ruby installer executable from https://rubyinstaller.org/downloads/ and then install:

rubyinstaller-devkit-3.2.2-1-x64.exe /silent /currentuser /noicons /dir=C:\aspera-cli

3.2.2 macOS: brew

<u>macOS</u> comes with Ruby 2.6. It is an old unsupported version and Apple has deprecated it. It will be removed from macOS in the future. Do not use it.

The recommended way is to use Homebrew.

```
brew install ruby
```

This installs a recent Ruby suitable for ascli.

To add PATH to Ruby on Apple Silicon, add this in your shell configuration file (e.g. \(\bigcup / .bash_profile \) or \(\bigcup / .zshrc \):

```
use_ruby(){
   local version=$1
   case $version in list) for r in $(brew list -1 | grep '^ruby'); do
     echo "$(brew info --json=v1 $r | jq -r '.[0].installed[0].version') : use_ruby $r"
   done|gsort -k1,1V;return;;; esac
   local prefix=$(brew --prefix ruby${version:+@}$version)
   if ! test -d "$prefix";then
       echo "No such ruby version: $version"
       brew list|grep ruby
       return 1
   fi
   PATH="$prefix/bin:$(echo "$PATH" | tr ':' '\n' | grep -v '/ruby' | paste -sd ':' -)"
   PATH="$(gem env gemdir)/bin:$PATH"
   export LDFLAGS="-L$prefix/lib"
   export CPPFLAGS="-I$prefix/include"
   export PKG CONFIG PATH="$prefix/lib/pkgconfig"
   echo "Using: $prefix"
   ruby -v
use_ruby
```

3.2.3 Linux: Package

If your Linux distribution provides a standard Ruby package, you can use it provided that the version supported.

Example: RHEL 8+, Rocky Linux 8+: with extensions to compile native gems

Check available Ruby versions:

```
dnf module list ruby
```

• If Ruby was already installed with an older version, remove it:

```
dnf module -y reset ruby
```

• Install packages needed to build native gems:

```
dnf install -y make automake gcc gcc-c++ kernel-devel
```

Enable the Ruby version you want:

```
dnf module -y enable ruby:3.1
dnf install -y ruby-devel
```

Example: Ubuntu

```
apt-get install -y ruby-full
```

Other examples:

```
yum install -y ruby ruby-devel rubygems ruby-json
```

```
apt install -y ruby ruby-dev rubygems ruby-json
```

One can remove all installed gems, for example to start fresh:

```
gem uninstall -axI $(ls $(gem env gemdir)/gems/|sed -e 's/-[^-]*$//'|sort -u)
```

3.2.4 Unix-like: RVM: Single user installation (not root)

Install rvm . Follow https://rvm.io/.

Execute the shell/curl command. As regular user, it installs in the user's home: ~/.rvm.

\curl -sSL https://get.rvm.io | bash -s stable

Follow on-screen instructions to install keys, and then re-execute the command.

Upon RVM installation, open a new terminal or initialize with:

source ~/.rvm/scripts/rvm

It is advised to get one of the pre-compiled Ruby version, you can list with:

rvm list --remote

Install the chosen pre-compiled Ruby version:

rvm install 3.2.2

Ruby is now installed for the user, go to Gem installation.

Alternatively RVM can be installed system-wide, for this execute as root. It then installs by default in /usr/local/rvm for all users and creates /etc/profile.d/rvm.sh. One can install in another location with:

```
curl -sSL https://get.rvm.io | bash -s -- --path /usr/local
```

As root, make sure this will not collide with other application using Ruby (e.g. Faspex). If so, one can rename the environment script so that it is not loaded by default:

mv /etc/profile.d/rvm.sh /etc/profile.d/rvm.sh.ok

To activate Ruby (and ascli) later, source it:

source /etc/profile.d/rvm.sh.ok

rvm version

On macOS, one way to force use of OpenSSL 3.0 is:

RUBY_CONFIGURE_OPTS="--with-openssl-dir=\$(brew --prefix openssl@3.0)" rvm install 3.4.0

3.2.5 Unix-like: rbeny

If you don't have root access, you can install Ruby in your home directory using rbeny see rbeny-installer:

curl -fsSL https://github.com/rbenv/rbenv-installer/raw/HEAD/bin/rbenv-installer | bash

Then open a new terminal, or source the shell initialization script:

source ~/.bashrc

Then install Ruby:

rbenv install 3.2.2

3.2.6 Other Unixes (AIX)

Ruby is sometimes made available as an installable package through third party providers. For example for AIX, one can look at:

https://www.ibm.com/support/pages/aix-toolbox-open-source-software-downloads-alpha#R

If your Unix does not provide a pre-built Ruby, you can get it using one of those methods.

For instance to build from source and install in /opt/ruby:

```
wget https://cache.ruby-lang.org/pub/ruby/2.7/ruby-2.7.2.tar.gz
gzip -d ruby-2.7.2.tar.gz
tar xvf ruby-2.7.2.tar
cd ruby-2.7.2
```

```
./configure --prefix=/opt/ruby
make ruby.imp
make
make
```

3.2.7 **JRuby**

ascli can also run with the JRuby interpreter. All what is needed is a JVM (Java Virtual Machine) on your system (java). The JRuby package comes pre-complied and does not require compilation of native extensions. Use a version of JRuby compatible with Ruby version supported by ascli. Refer to the Wikipedia page to match JRuby and Ruby versions. Choose the latest version from:

https://www.jruby.org/download

Caution

The startup time is slightly longer using jruby than the native Ruby. Refer to the JRuby wiki for details. This can be reduced by using the --dev option. The transfer speed is not impacted (executed by ascp binary).

Note

JRuby can be installed using rvm.

Example: start ascli with JRuby and reduce startup time:

```
export JRUBY_OPTS=--dev
ascli -v

or

JRUBY_OPTS=--dev ascli -v
```

3.2.8 Optional gems

Some additional gems are required for some specific features. Those are not installed as part of dependencies because they involve compilation of native code but concern less-used features.

See Gemfile:

name	version	comment
grpc	~> 1.71	(no jruby) for Aspera Transfer Daemon
mimemagic	~> 0.4	for preview
rmagick	~> 6.1	(no jruby) for terminal view
symmetric-encryption	~> 4.6	for encrypted hash file secrets
bigdecimal	~> 3.1	if RUBY_VERSION >= '3.4' for symmetric-encryption?
sqlite3	~> 2.7	(no jruby) for async DB
jdbc-sqlite3	~> 3.46	(jruby) for async DB
sequel	~> 5.96	(jruby) for async DB

Install like this:

```
gem install grpc -v '~> 1.71'
gem install mimemagic -v '~> 0.4'
gem install rmagick -v '~> 6.1'
gem install symmetric-encryption -v '~> 4.6'
gem install bigdecimal -v '~> 3.1'
gem install sqlite3 -v '~> 2.7'
gem install jdbc-sqlite3 -v '~> 3.46'
gem install sequel -v '~> 5.96'
```

3.3 Ruby Gem: aspera-cli

Once you have Ruby and rights to install gems, install the aspera-cli gem and its dependencies:

```
gem install aspera-cli --pre
```

To upgrade to the latest version:

```
gem update aspera-cli
```

During its execution, ascli checks every week if a new version is available and notifies the user in a WARN log. To deactivate this feature, globally set the option version_check_days to 0, or specify a different period in days.

To check if a new version is available (independently of version check days):

ascli config check_update

3.3.1 Gem installation with signature verification

The gem is signed with a private key, and the public certificate is available in the GitHub repository (certs/aspera-cli-public). When installing the gem, the signature can be optionally verified.

For secure installation, one can install the gem with the public key:

Import the verification certificate:

```
gem cert --add <(curl -Ls

→ https://raw.githubusercontent.com/IBM/aspera-cli/main/certs/aspera-cli-public-cert.pem)
```

The user installs the gem with HighSecurity or MediumSecurity: this will succeed only of the gem is trusted:

gem install -P MediumSecurity aspera-cli

3.3.2 Beta release of gem

Beta version of gem can be found here: https://ibm.biz/aspera-cli-beta

On Linux/macOS, install in a terminal:

```
curl -sLo aspera-cli-beta.gem https://ibm.biz/aspera-cli-beta
gem install aspera-cli-beta.gem
```

On Windows, download the link, that saves the file: aspera-cli-beta.gem, then install with gem install aspera-cli-beta.gem

3.4 FASP Protocol: ascp

File transfers are primarily performed using the <u>FASP</u> protocol via the <u>ascp</u> utility.

To execute an Aspera transfer, only two additional files are required, both included with the Aspera Transfer Daemon:

- ascp the transfer executable
- aspera-license the license file (located in the same directory as ascp or in ../etc)

These components can be installed either by installing the Aspera transfer software or by executing the asclicommand.

3.4.1 Installation of ascp through transferd

The easiest option to install ascp is through the use of the IBM Aspera Transfer Daemon (transferd). Install using ascli for the current platform with:

```
ascli config transferd install
```

or

```
ascli config ascp install
```

The installation of the transfer binaries follows those steps:

- Select the SDK package to use. Check the sdk_url option:
 - If the value is not the default value (DEF), it directly specifies the archive URL to download.
 - If the value is DEF, ascli downloads the YAML file from the URL specified by the locations_url option (default: https://ibm.biz/sdk_location).
 - * This YAML file lists supported architectures (OS, CPU) and Aspera Transfer Daemon versions with their associated package URLs.
 - * If an additional positional parameter is provided, it specifies the SDK version to use; otherwise the latest version is selected.
 - * The package URL matching the current system architecture is then used.
- · Extract the archive
 - By default, the archive is extracted to \$HOME/.aspera/sdk.
 - The destination folder can be changed by setting the sdk_folder option.

Option	Default	Description
sdk_url	DEF	URL to download the Aspera Transfer Daemon archive. DEF means: select from available archives.
locations_url	https://ibr	URL to get download URLs of Aspera Transfer Daemon from IBM official repository.
sdk_folder	\$HOME/.aspe	Folder where the SDK archive is extracted.

Available Transfer Daemon versions available from locations_url can be listed with: ascli config transferd list

To install a specific version, e.g. 1.1.3:

```
ascli config ascp install 1.1.3
```

To get the download URL for a specific platform and version:

```
ascli config transferd list --select=@json:'{"platform":"osx-arm64","version":"1.1.3"}' --fields=url
```

To download it, pipe to config download:

If installation from a local file preferred (air-gapped installation) instead of fetching from internet: one can specify the location of the SDK file with option sdk url:

```
ascli config ascp install --sdk-url=file:///macos-arm64-1.1.3-c6c7a2a.zip
```

The format is: file:///<path> , where <path> can be either a relative path (not starting with /), or an absolute path.

Supported platforms are listed in the Release Notes and archives can be downloaded from Downloads.

3.4.2 Installation of ascp through other component

If the embedded method is not used, the following packages are also suitable:

- IBM Aspera Connect Client (Free)
- IBM Aspera Desktop Client (Free)
- IBM Aspera High Speed Transfer Server (Licensed)
- IBM Aspera High Speed Transfer Endpoint (Licensed)

For instance, Aspera Connect Client can be installed by visiting the page: https://www.ibm.com/aspera/connect/.

ascli will detect most of Aspera transfer products in standard locations and use the first one found by default. Refer to section FASP for details on how to select a client or set path to the FASP protocol.

Several methods are provided to start a transfer. Use of a local client (direct transfer agent) is one of them, but other methods are available. Refer to section: Transfer Agents

3.5 Installation in air gapped environment

• Note

No pre-packaged version is provided yet.

3.5.1 Gem files and dependencies

The sample script: windows/build_package.sh can be used to download all necessary gems and dependencies in a tar.gz.

```
./build_package.sh aspera-cli 4.18.0
Archive: aspera-cli-4.18.0-gems.tgz
```

3.5.2 Unix-like

A method to build one is provided here:

The procedure:

- Follow the non-root installation procedure with RVM, including gem
- Archive (zip, tar) the main RVM folder (includes ascli):

```
cd $HOME && tar zcvf rvm-ascli.tgz .rvm
```

- Download the Transfer Daemon archive for the selected architecture, follow Install ascp
- Transfer those 2 files to the target system
- On target system

```
cd $HOME

tar zxvf rvm-ascli.tgz

source ~/.rvm/scripts/rvm

ascli config ascp install --sdk-url=file:///[SDK archive file path]
```

Add those lines to shell environment (.profile)

source ~/.rvm/scripts/rvm

3.5.3 Windows

Installation without network:

It is essentially the same procedure as installation for Windows with internet, but instead of retrieving files from internet, copy the files from a machine with internet access, and then install from those archives:

Download the Ruby installer from https://rubyinstaller.org/downloads/

```
v=$(curl -s https://rubyinstaller.org/downloads/ | sed -nEe

→ 's|.*(https://.*/releases/download/.*exe).*|\1|p' | head -n 1)

curl -o ${v##*/} $v
```

- Create an archive with necessary gems like in previous section
- Download the Transfer Daemon following: Install ascp
- Create a Zip with all those files and transfer to the target system.

Then, on the target system:

- Unzip the archive
- Execute the installer:

rubyinstaller-devkit-3.2.2-1-x64.exe /silent /currentuser /noicons /dir=C:\aspera-cli

• Install the gems: Extract the gem archive and then:

```
gem install --force --local *.gem
```

• Install the Aspera Transfer Daemon SDK

ascli config ascp install --sdk-url=file:///sdk.zip

• Note

An example of installation script is provided: windows/install.bat

3.6 Container

The container image is: docker.io/martinlaurent/ascli. The container contains: Ruby, ascli and the Aspera Transfer Daemon. To use the container, ensure that you have podman (or docker) installed.

podman --version

3.6.1 Container: Quick start

Wanna start quickly? With an interactive shell?

Execute this:

podman run --rm --tty --interactive --entrypoint bash docker.io/martinlaurent/ascli:latest

Note

This command changes the entry point to an interactive shell instead of direct execution of ascli.

Then, execute individual ascli commands such as:

```
ascli config init
ascli config preset overview
ascli config ascp info
ascli server ls /
```

That is simple, but there are limitations:

- Everything happens in the container
- Any generated file in the container will be lost on container (shell) exit. Including configuration files and downloaded files.
- No possibility to upload files located on the host system

3.6.2 Container: Details

The container image is built from this Dockerfile. The entry point is ascli and the default command is help.

The container can be executed for individual commands like this: (add ascli commands and options at the end of the command line, e.g. -v to display the version)

```
podman run --rm --tty --interactive docker.io/martinlaurent/ascli:latest
```

For more convenience, you may define a shell alias:

```
alias ascli='podman run --rm --tty --interactive docker.io/martinlaurent/ascli:latest'
```

Then, you can execute the container like a local command:

```
ascli -v
```

```
4.25.0.pre
```

In order to keep persistency of configuration on the host, you should specify your user's configuration folder as a volume for the container. To enable write access, a possibility is to run as root in the container (and set the default configuration folder to /home/cliuser/.aspera/ascli). Add options:

--user root --env ASCLI_HOME=/home/cliuser/.aspera/ascli --volume
 \$HOME/.aspera/ascli:/home/cliuser/.aspera/ascli

Note

If you are using a podman machine, e.g. on macOS, make sure that the folder is also shared between the VM and the host, so that sharing is:

container → VM → Host: podman machine init ... --volume="/Users:/Users"

As shown in the quick start, if you prefer to keep a running container with a shell and ascli available, you can change the entry point, add option:

--entrypoint bash

You may also probably want that files downloaded in the container are directed to the host. In this case you need also to specify the shared transfer folder as a volume:

--volume \$HOME/xferdir:/xferfiles

Caution

ascli is run inside the container, so transfers are also executed inside the container and do not have access to host storage by default.

And if you want all the above, simply use all the options:

```
alias asclish="podman run --rm --tty --interactive --user root --env

ASCLI_HOME=/home/cliuser/.aspera/ascli --volume $HOME/.aspera/ascli:/home/cliuser/.aspera/ascli

--volume $HOME/xferdir:/xferfiles --entrypoint bash docker.io/martinlaurent/ascli:latest"
```

export xferdir=\$HOME/xferdir mkdir -p \$xferdir chmod -R 777 \$xferdir mkdir -p \$HOME/.aspera/ascli asclish

3.6.3 Container: Sample start script

A convenience sample script is also provided: download the script dascli from the GIT repo:

• Note

If you have installed ascli, the script dascli can also be found like this: cp \$(ascli config gem path)/../container/dascli ascli

Some environment variables can be set for this script to adapt its behavior:

env var	Description	Default	Example
ASCLI_HOME	Configuration folder (persistency)	<pre>\$HOME/.aspera/ascli</pre>	\$HOME/.ascli_config
docker_args	Additional options to podman	<empty></empty>	volume /Users:/Users
image	Container image name	docker.io/martinlauren	n/a
version	Container image version	Latest	4.8.0.pre

The wrapping script maps the folder \$ASCLI_HOME on host to /home/cliuser/.aspera/ascli in the container. (value expected in the container). This allows having persistent configuration on the host.

To add local storage as a volume, you can use the env var docker_args:

Example of use:

```
curl -o ascli https://raw.githubusercontent.com/IBM/aspera-cli/main/container/dascli
chmod a+x ascli
export xferdir=$HOME/xferdir
mkdir -p $xferdir
```

```
chmod -R 777 $xferdir
export docker_args="--volume $xferdir:/xferfiles"

./ascli config init
echo 'Local file to transfer' > $xferdir/samplefile.txt
./ascli server upload /xferfiles/samplefile.txt --to-folder=/Upload
```

• Note

The local file (samplefile.txt) is specified relative to storage view from container (/xferfiles) mapped to the host folder \$HOME/xferdir

Caution

Do not use too many volumes, as the legacy aufs limits the number. (anyway, prefer to use overlay2)

3.6.4 Container: Offline installation

• First create the image archive:

```
podman pull docker.io/martinlaurent/ascli
podman save docker.io/martinlaurent/ascli|gzip>ascli_image_latest.tar.gz
```

• Then, on air-gapped system:

podman load -i ascli_image_latest.tar.gz

3.6.5 Container: aspera.conf

ascp's configuration file aspera.conf is located in the container at: /ibm_aspera/aspera.conf (see Dockerfile). As the container is immutable, it is not recommended modifying this file. If one wants to change the content, it is possible to tell ascp to use another file using ascp option -f, e.g. by locating it on the host folder \$HOME/.aspera/ascli mapped to the container folder /home/cliuser/.aspera/ascli:

```
echo '<CONF/>' > $HOME/.aspera/ascli/aspera.conf
```

Then, tell ascp to use that other configuration file:

--transfer-info=@json:'{"ascp_args":["-f","/home/cliuser/.aspera/ascli/aspera.conf"]}'

3.6.6 Container: Singularity

Singularity is another type of use of container.

On Linux install:

dnf install singularity-ce

Build an image like this:

singularity build ascli.sif docker://docker.io/martinlaurent/ascli

Then, start ascli like this:

singularity run ascli.sif

Or get a shell with access to ascli like this:

singularity shell ascli.sif

3.7 SSL library

ascli uses the Ruby openssl gem which uses by default the system's openssl library and its CA certificate bundle.

To display the version of OpenSSL used in ascli:

```
ascli config echo @ruby:OpenSSL::OPENSSL_VERSION --format=text
```

It is possible to specify to use another SSL library or version by executing:

```
gem install openssl -- --with-openssl-dir=[openssl library folder]
```

Where [openssl library folder] is the path to the folder containing the lib and include folders of the openssl library.

For example, on macOS, to use the openss1@3 library installed with brew:

```
openssl version -e|sed -n 's|ENGINESDIR: "\(.*\)/lib[^/]*/.*|\1|p'
/opt/homebrew/Cellar/openssl@3/3.3.0
```

Then install the openss1 gem with:

3.8 SSL CA certificate bundle

SSL certificates are validated using a certificate store, by default it is the one of the system's openss1 library.

To display trusted certificate store locations:

```
ascli --show-config --fields=cert_stores
```

Certificates are checked against the Ruby default certificate store OpenSSL::X509::DEFAULT_CERT_FILE and OpenSSL::X509::DEFAULT_CERT_DIR, which are typically the ones of openssl on Unix-like systems (Linux, macOS, etc...). Ruby's default values can be overridden using env vars: SSL_CERT_FILE and SSL_CERT_DIR.

One can display those default values:

```
ascli config echo @ruby:OpenSSL::X509::DEFAULT_CERT_DIR --format=text
ascli config echo @ruby:OpenSSL::X509::DEFAULT_CERT_FILE --format=text
```

In order to get certificate validation, the CA certificate bundle must be up-to-date. Check this repository on how to update the system's CA certificate bundle: https://github.com/millermatt/osca.

For example on RHEL/Rocky Linux:

```
dnf install -y ca-certificates
update-ca-trust extract
```

The SSL CA certificate bundle can be specified using the cert_stores option, which accepts a list of files or directories. By default, Ruby's system certificate store is used.

When cert_stores is provided:

- It overrides the default locations, which can still be included explicitly using the special value DEF .
- The option accepts either a String (single path) or an Array (multiple paths).
- Each use of the option appends to the list of search paths incrementally.
- If a directory is specified, all files within that directory are automatically included.

• Note

JRuby uses its own implementation and CA bundles.

For example, on Linux to force the use the system's certificate store:

```
--cert-stores=$(openssl version -d|cut -f2 -d'"')/cert.pem
```

ascp also needs to validate certificates when using WSS for transfer TCP part (instead of SSH).

By default, ascp uses a hard coded root location OPENSSLDIR. Original ascp 's hard coded locations can be found using:

```
ascli config ascp info --fields=openssldir
```

E.g. on macOS: /Library/Aspera/ssl . Then trusted certificates are taken from [OPENSSLDIR]/cert.pem and files in [OPENSSLDIR]/certs . ascli overrides the default hard coded location used by ascp for WSS and uses the same locations as specified in cert_stores (using the -i option of ascp).

To update trusted root certificates for <code>ascli</code>: Display the trusted certificate store locations used by <code>ascli</code>. Typically done by updating the system's root certificate store.

An up-to-date version of the certificate bundle can also be retrieved with:

ascli config echo @uri:https://curl.haxx.se/ca/cacert.pem --format=text

To download that certificate store:

ascli config echo @uri:https://curl.haxx.se/ca/cacert.pem --format=text --output=/tmp/cacert.pem

Then, use this store by setting the option cert_stores (or env var SSL_CERT_FILE).

To trust a specific certificate (e.g. self-signed), provided that the CN is correct, save the certificate chain to a file:

ascli config remote_certificate chain https://localhost:9092 --insecure=yes --output=myserver.pem

Note

Use command name to display the remote common name of the remote certificate.

Then, use this file as certificate store (e.g. here, Node API):

ascli config echo @uri:https://localhost:9092/ping --cert-stores=myserver.pem

Chapter 4

Command Line Interface

The command line tool is: ascli

The aspera-cli gem provides a command line interface (CLI) which interacts with Aspera Products (mostly using REST APIs):

- IBM Aspera High Speed Transfer Server (FASP and Node)
- IBM Aspera on Cloud (including ATS)
- IBM Aspera Faspex
- IBM Aspera Shares
- IBM Aspera Console
- IBM Aspera Orchestrator
- And more...

ascli provides the following features:

- Commands to Aspera server products (on-premise and SaaS)
- Any command line <u>options</u> (products URL, credentials or any option) can be provided on command line, in configuration file, in env var, in files, ...
- Commands, Options, and Option values shortcuts
- FASP Transfer Agents can be: local ascp, or Connect Client, or any transfer node
- Transfer parameters can be altered by modification of transfer-spec, this includes requiring multi-session
- Allows transfers from products to products, essentially at node level (using the node transfer agent)
- faspstream creation (using Node API)
- Watchfolder creation (using Node API)
- Additional command plugins can be written by the user
- Download of Faspex and Aspera on Cloud "external" links
- Legacy SSH based FASP transfers and remote commands (ascmd)

Basic usage is displayed by executing:

ascli -h

Refer to sections: Usage.

• Note

ascli features are not fully documented here, the user may explore commands on the command line.

4.1 ascp command line

If you want to use ascp directly as a command line, refer to IBM Aspera documentation of either Desktop Client, Endpoint or Transfer Server where a section on ascp can be found.

Using ascli with plugin server for command line gives advantages over ascp:

- · Automatic resume on error
- · Configuration file

- Choice of transfer agents
- Integrated support of multi-session

All ascp options are supported either through transfer spec parameters (listed with conf ascp spec) and with the possibility to provide ascp arguments directly when the direct agent is used (ascp_args in transfer_info).

4.2 Command line parsing, Special Characters

ascli is typically executed in a shell, either interactively or in a script. ascli receives its arguments on the command line. The way arguments are parsed and provided to ascli depend on the Operating System and shell.

4.2.1 Shell parsing for Unix-like systems: Linux, macOS, AIX

Linux command line parsing is well-defined: It is fully documented in the shell's documentation.

On Unix-like environments, this is typically a POSIX-like shell (bash , zsh , ksh , sh). A c-shell (csh , tcsh) or other shell can also be used. In this environment the shell parses the command line, possibly replacing variables, etc... See bash shell operation. The shell builds the list of arguments and then fork / exec Ruby with that list. Ruby receives a list command line arguments from shell and gives it to ascli. Special character handling (quotes, spaces, env vars, ...) is handled by the shell for any command executed.

4.2.2 Shell parsing for Windows

On Windows, command line parsing first depends on the shell used (see next sections). MS Windows command line parsing is not like Unix-like systems simply because Windows does not provide a list of arguments to the executable (Ruby): it provides the whole command line as a single string, but the shell may interpret some special characters.

So command line parsing is not handled by the shell (cmd.exe), not handled by the operating system, but it is handled by the executable (Ruby). Typically, Windows executables use the Microsoft library for this parsing.

As far as ascli is concerned: the executable is Ruby. It has its own parsing algorithm, close to a Linux shell parsing.

Thankfully, ascli provides a command to check the value of an argument after parsing: config echo. One can also run ascli with option --log-level=debug to display the command line after parsing.

It is also possible to display arguments received by Ruby using this command:

```
C:> ruby -e 'puts ARGV' "Hello World" 1 2
Hello World
1
2
```

• Note

Use pp instead of puts to display as Ruby Array.

Once the shell has dealt with the command line "special" characters for it, the shell calls Windows' CreateProcess with just the whole command line as a single string. (Unlike Unix-like systems where the command line is split into arguments by the shell.)

It's up to the program to split arguments:

- Windows: How Command Line Parameters Are Parsed
- Understand Quoting and Escaping of Windows Command Line Arguments

is a Ruby program, so Ruby parses the command line (receibed with GetCommandLineW) into arguments and provides them to the Ruby code (\$0 and ARGV). Ruby vaguely follows the Microsoft C/C++ parameter parsing rules. (See w32_cmdvector in Ruby source win32.c):

- Space characters: split arguments (space, tab, newline)
- Backslash: \ escape single special character
- Globing characters: *?[]{} for file globing
- Double guotes:
- Single quotes:

4.2.3 Shell parsing for Windows: cmd.exe

The following examples give the same result on Windows using cmd.exe:

Single quote protects the double quote

```
ascli config echo @json:'{"url":"https://..."}'
```

• Triple double quotes are replaced with a single double quote

```
ascli config echo @json:{"""url""":""https://..."""}
```

• Double quote is escaped with backslash within double quotes

```
ascli config echo @json:"{\"url\":\"https://...\"}"
```

cmd.exe handles some special characters: ^"<>|\implies \text{\alpha} \text{. Basically it handles I/O redirection (<>|), shell variables (\mathbb{\alpha}), multiple commands (\mathbb{\alpha}) and handles those special characters from the command line. Eventually, all those special characters are removed from the command line unless escaped with \(\text{\alpha} \) or \(\text{\alpha} \) . \(\text{\alpha} \) are kept and given to the program.

4.2.4 Shell parsing for Windows: Powershell

For Powershell, it actually depends on the version of it (7.3+).

A difficulty is that Powershell parses the command line for its own use and manages special characters, but then it passes the command line to the program (Ruby) as a single string, possibly without the special characters. Since we usually do not use powershell features, it is advised to use the "stop-parsing" token --%.

Details can be found here:

- · Passing arguments with quotes
- quoting rules

The following examples give the same result on Windows using Powershell:

```
PS C:\> echo $psversiontable.psversion

Major Minor Build Revision

5 1 19041 4046

PS C:\> ascli config echo --% @json:'{"k":"v","x":"y"}'

PS C:\> ascli config echo @json:'{"""k""":""v"","""x""":""y""}'
```

Note

The special Powershell argument --% places Powershell in "stop-parsing" mode.

4.2.5 Extended Values (JSON, Ruby, ...)

Some values provided to ascli (options, Command Parameters) are expected to be Extended Values, i.e. not a simple String, but a composite structure (Hash, Array). Typically, the @json: modifier is used, it expects a JSON string. JSON itself has some special syntax: for example " is used to enclose a String.

4.2.6 Testing Extended Values

In case of doubt of argument values after parsing, one can test using command config echo . config echo takes exactly <u>one</u> argument which can use the <u>Extended Value</u> syntax. Unprocessed command line arguments are shown in the error message.

Example: The shell parses three arguments (as String: 1, 2 and 3), so the additional two arguments are not processed by the echo command.

```
ascli config echo 1 2 3
"1"
<u>ERROR: Argument:</u> unprocessed <u>values:</u> ["2", "3"]
```

config echo displays the value of the <u>first</u> argument using Ruby syntax: it surrounds a string with <u>under and add to before special characters.</u>

Note

It gets its value after shell command line parsing and ascli extended value parsing.

In the following examples (using a POSIX shell, such as bash), several equivalent commands are provided. For all example, most special character handling is not specific to ascli: It depends on the underlying syntax: shell, JSON, etc... Depending on the case, a different format option is used to display the actual value.

For example, in the simple string Hello World, the space character is special for the shell, so it must be escaped so that a single value is represented.

Double quotes are processed by the shell to create a single string argument. For <u>POSIX shells</u>, single quotes can also be used in this case, or protect the special character _ (space) with a backslash.

```
ascli config echo "Hello World" --format=text
ascli config echo 'Hello World' --format=text
ascli config echo Hello\ World --format=text
Hello World
```

4.2.7 Using a shell variable, parsed by shell, in an extended value

To be evaluated by shell, the shell variable must not be in single quotes. Even if the variable contains spaces it results only in one argument for ascli because word parsing is made before variable expansion by shell.

• Note

We use a simple shell variable in this example. Note that it does not need to be exported as an environment variable.

```
MYVAR="Hello World"
ascli config echo @json:'{"title":"'$MYVAR'"}' --format=json
ascli config echo @json:{\"title\":\"$MYVAR\"} --format=json
{"title":"Hello World"}
```

4.2.8 Double quote in strings in command line

Double quote is a shell special character. Like any shell special character, it can be protected either by preceding with a backslash or by enclosing in a single quote.

```
ascli config echo \"
ascli config echo '"'
"
```

Double quote in JSON is a little tricky because "is special both for the shell and JSON. Both shell and JSON syntax allow protecting in, but only the shell allows protection using single quote.

```
ascli config echo @json:'"\""' --format=text
ascli config echo @json:\"\\\"\" --format=text
ascli config echo @ruby:\'\"\' --format=text
"
```

Here a single quote or a backslash protects the double quote to avoid shell processing, and then an additional \setminus is added to protect the \square for JSON. But as \setminus is also shell special, then it is protected by another \setminus .

4.2.9 Shell and JSON or Ruby special characters in extended value

Construction of values with special characters is done like this:

- First select a syntax to represent the extended value, e.g. JSON or Ruby
- Write the expression using this syntax, for example, using JSON:

```
{"title":"Test \" ' & \\"}
```

or using Ruby:

```
{"title"=>"Test \" ' & \\"}
{'title'=>%q{Test " ' & \\}}
```

Both \square and \square are special characters for JSON and Ruby and can be protected with \square (unless Ruby's extended single quote notation \square is used).

• Then, since the value will be evaluated by shell, any shell special characters must be protected, either using preceding \ for each character to protect, or by enclosing in single quote:

```
ascli config echo @json:{\"title\":\"Test\ \\\"\ \'\ \&\ \\\\"} --format=json
ascli config echo @json:'{"title":"Test \" '\'' & \\"}' --format=json
ascli config echo @ruby:"{'title'=>%q{Test \" ' & \\\\}}" --format=json
{"title":"Test \" ' & \\"}
```

4.2.10 Reading special characters interactively

If ascli is used interactively (a user typing on terminal), it is easy to require the user to type values:

```
ascli config echo @ruby:"{'title'=>gets.chomp}" --format=json
```

gets is Ruby's method of terminal input (terminated by \n), and \n chomp removes the trailing \n .

4.2.11 Command line arguments from a file

If you need to provide a list of command line argument from lines that are in a file, on Linux you can use the xargs command:

```
xargs -a lines.txt -d \\n ascli config echo
```

This is equivalent to execution of:

```
ascli config echo [line1] [line2] [line3] ...
```

If there are spaces in the lines, those are not taken as separator, as we provide option -d \\n to xargs.

4.2.12 Extended value using special characters read from environmental variables or files

Using a text editor or shell: create a file title.txt (and env var) that contains exactly the text required: Test " ' & \ :

```
export MYTITLE='Test " '\'' & \'
echo -n $MYTITLE > title.txt
```

Using those values will not require any escaping of characters since values do not go through shell or JSON parsing.

If the value is to be assigned directly to an option of ascli, then you can directly use the content of the file or env var using the <code>@file:</code> or <code>@env:</code> readers:

```
ascli config echo @file:title.txt --format=text
ascli config echo @env:MYTITLE --format=text
Test " ' & \
```

If the value to be used is in a more complex structure, then the <code>@ruby:</code> modifier can be used: it allows any Ruby code in expression, including reading from file or env var. In those cases, there is no character to protect because values are not parsed by the shell, or JSON or even Ruby.

```
ascli config echo @ruby:"{'title'=>File.read('title.txt')}" --format=json
ascli config echo @ruby:"{'title'=>ENV['MYTITLE']}" --format=json
{"title":"Test \" ' & \\"}
```

4.3 Positional Arguments and Options

Command line arguments are the units of command line typically separated by spaces (the argv of C). The tokenization of the command line is typically done by the shell, refer to the previous section Command Line Parsing.

ascli handles two types of command line arguments:

- Positional Arguments : position is significant
- Options : only order is significant, but not absolute position

For example:

ascli command subcommand --option-name=VAL1 VAL2

- Executes Command and its Command Parameters (Positional Arguments): command subcommand VAL2
- With one Option: option_name and its value: VAL1

If the value of a command, option or argument is constrained by a fixed list of values, then it is possible to use a few of the first letters of the value, provided that it uniquely identifies the value. For example ascli config pre ov is the same as ascli config preset overview.

The value of Options and Positional Arguments is evaluated with the Extended Value Syntax.

4.3.1 Positional Arguments

Positional Arguments are either:

- · Commands, typically at the beginning
- · Command Parameters, mandatory arguments, e.g. creation data or entity identifier

When options are removed from the command line, the remaining arguments are typically <u>Positional Arguments</u> with a pre-defined order.

<u>Commands</u> are typically entity types (e.g. <u>users</u>) or verbs (e.g. <u>create</u>) to act on those entities. Its value is a <u>String</u> that must belong to a fixed list of values in a given context.

Example:

ascli config ascp info

- ascli is the executable executed by the shell
- conf is the first level command: name of the plugin to be used
- ascp is the second level command: name of the component (singleton)
- info is the third level command: action to be performed

Typically, <u>Commands</u> are located at the <u>beginning</u> of the command line. Order is significant. The provided command must match one of the supported commands in the given context. If wrong, or no command is provided when expected, an error message is displayed and the list of supported commands is displayed.

Standard <u>Commands</u> are: <u>create</u>, <u>show</u>, <u>list</u>, <u>modify</u>, <u>delete</u>. Some entities also support additional commands. When those additional commands are related to an entity also reachable in another context, then those commands are located below command <u>do</u>. For example sub-commands appear after entity selection (identifier), e.g. ascli acc admin node do 1234 browse /: browse is a sub-command of node.

Command Parameters are typically mandatory values for a command, such as entity creation data or entity identifier.

• Note

It could also have been designed as an option. But since it is mandatory and typically these data do not need to be set in a configuration file, it is better designed as a Command Parameter, rather than as an additional specific option. The advantages of using a <u>Command Parameter</u> instead of an option for the same are that the command line is shorter (no option name, just the position), the value is clearly mandatory and position clearly indicates its role. The disadvantage is that it is not possible to define a default value in a configuration file or environment variable using an option value. Nevertheless, <u>Extended Values</u> syntax is supported, so it is possible to retrieve a value from the configuration file (using <u>Qpreset:</u>) or environment variable (using <u>Qenv:</u>).

If a <u>Command Parameter</u> begins with -, then either use the <u>@val:</u> syntax (see <u>Extended Values</u>), or use the <u>--</u> separator (see below).

A few Command Parameters are optional, they are always located at the end of the command line.

4.3.2 Options

Command-line options, such as --log-level=debug, follow these conventions:

- Prefix: All options begin with ___.
- <u>Naming</u>: Option names on command line use lowercase letters and hyphens () as word separators. Option name in config file use underscores () as word separators. Example: | --log-level=debug | is | log_level | in config file.
- Values: An option's value is assigned using = (e.g., --log-level=debug).
- Prefix Usage: Options can be abbreviated by a unique prefix, though this is not recommended. Example: --log-ledebug is equivalent to --log-level=debug.
- Optionality: Most options are optional; they either have a default value or do not require one.
- Order: Options can appear in any position on the command line, but their order may affect processing.

Exceptions and Special Cases:

- Short Forms: Some options have short forms. For example, -Ptoto is equivalent to --preset=toto. Refer to the manual or -h for details.
- Flags: Certain options are flags and do not require a value (e.g., -N).
- Option Terminator: The special option —— ends option parsing. All subsequent arguments, including those starting with —, are treated as positional arguments.
- <u>Dot Notation for Hashes</u>: If an option name contains a dot (...), it is interpreted as a <u>Hash</u>. Each segment separated by a dot represents a key in a nested structure. <u>ascli</u> tries to convert the value to the simplest type (bool, int, float, string). If a specific type is required, it can be specified using the <u>@json</u>: or <u>@ruby</u>: syntax. For example, --a.b.c=1 is equivalent to --a=@json'{"b":{"c":1}}'. This allows specifying nested keys directly on the command line using a concise <u>dot-separated</u> syntax.

Example:

ascli config echo -- --sample

"--sample"

• Note

Here, --sample is taken as an argument, and not as an option, due to --.

Options may have a (hard coded) default value.

Options can be placed anywhere on command line and are evaluated in order. Usually the last value evaluated overrides previous values, but some options are cumulative, e.g. ts.

Options are typically optional: to change the default behavior. But some are mandatory, so they can be placed in a configuration file, for example: connection information.

The value for <u>any</u> options can come from the following locations (in this order, last value evaluated overrides previous value):

- · Configuration file
- Environment variable
- · Command line

Environment variable starting with prefix: ASCLI_ are taken as option values, e.g. ASCLI_OPTION_NAME is for --option-name .

Option show_config dry runs the configuration, and then returns currently set values for options.

ascli --show-config outputs global options only, and ascli [plugin] --show-config outputs global and plugin default options. In addition, option --show-config can be added at the end of any full command line, this displays the options that would be used for the command.

A command line argument is typically designed as option if:

- It is optional, or
- It is a mandatory parameter with a default value that would benefit from being set persistently (i.e. in a configuration file or environment variable, e.g. URL and credentials).

4.4 Interactive Input

Some options and <u>Command Parameters</u> are mandatory and other optional. By default, <u>ascli</u> will ask for missing mandatory options or Command Parameters for interactive execution.

The behavior can be controlled with:

- --interactive=<yes|no> (default=yes if STDIN is a terminal, else no)
 - yes: missing mandatory parameters/arguments are asked to the user
 - no: missing mandatory parameters/arguments raise an error message
- --ask-options=<yes|no> (default=no)
 - optional parameters/arguments are asked to user

4.5 Output

Command execution will result in output (terminal, stdout/stderr). The information displayed depends on the action.

To redirect results to a file, use option output.

4.5.1 Types of output data

Depending on action, the output will contain:

Result Type	Description
single_object	Displayed as a 2 dimensional table: one line per field, first column is field name, and second is field value. Nested hashes are collapsed.
object_list	Displayed as a 2 dimensional table: one line per item, one column per field.
value_list	A table with one column.
empty	nothing
status	A message.
other_struct	A complex structure that cannot be displayed as an array.

4.5.2 Option: format

The style of output can be set using the format option:

format	Output formatting
table	Text table (default)
text	Value as String
ruby	Ruby code
json	JSON code
jsonpp	JSON pretty printed
yaml	YAML
CSV	Comma Separated Values
image	Image URL or data
nagios	Suitable for Nagios

By default, result of type single_object and object_list are displayed using format table.

4.5.3 Option: table_style

The table style can be customized with option: table_style which expects a Hash.

For format=table, options are the ones described in gem terminal-table.

For format=csv, options are described in gem csv.

For example, to display a table with thick Unicode borders:

```
ascli config preset over --table-style=@ruby:'{border: :unicode_thick_edge}'
```

For example, to display a CSV with headers and quotes:

```
ascli config echo @json:'[{"name":"foo","id":1},{"name":"bar","id":8}]' --format=csv

--table=@json:'{"headers":true,"force_quotes":true}'
```

• Note

Other border styles exist, not limited to: :unicode , :unicode_round .

4.5.4 Option: flat_hash: Single level Hash

This option controls how object fields are displayed for complex objects.

Effective only when format is table to display single_object or object_list.

If value is no, then object's field names are only the first level keys of the Hash result and values that are Hash are displayed as such in Ruby syntax.

If value is yes (default), then object are flattened, i.e. deep Hash are transformed into 1-level Hash, where keys are -junction of deep keys. In this case, it is possible to filter fields using the option fields using the compound field name using (dot) as separator.

Example: Result of command is a list of objects with a single object:

field	value
A	a
B.B1	b1
B.B2	b2
C.O.C1	c1
C.1.C2	c2
D.D1	d1
D.D2	d2
	İ

```
ascli config echo @json:'{"A":"a","B":[{"name":"B1","value":"b1"},{"name":"B2","value":"b2"}],"C":

→ [{"C1":"c1"},{"C2":"c2"}],"D":{"D1":"d1","D2":"d2"}}' -
-flat=no
```

```
field value

A          a
B          [{"name" => "B1", "value" => "b1"}, {"name" => "B2", "value" => "b2"}]
C          [{"C1" => "c1"}, {"C2" => "c2"}]
D          {"D1" => "d1", "D2" => "d2"}
```

4.5.5 Enhanced display of special values

Special values are highlighted as follows::

Value		Display	
nil		<null></null>	
empty	String	<empty< td=""><td>string></td></empty<>	string>
empty	Array	<empty< td=""><td>list></td></empty<>	list>
empty	Hash	<empty< td=""><td>dict></td></empty<>	dict>

Example:

ascli config echo @json:'{"ni":null,"es":"","ea":[],"eh":{}}'

4.5.6 Option: multi_single

This option controls how result fields are displayed as columns or lines, when option format is set to table. Default is no. There are two types of results that are affected by this option:

Result	Description
single_object	A single item with multiple fields.
object_list	A list of items, each with multiple fields.

An item (object) is displayed in one of those 2 ways:

Display	rows	columns
Simple	Fields	field and value
Transposed	Items	Fields

The display of result is as follows:

Result	no	yes	single
single_object	Simple	Simple	Simple
object_list	Transposed	Simple (Multiple objects)	Simple if 1 object, transposed if 2+ objects

This parameter can be set as a global default with:

ascli config preset set GLOBAL multi_single single

Examples:

Simulate a result by executing this command:

ascli config echo @json:'<json value here>' --multi-single=<no|yes|single>

Example 1: A list of one object

[{"user":{"id":1,"name":"toto"},"project":"blash"}]

Display with no (Transposed):

user.id	user.name	project
1	toto	blash

Display with yes and single (Simple):

field	value
user.id	1
user.name	toto
project	blash

Example 2: A list of two objects:

```
[{"id":1,"speed":111},{"id":"2","speed":222}]
```

Display with no and single (Transposed):

Display with yes (multiple Simple):

field	value
id speed	1 111
field	value
id speed	2 222

4.5.7 Option: display: Verbosity of output

Output messages are categorized in 3 types:

- info output contain additional information, such as number of elements in a table
- data output contain the actual output of the command (object, or list of objects)
- error output contain error messages

The option display controls the level of output:

- info displays all messages: info, data, and error
- data display data and error messages
- error display only error messages.

4.5.8 Option: show secrets: Hide or show secrets in results

- If value is no (default), then secrets are redacted from command results.
- If value is yes, then secrets shown in clear in results.
- If display is data, secrets are included to allow piping results.

4.5.9 Option: fields: Selection of output object fields

Depending on the command, results may include by default all fields, or only some selected fields. It is possible to define specific columns to be displayed, by setting the fields option.

The fields option is a list that can be either a comma separated list or an extended value Array.

Individual elements of the list can be:

- property: add property to the current list
- property: remove property from the current list

- DEF: default list of fields (that's the default, when not set)
- ALL: all fields
- A Ruby RegEx : using @ruby: '/.../', or @re:... add those matching to the list

Examples:

- a, b, c : the list of attributes specified as a comma separated list (overrides the all default)
- @json: '["a", "b", "c"] ': Array extended value: same as above
- b, DEF, -a : default property list, remove a and add b in first position
- @ruby: '/^server/' : Display all fields whose name begin with server

4.5.10 Option: select

Table output (object_list) can be filtered using option select. This option is either a Hash or Proc. The Proc takes as argument a line (Hash) in the table and is a Ruby lambda expression that shall return true to select or false to remove an entry.

Example:

```
ascli aoc admin user list --fields=name,email,ats_admin --query=@json:'{"sort":"name"}'

--select=@json:'{"ats_admin":true}'
```

name	email	ats_admin
John Curtis	john@example.com	true
Laurent Martin	laurent@example.com	true

Mote

Option select filters elements from the result of command, while the query option gives filtering parameters to the API when listing elements.

In above example, the same result is obtained with option:

```
--select=@ruby:'->(i){i["ats_admin"]}'
```

Option select applies the filter after a possible "flattening" with option: flat_hash.

4.5.11 Percent selector

The percent selector allows identification of an entity by another unique identifier other than the native identifier.

```
Syntax: %<field>:<value>
```

When a command is executed on a single entity, the entity is identified by a unique identifier that follows the command. For example, in the following command, 1234 is the user's identifier:

```
ascli aoc admin user show 1234
```

Some commands provide the following capability: If the entity can also be uniquely identified by a name, then the name can be used instead of the identifier, using the <u>percent selector</u>. For example, if the name of the user is <u>john</u> and a field for this entity named <u>name</u> has a value <u>john</u>:

```
ascli aoc admin user show %name:john
```

4.6 Extended Value Syntax

Most options and arguments are specified by a simple string (e.g. username or url). Sometimes it is convenient to read a value from a file: for example read the PEM value of a private key, or a list of files. Some options expect a more complex value such as Hash or Array.

The <u>Extended Value</u> Syntax allows to specify such values and even read values from other sources than the command line itself.

The syntax is:

<0 or more decoders><some text value or nothing>

Decoders act like a function with its parameter on right-hand side and are recognized by the prefix: **(a)** and suffix **(!)** The following decoders are supported:

	Parame-		
Decoder	ter	Returns	Description
base64	String	String	Decode a base64 encoded string
csvt	String	Array	Decode a titled CSV value
env	String	String	Read from a named env var name, e.gpassword=@env:MYPASSVAR
file	String	String	Read value from specified file (prefix ~/ is replaced with the users home folder), e.gkey=@file:~/.ssh/mykey
json	String	Any	Decode JSON values (convenient to provide complex structures)
lines	String	Array	Split a string in multiple lines and return an array
list	String	Array	Split a string in multiple items taking first character as separator and return an array
none	None	Nil	A null value
path	String	String	Performs path expansion on specified path (prefix ~/ is replaced with the users home folder), e.gconfig-file=@path:~/sample_config.yml
preset	String	Hash	Get whole option preset value by name. Sub-values can also be used, using as separator. e.g. foo.bar is conf[foo][bar]
extend	String	String	Evaluates embedded extended value syntax in string
re	String	Regexp	Ruby Regular Expression (short for @ruby://)
ruby	String	Any	Execute specified Ruby code
secret	None	String	Ask password interactively (hides input)
stdin	None	String	Read from stdin in text mode (no value on right)
stdbin	None	String	Read from stdin in binary mode (no value on right)
uri	String	String	Read value from specified URL, e.gfpac=@uri:http://serv/f.pac
val	String	String	Prevent decoders on the right to be decoded. e.gkey=@val:@file:foo sets
voml	Ctring	A 10. /	the option key to value @file:foo .
yaml	String	Any	Decode YAML
zlib	String	String	Decompress data using zlib

Note

A few commands support a value of type $\begin{tabular}{ll} Proc \\ @ruby:'->(i){i["attr"]}' \end{tabular}$ is a lambda expression that returns the value for key $\begin{tabular}{ll} attr \\ attr \\ \hline \end{tabular}$

To display the result of an extended value, use the config echo command.

The extend decoder is useful to evaluate embedded extended value syntax in a string. It expects a @ to close the embedded extended value syntax.

Example: Create a Hash value with the convenient @json: decoder:

ascli config echo @json:'{"key1":"value1","key2":"value2"}'

Example: read the content of the specified file, then, base64 decode, then unzip:

ascli config echo @zlib:@base64:@file:myfile.dat

Example: Create a Hash value with one key and the value is read from a file:

ascli config echo @ruby:'{"token_verification_key"=>File.read("mykey.txt")}'

Example: read a CSV file and create an Array of Hash for bulk provisioning:

cat test.csv

name,email

lolo,laurent@example.com

toto,titi@tutu.tata

ascli config echo @csvt:@file:test.csv

Example: create a Hash with values coming from a preset named config

ascli config echo @json:@extend:'{"hello":true,"version":"@preset:config.version@"{'

field	value
hello	true
version	4.21.1

Example: Create a Hash from YAML provided as shell Here document:

```
ascli config echo @yaml:@stdin: --format=json<<EOF
key1: value1
key2:
- item1
- item2
key3:
   key4: value4
   key5: value5
EOF
```

{"key1":"value1","key2":["item1","item2"],"key3":{"key4":"value4","key5":"value5"}}

4.7 Configuration and Persistency Folder

ascli configuration and persistency files (token cache, file lists, persistency files) are stored by default in [User's home folder]/.aspera/ascli.

• Note

[User's home folder] is found using Ruby's Dir.home (rb_w32_home_dir). It uses the HOME env var primarily, and on MS Windows it also looks at %HOMEDRIVE%%HOMEPATH% and %USERPROFILE%. ascli sets the env var %HOME% to the value of %USERPROFILE% if set and exists. So, on Windows %USERPROFILE% is used as it is more reliable than %HOMEDRIVE%%HOMEPATH%.

The configuration folder can be displayed using:

ascli config folder

/Users/kenji/.aspera/ascli

1 Note

This is equivalent to: ascli --show-config --fields=home

It can be overridden using option home.

Example (Windows):

```
set ASCLI_HOME=C:\Users\Kenji\.aspera\ascli
ascli config folder
C:\Users\Kenji\.aspera\ascli
```

When OAuth is used (AoC, Faspex5) ascli keeps a cache of generated bearer tokens in folder persist_store in configuration folder by default. Option cache_tokens (yes/no) allows controlling if OAuth tokens are cached on file system, or generated for each request. The command config tokens flush clears that cache. Tokens are kept on disk

for a maximum of 30 minutes (TOKEN_CACHE_EXPIRY_SEC) and garbage collected after that. When a token has expired, then a new token is generated, either using a refresh token if it is available, or by the default method.

4.8 Invalid Filename Characters

Some commands of ascli may create files or folders based on input that may contain invalid characters for the local file system. The option invalid_characters allows specifying a replacement character for a list of characters that are invalid in filenames on the local file system and replaces them with the specified character.

The first character specifies the replacement character, and the following characters are the invalid ones. This is used when a folder or file is created from a value that potentially contains invalid characters. For example, using the option package_folder. The default value is _<>: "/\|?*, corresponding to replacement character _ and characters not allowed on Windows.

• Note

This option is different from the replace_illegal_chars parameter in aspera.conf, which applies to transfers only.

4.9 Temporary files

Some temporary files may be needed during runtime. The temporary folder may be specified with option: temp_folder. Temporary files are deleted at the end of execution unless option: clean temp is set to no.

4.10 Configuration file

On the first execution of ascli, an empty configuration file is created in the configuration folder (ascli config folder). There is no mandatory information required in this file. The use of it is optional as any option can be provided on the command line.

Although the file is a standard YAML file, ascli provides commands to read and modify it using the config command.

All options for ascli can be set on command line, or by env vars, or using Option Preset in the configuration file.

A configuration file provides a way to define default values, especially for authentication options, thus avoiding to always having to specify those options on the command line.

The default configuration file is: \$\text{\$HOME/.aspera/ascli/config.yaml}\$ (this can be overridden with option --config-file=path or its env var).

The configuration file is a catalog of named lists of options, called: Option Preset. Then, instead of specifying some common options on the command line (e.g. address, credentials), it is possible to invoke the ones of an Option Preset (e.g. mypreset) using the option preset: --preset=mypreset or its shortcut: -Pmypreset.

4.10.1 Option Preset

An Option Preset is a collection of options and their associated values in a named section in the configuration file.

A named Option Preset can be modified directly using ascli, which will update the configuration file:

```
ascli config preset set|delete|show|initialize|update <option preset>
```

The command update allows the easy creation of Option Preset by simply providing the options in their command line format, e.g. :

```
ascli config preset update demo_server --url=ssh://demo.asperasoft.com:33001 --username=asperaweb --password=my_password_here --ts=@json:'{"precalculate_job_size":true}'
```

This creates an Option Preset demo_server with all provided options.

The command set allows setting individual options in an Option Preset.

```
ascli config preset set demo_server password my_password_here
```

The command initialize, like update allows to set several options at once, but it deletes an existing configuration instead of updating it, and expects a Hash Extended Value.

A full terminal based overview of the configuration can be displayed using:

```
ascli config preset over
```

A list of Option Preset can be displayed using:

```
ascli config preset list
```

A good practice is to not manually edit the configuration file and use modification commands instead. If necessary, the configuration file can be opened in a text editor with:

ascli config open

• Note

This starts the editor specified by env var **EDITOR** if defined.

The former format for commands is still supported:

```
ascli config preset set|delete|show|initialize|update <name>
ascli config preset over
ascli config preset list
```

It is possible to load an Option Preset from within another Option Preset using the preset option. For example if prommon is a preset with common options, and precific is a preset with specific options, then precific can load prommon using:

```
ascli config preset set pspecific preset pcommon
```

When pspecific is loaded, then cumulative option preset will be set, and it will also load pcommon.

4.10.2 Special Option Preset: config

This preset name is reserved and contains a single key: version. This is the version of ascli which created the file.

4.10.3 Special Option Preset: default

This preset name is reserved and contains an array of key-value, where the key is the name of a plugin, and the value is the name of another preset.

When a plugin is invoked, the preset associated with the name of the plugin is loaded, unless the option --no-default (or -N) is used.

• Note

Special plugin name: config can be associated with a preset that is loaded initially, typically used for default values.

Operations on this preset are done using regular config operations:

```
ascli config preset set default _plugin_name_ _default_preset_for_plugin_
ascli config preset get default _plugin_name_
"_default_preset_for_plugin_"
```

4.10.4 Plugin: config : Configuration

Plugin config provides general commands for ascli:

- Option Preset operations (configuration file)
- wizard

- vault
- ascp
- transferd

The default preset for config is read for any plugin invocation, this allows setting global options, such as --log-level or --interactive. When ascli starts, it looks for the default Option Preset and checks the value for config. If set, it loads the options independently of the plugin used.

• Note

If no global default is set by the user, <code>ascli</code> will use <code>global_common_defaults</code> when setting global options (e.g. <code>config ascp use</code>)

? Tip

If you don't know the name of the global preset, you can use GLOBAL to refer to it.

Show current default (global) Option Preset (config plugin):

```
ascli config preset get default config
global_common_defaults
```

Set a global parameter:

```
ascli config preset set GLOBAL version_check_days 0
```

If the default global Option Preset is not set, and you want to use a different name:

```
ascli config preset set GLOBAL version_check_days 0
```

ascli config preset set default config my_common_defaults

4.11 Tested commands for config

1 Note

Add ascli config in front of the following commands:

```
ascp connect info 'Aspera Connect for Windows'
ascp connect list
ascp connect version 'Aspera Connect for Windows' download 'Windows Installer' --to-folder=.
ascp connect version 'Aspera Connect for Windows' list
ascp connect version 'Aspera Connect for Windows' open documentation
ascp errors
ascp info --sdk-folder=sdk_test_dir
ascp install
ascp install --sdk-folder=sdk_test_dir
ascp install 1.1.3
ascp products list
ascp products use 'IBM Aspera Connect'
ascp schema --format=jsonpp
ascp show
ascp spec
check_update
coffee
coffee --ui=text
coffee --ui=text --image=@json:'{"text":true,"double":false}'
coffee --ui=text --image=@json:'{"text":true}'
detect https://faspex5.example.com/path
detect https://node.example.com/path
detect https://shares.example.com/path shares
detect https://tst.example.com/path faspio
detect https://tst.example.com/path httpgw
detect my_org aoc
```

```
doc
doc transfer-parameters
echo '<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"><circle cx="50" cy="50" r="50"

    fill="#006699"/></svg>' --format=image
echo -- --special-string
echo @base64:SGVsbG8gV29ybGQK
echo @csvt:@stdin:
echo @env:USER
echo @json:'[{"user":{"id":1,"name":"foo"},"project":"bar"}]' --<mark>multi-single=single</mark>
echo @json:'[{"user":{"id":1,"name":"foo"},"project":"bar"}]' --multi-single=yes
echo @lines:@stdin:
echo @list:,1,2,3
echo @secret:
echo @uri:/etc/hosts
echo @uri:file:/etc/hosts
echo @uri:http://ifconfig.me
echo @uri:https://ifconfig.me
echo @vault:my_preset.password
echo @zlib:@stdin:
echo hello
email_test --notify-to=my_email_external
file --log-level=debug --log-format=@ruby:'->(s, d, p, m){"#{d.strftime("%Y-%m-%d %H:%M:%S")} - #{p}
\rightarrow - #{s} - #{m}\n"}
file --log-level=debug --log-format=default
file --log-level=debug --log-format=standard
file --log-level=debug --logger=syslog
folder
gem name
gem path
gem version
genkey my_key
genkey my_key 4096
image https://eudemo.asperademo.com/wallpaper.jpg
initdemo
open
plugins create my_command
plugins list
preset delete conf name
preset initialize conf name @json:'{"p1":"v1","p2":"v2"}'
preset list
preset overview
preset set GLOBAL version check days 0
preset set conf_name param value
preset set default shares conf_name
preset show conf_name
preset unset conf_name param
preset update conf_name --p1=v1 --p2=v2
proxy_check --fpac=@file:proxy.pac https://eudemo.asperademo.com --proxy-credentials=@list:,user,pass
pubkey @file:my_key
remote_certificate chain https://node.example.com/path
remote_certificate name https://node.example.com/path
remote_certificate only https://node.example.com/path
smtp_settings
tokens flush
tokens list
tokens show foobar
transferd install
transferd list
vault create @json:'{"label":"my_label","password":"my_password_here","description":"my secret"}'
vault delete my label
vault info
vault list
vault show my_label
wizard https://console.example.com/path console
wizard https://faspex4.example.com/path faspex --username=test --password=test
wizard https://faspex5.example.com/path faspex5 --key-path=my_private_key
```

```
wizard https://node.example.com/path node --username=test --password=test
wizard https://orch.example.com/path orchestrator --username=test --password=test
wizard https://shares.example.com/path shares --username=test --password=test
wizard my_org aoc --key-path=my_private_key --username=my_user_email --use-generic-client=yes
wizard my_org aoc mypreset --key-path=my_private_key --username=my_user_email
```

4.11.1 Format of file

The configuration file is a Hash in a YAML file. Example:

```
config:
    version: 0.3.7

default:
    config: cli_default
    server: demo_server

cli_default:
    interactive: no

demo_server:
    url: ssh://demo.asperasoft.com:33001
    username: asperaweb
    password: my_password_here
```

We can see here:

- The configuration was created with ascli version 0.3.7
- The default Option Preset to load for server plugin is: demo_server
- The Option Preset demo_server defines some options: the URL and credentials
- The default Option Preset to load in any case is: cli_default

Two Option Presets are reserved:

- config contains a single value: version showing the version used to create the configuration file. It is used to check compatibility.
- default is reserved to define the default Option Preset name used for known plugins.

The user may create as many Option Preset as needed. For instance, a particular Option Preset can be created for a particular application instance and contain URL and credentials.

Values in the configuration also follow the Extended Value Syntax.

• Note

If the user wants to use the Extended Value Syntax inside the configuration file, using the config preset update command, the user shall use the <code>@val:</code> prefix.

Example:

ascli config preset set my_aoc_org private_key @val:@file:"\$HOME/.aspera/ascli/my_private_key"

This creates the Option Preset:

```
my_aoc_org:
   private_key: "@file:/Users/laurent/.aspera/ascli/my_private_key"
```

So, the key file will be read only at execution time, but not be embedded in the configuration file.

4.11.2 Evaluation order of options

Some options are global, some options are available only for some plugins. (the plugin is the first level command).

Options are loaded using this algorithm:

- If option --no-default (or -N) is specified, then no default value is loaded for the plugin
- Else it looks for the name of the plugin as key in section default, the value is the name of the default Option Preset for it, and loads it.
- If option --preset=<name or extended value hash> is specified (or -Pxxxx), this reads the Option Preset specified from the configuration file, or if the value is a Hash, it uses it as options values.
- · Environment variables are evaluated

· Command line options are evaluated

Options are evaluated in the order of command line.

To avoid loading the default Option Preset for a plugin, use: -N

On command line, words in option names are separated by a dash (). In configuration file, separator is an underscore. E.g. --xxx-yyy on command line gives xxx_yyy in configuration file.

The main plugin name is config, so it is possible to define a default Option Preset for the main plugin with:

```
ascli config preset set cli_default interactive no
ascli config preset set default config cli_default
```

An Option Preset value can be removed with unset:

```
ascli config preset unset cli_default interactive
```

Example: Define options using command line:

```
ascli -N --url=_url_here_ --password=my_password_here --username=_name_here_ node --show-config
```

Example: Define options using a Hash:

4.11.3 Wizard

The wizard is a command that asks the user for information and creates an Option Preset with the provided information for a given application.

It takes three optional arguments:

#	Parameter description	If not provided, ascli
1	URL or hostname of the application	ask for it.
2	The <u>plugin name</u> to limit search to	tries to detect known plugins from the URL
3	The <u>preset name</u> to save in config file	generates a unique preset name

Options are also available for the wizard:

Option	Value	Description
default	[yes]/no	Set as default configuration for specified plugin.
override	yes/[no]	Override existing default preset name for the plugin, if it exists.
key_path	path	Path to private key for JWT.

Other plugin-specific options can be provided to the wizard, such as --username, etc... They will be added to the Option Preset created by the wizard.

The simplest invocation is:

```
ascli config wizard
```

If the application requires a private key, the user can either provide the path to it with option key_path. The user is told where to place the associated public pey PEM in the application.

4.11.4 Example of configuration for a plugin

For Faspex, Shares, Node (including ATS, Aspera Transfer Service), Console, only username/password and URL are required (either on command line, or from configuration file). Those can be usually provided on the command line:

```
ascli shares repo browse / --url=https://10.25.0.6 --username=john --password=my_password_here
```

This can also be provisioned in a configuration file:

• Build Option Preset

```
ascli config preset set shares06 url https://10.25.0.6
ascli config preset set shares06 username john
ascli config preset set shares06 password my_password_here
```

This can also be done with one single command:

```
ascli config preset init shares06

→ @json:'{"url":"https://10.25.0.6","username":"john","password":"my_password_here"}'

or

ascli config preset update shares06 --url=https://10.25.0.6 --username=john

→ --password=my_password_here
```

• Define this Option Preset as the default Option Preset for the specified plugin (shares)

ascli config preset set default shares shares06

Display the content of configuration file in table format

ascli config preset overview

• Execute a command on the Shares application using default options

ascli shares repo browse /

4.12 Secret Vault

Secrets, e.g. passwords, keys, are needed when connecting to applications. Those secrets are usually provided as command options, on command line, env vars, files etc.

For security reasons, those secrets shall not be exposed in clear, either:

- On terminal during input
- In logs
- In command output

Instead, they shall be hidden (logs, output) or encrypted (configuration).

Terminal output (command result) secret removal is controlled by option show_secrets (default: no). Log secret removal is controlled by option log_secrets (default: no). Mandatory command line options can be requested interactively (e.g. password) using option interactive. It is possible to use extended value @secret:[name] to ask for a secret interactively. It is also possible to enter an option as an environment variable, e.g. ASCLI_PASSWORD for option password and read the env var like this:

```
read -s ASCLI_PASSWORD
export ASCLI_PASSWORD
```

Another possibility is to retrieve values from a secret vault.

The vault is used with options vault and vault_password.

vault shall be a Hash describing the vault:

```
{"type":"system","name":"ascli"}
```

vault_password specifies the password for the vault.

Although it can be specified on command line, for security reason you should avoid exposing the secret. For example, it can be securely specified on command line like this:

```
read -s ASCLI_VAULT_PASSWORD
export ASCLI_VAULT_PASSWORD
```

4.12.1 Vault: IBM HashiCorp Vault

https://developer.hashicorp.com/vault

Quick start macOS:

```
brew tap hashicorp/tap
brew install hashicorp/tap/vault
vault server -dev -dev-root-token-id=dev-only-token
```

Parameter	Example	Description
type	vault	The type of the vault
url	http://127.0.0.1:8200	The URL of the vault
token	dev-only-token	The token for the vault, by default uses parameter vault_password

--vault=@json:'{"type":"vault","url":"http://127.0.0.1:8200"}' --vault_password=dev-only-token

4.12.2 Vault: System keychain

Caution macOS only

It is possible to manage secrets in macOS keychain (only read supported currently).

Parameter	Example	Description
type name	system ascli	The type of the vault. The name of the keychain to use.

--vault=@json:'{"type":"system","name":"ascli"}'

4.12.3 Vault: Encrypted file

It is possible to store and use secrets encrypted in a file using option vault set to:

{"type":"file", "name":"vault.bin"}

Parameter Example Description

type file The type of the vault.
name vault.bin File path, absolute, or relative to the configuration folder ASCLI_HOME.

4.12.4 Vault: Operations

For this use the config vault command.

Then secrets can be manipulated using commands:

- create
- show
- list
- delete

ascli config vault create @json:'{"label":"mylabel","password":"my_password_here","description":"for this account"}'

4.12.5 Configuration Finder

When a secret is needed by a sub command, the command can search for existing configurations in the configuration file. The lookup is done by comparing the service URL and username (or access key).

4.12.6 Securing passwords and secrets

A password can be saved in clear in an Option Preset together with other account information (URL, username, etc...). Example:

```
ascli config preset update myconf --url=... --username=... --password=...
```

For a more secure storage one can do:

ascli config preset update myconf --url=... --username=... --password=@val:@vault:myconf.password
ascli config vault create @json:'{"label":"myconf","password":"my_password_here"}'

A Note

Use @val: in front of @vault: so that the extended value is not evaluated.

4.13 Private Key

Some Aspera applications allow the user to be authenticated using Public Key Cryptography:

- · for SSH: Server
- for OAuth JWT: AoC, Faspex5, Faspex, Shares

It consists in using a pair of associated keys: a private key and a public key. The same pair can be used for multiple applications. The file containing the private key (key pair) can optionally be protected by a passphrase. If the key is protected by a passphrase, then it will be prompted when used. (some plugins support option passphrase)

The following sample commands use the shell variable KEY_PAIR_PATH. Set it to the desired safe location of the private key. Typically, located in folder \$HOME/.ssh or \$HOME/.aspera/ascli. For example:

```
KEY_PAIR_PATH=~/.aspera/ascli/my_private_key
```

Several methods can be used to generate a key pair.

The format expected for keys is PEM.

If another format is used, such as DER, it can be converted to PEM, e.g. using openss1.

4.13.1 ascli for key generation

The generated key is of type RSA, by default: 4096 bit. For convenience, the public key is also extracted with extension .pub. The key is not passphrase protected.

```
ascli config genkey ${KEY_PAIR_PATH} 4096
```

To display the public key of a private key:

ascli config pubkey @file:\${KEY_PAIR_PATH}

4.13.2 ssh-keygen

Both private and public keys are generated, option -N is for passphrase.

```
ssh-keygen -t rsa -b 4096 -m PEM -N '' -f ${KEY_PAIR_PATH}
```

4.13.3 openss1

To generate a key pair with a passphrase the following can be used on any system:

```
openssl genrsa -passout pass:_passphrase_here_ -out ${KEY_PAIR_PATH} 4096
openssl rsa -pubout -in ${KEY_PAIR_PATH} -out ${KEY_PAIR_PATH}.pub
```

openss1 is sometimes compiled to support option -nodes (no DES, i.e. no passphrase, e.g. on macOS). In that case, add option -nodes instead of -passout pass:_passphrase_here_ to generate a key without passphrase.

If option -nodes is not available, the passphrase can be removed using this method:

```
openssl rsa -passin pass:_passphrase_here_ -in ${KEY_PAIR_PATH} -out ${KEY_PAIR_PATH}.no_des
mv ${KEY_PAIR_PATH}.no_des ${KEY_PAIR_PATH}
```

To change (or add) the passphrase for a key do:

```
openssl rsa -des3 -in ${KEY_PAIR_PATH} -out ${KEY_PAIR_PATH}.with_des
mv ${KEY_PAIR_PATH}.with_des ${KEY_PAIR_PATH}
```

4.13.4 Using an application to generate a key pair

Many applications are available, including on internet, to generate key pairs. For example: https://cryptotools.net/rsagen

Caution

Be careful that private keys are sensitive information, and shall be kept secret (like a password), so using online tools is risky.

4.14 Web service

Some plugins start a local web server. This server can server HTTP or HTTPS (with certificate):

The following parameters are supported:

Parameter	Туре	Default	Description
url	String	http://localhost:80	Base URL on which requests are listened, a path can be provided.
cert	String	-	(HTTPS) Path to certificate file (with extpfx or .p12 for PKCS12).
key	String	-	(HTTPS) Path to private key file (PEM), or passphrase for PKCS12.
chain	String	-	(HTTPS) Path to certificate chain (PEM only).

Parameter url (base URL) defines:

- If http or https is used
- The local port number (default 443 for HTTPS, 80 for HTTP)
- The base path, i.e. the path under which requests are received, if a reverse proxy is used this can be used to route.

4.15 Image and video thumbnails

ascli can display thumbnails for images and videos in the terminal. This is available:

- In the thumbnail command of node when using gen4/access key API.
- When using the show command of preview plugin.
- coffee and image commands of config plugin.
- Any displayed value which is a URL to image can be displayed with option format set to image

The following options can be specified in the option image:

Option	Туре	Description
reserve	Integer	Lines reserved to display a status.
text	Bool	Display text instead of image.
double	Bool	Display double text resolution (half characters).
font_ratio	Float	Font height/width ratio in terminal.

```
ascli config image https://eudemo.asperademo.com/wallpaper.jpg --ui=text
---image=@json:'{"text":true}'

curl -so - https://eudemo.asperademo.com/wallpaper.jpg | ascli config image @stdbin:

echo -n https://eudemo.asperademo.com/wallpaper.jpg | ascli config image @uri:@stdin:
```

4.16 Graphical Interactions: Browser and Text Editor

Some actions may require the use of a graphical tool:

- A browser for Aspera on Cloud authentication (web auth method)
- · A text editor for configuration file edition

By default, ascli assumes that a graphical environment is available on Windows, and on other systems, rely on the presence of the DISPLAY environment variable. It is also possible to force the graphical mode with option --ui:

- --ui=graphical forces a graphical environment, a browser will be opened for URLs or a text editor for file edition.
- --ui=text forces a text environment, the URL or file path to open is displayed on terminal.

4.17 Logging, Debugging

The gem is equipped with traces, mainly for debugging and learning APIs.

To increase debug level, use option log_level (e.g. using command line --log-level=xx , env var ASCLI_LOG_LEVEL , or an Option Preset).

• Note

When using the direct agent (ascp), additional transfer logs can be activated using ascp options and ascp_args, see direct.

By default, passwords and secrets are redacted from logs. Set option log secrets to yes to include secrets in logs.

Option	Values	Description
logger	stdout stderr syslog	Type of output. Default: stderr
log_level	trace2 trace1 debug info warn error	Minimum level displayed. Default: warn
log_secrets	yes	Show or hide secrets in logs. Default: no (Hide)
log_format	Proc String	The name of a formatter or a lambda function that formats the log (see below). Default: default Alternative: standard

Option log_format is typically set using @ruby: . It is a lambda that takes 4 arguments, see: Ruby Formatter: severity, time, progname, msg. The default formatter is:

```
->(s, _d, _p, m){"#{s[0..2]} #{m}\n"}
```

Available formatters for log_format:

Name	Description
default	Default formatter.
standard	Standard Ruby formatter.

Examples:

• Display debugging log on stdout:

ascli config pre over --log-level=debug --logger=stdout

• Log errors to syslog:

ascli config pre over --log-level=error --logger=syslog

Note

When ascli is used interactively in a shell, the shell itself will usually log executed commands in the history file (history | grep ascli).

4.18 Learning Aspera Product APIs (REST)

ascli uses mainly REST APIs to interact with Aspera applications.

To get traces of execution, with dump of API calls, use argument: --log-level=debug.

To display HTTP/S traffic set option log_level to trace2: --log-level=trace2. It will display the exact content of HTTP requests and responses.

4.19 HTTP socket parameters

To ignore SSL certificate for <u>any</u> address/port, use option: <u>insecure</u>, i.e. --insecure=yes. To ignore SSL certificate for a list of specific address/port, use option <u>ignore_certificate</u>, set to an <u>Array</u> of URL for which certificate will be ignored (only the address and port are matched), e.g. --ignore-certificate=@list:, https://127.0.0.1:9092

• Note

Ignoring certificate also applies to ascp WSS.

Ignoring a certificate is <u>not recommended</u>. It is preferable to add the certificate to the trusted store. When certificate validation is skipped, a warning is displayed. To disable this warning, set the option warn insecure to no.

HTTP connection parameters (not ascp WSS) can be adjusted using option <a href="http://https:/

Parameter	Туре	Default	Handler
read_timeout	Integer	60	Ruby
write_timeout	Integer	60	Ruby
open_timeout	Integer	60	Ruby
keep_alive_timeout	Integer	2	Ruby
ssl_options	Array	See below	Ruby
user_agent	Integer	ascli	ascli
download_partial_suffix	Integer	.http_partial	ascli
retry_on_error	Bool	false	ascli
retry_on_timeout	Bool	true	ascli
retry_on_unavailable	Bool	true	ascli
retry_max	Integer	1	ascli
retry_sleep	Integer	4	ascli

Time values are in set <u>seconds</u> and can be of type either <u>Integer</u> or <u>Float</u>. Default values are the ones of Ruby: For a full list, refer to the Ruby library: <u>Net::HTTP</u>.

Like any other option, those can be set either on command line, or in configuration file, either in a global preset or server-specific one.

Example:

ascli aoc admin package list --http-options=@json:'{"read_timeout":10.0}'

ssl_options corresponds to a list of options as listed in man SSL_CTX_set_options. The default initial value is the default of Ruby as specified in openssl/ssl.rb . Each option can be specified as a String with the same name as in the OpenSSL library by removing the prefix: SSL_OP_, or an Integer (e.g. O resets to no option). If the name appears in the list, the option is set. If the name appears in the list prefixed with a hyphen (-), the option is unset. For example to enable option SSL_OP_CIPHER_SERVER_PREFERENCE , add it to the list as CIPHER_SERVER_PREFERENCE . To disable option SSL_OP_SAFARI_ECDHE_ECDSA_BUG , add it as -SAFARI_ECDHE_ECDSA_BUG .

Example:

```
{"ssl options":["CIPHER SERVER PREFERENCE","-SAFARI ECDHE ECDSA BUG"]}
```

4.20 Proxy

There are several types of network connections, each of them use a different mechanism to define a (forward) proxy:

- REST calls (APIs) and HTTP Gateway
- ascp WSS and Legacy Aspera HTTP/S Fallback
- ascp SSH and UDP (Aspera FASP)

Refer to the following sections.

4.20.1 Proxy for REST and HTTP Gateway

REST API calls and transfers based on HTTP Gateway both use Ruby's Net::HTTP class. Refer to Ruby find proxy. When Ruby HTTP is used, there are two possibilities to define an HTTP proxy to be used.

The http_proxy environment variable (<u>lower case</u>) can be set to the <u>URL</u> of the proxy (with optional credentials). Syntax is: (http|https)://[user:password@]host:port . E.g. http://myproxy.org.net:3128 .

Note

Ruby expects a URL and myproxy.org.net:3128 alone is not valid.

Credentials for proxy are optional but can also be specified:

```
export http_proxy=http://my_user_here:my_pass_here@proxy.example.com:3128
```

Option http_proxy does the same (set env var) but on command line:

```
ascli --http-proxy=http://my_user_here:my_pass_here@host:port ...
```

Alternatively, the fpac option (function for proxy auto config) can be set to a Proxy Auto Configuration (PAC) JavaScript value.

Note that proxy credentials are not supported in PAC files.

To read the script from a URL (http: , https: and file:), use prefix: @uri: . A minimal script can be specified to define the use of a local proxy:

```
ascli --fpac='function FindProxyForURL(url, host){return "PROXY localhost:3128"}' ...
```

The result of a PAC file can be tested with command: config proxy_check. Example, using command line option:

```
ascli config proxy_check --fpac='function FindProxyForURL(url, host) {return "PROXY proxy.example.com:3128;DIRECT";}' http://example.com
```

PROXY proxy.example.com:1234;DIRECT

```
ascli config proxy_check --fpac=@file:./proxy.pac http://www.example.com
```

PROXY proxy.example.com:8080

ascli config proxy_check --fpac=@uri:http://server/proxy.pac http://www.example.com

```
PROXY proxy.example.com:8080
```

If the proxy found with the PAC requires credentials, then use option <code>proxy_credentials</code> with username and password provided as an <code>Array</code>:

```
ascli --proxy-credentials=@json:'["__username_here__","__password_here__"]' ...
ascli --proxy-credentials=@list::__username_here__:__password_here__ ...
```

4.20.2 Proxy for Legacy Aspera HTTP/S Fallback

Only supported with the direct agent: To specify a proxy for legacy HTTP fallback, use ascp native option -x and ascp_args: --transfer-info=@json:'{"ascp_args":["-x","url_here"]}'.

4.20.3 FASP proxy (forward) for transfers

To specify a FASP proxy (forward), set the transfer-spec parameter: proxy (only supported with the direct agent).

For example, for an Aspera forward proxy not encrypted (HTTP) without authentication running on port 9091, the option would be:

```
--ts=@json:'{"proxy":"dnat://proxy.example.org:9091"}'

Or, alternatively, (prefer transfer spec like above, generally):
--transfer-info=@json:'{"ascp_args":["--proxy","dnat://proxy.example.org:9091"]}'
```

4.21 FASP configuration

ascli uses one of the transfer agents to execute transfers.

By default, it uses the direct agent, which is basically a local ascp. Nevertheless, ascli does not come with ascp installed. This is the reason why it is advised to install the Aspera Transfer Daemon during installation (ascli config ascp install).

By default, ascli uses the ascp binary found in well known locations, i.e. typical Aspera product installation paths.

The way to specify the location of ascp is to use either options:

- ascp_path
- use_product

The config plugin allows finding and specifying the location of ascp. It provides the following commands for ascp sub-command:

- show: shows the path of ascp used
- use : specify the ascp path to use
- products: list Aspera transfer products available locally
- connect: list and download connect client versions available on internet

4.21.1 Show path of currently used ascp

```
ascli config ascp show

/Users/laurent/.aspera/ascli/sdk/ascp

ascli config ascp info

field value

ascp /Users/john/.aspera/sdk/ascp
...
```

4.21.2 Selection of ascp location for direct agent

By default, ascli uses any found local product with ascp, including Transfer Daemon.

To override and use an alternate | ascp | path use option | ascp_path (--ascp-path=)

For a permanent change, the command config ascp use sets the same option for the global default.

Using a POSIX shell:

```
ascli config ascp use @path:'~/Applications/Aspera CLI/bin/ascp'
ascp version: 4.0.0.182279
Updated: global_common_defaults: ascp_path <- /Users/laurent/Applications/Aspera CLI/bin/ascp
Saved to default global preset global_common_defaults
```

Windows:

ascli config ascp use C:\Users\admin\.aspera\ascli\sdk\ascp.exe

```
ascp version: 4.0.0.182279
Updated: global_common_defaults: ascp_path <- C:\Users\admin\.aspera\ascli\sdk\ascp.exe
Saved to default global preset global_common_defaults
```

If the path has spaces, read section: Shell and Command line parsing.

If option ascp_path is not set, then the product identified with option use_product is used.

If use_product is not set, then the first product found is used, this is equivalent to using option: --use-product=FIRST

Locally installed Aspera products can be listed with:

Using the option use_product finds the ascp binary of the selected product.

To permanently use the ascp of a product:

ascli config ascp connect list

```
ascli config ascp products use 'Aspera Connect' saved to default global preset /Users/laurent/Applications/Aspera Connect.app/Contents/Resources/ascp
```

4.21.3 Installation of Connect Client on command line

ascii coniig asch connect iist			
+			version
urn:uuid:589F9EE5-0489-4F73-998 urn:uuid:A3820D20-083E-11E2-892 urn:uuid:589F9EE5-0489-4F73-998 urn:uuid:55425020-083E-11E2-892 urn:uuid:D8629AD2-6898-4811-A46 urn:uuid:97F94DF0-22B1-11E2-81C	E-0800200C9A66 Aspera Con 2-A612FAC70C4E Aspera Con E-0800200C9A66 Aspera Con F-2AF386531BFF Aspera Con	nnect for Windows nnect for Windows 64-bit nnect for Windows XP nnect for Windows XP 64-bit nnect for Mac Intel nnect for Linux 64	3.11.2.63 3.11.2.63 3.11.2.63 3.11.2.63 3.11.2.63 3.11.2.63
ascli config ascp connect version	'Aspera Connect for Mac In	ntel' list	
	'		J
title → hreflang rel +	type 	href	·
Mac Intel Installer bin/IBMAsperaConnectInstaller enclosure	application/oc -3.11.2.63.dmg	ctet-stream	en
Mac Intel Installer → bin/IBMAsperaConnectInstaller → enclosure-one-click	application/oc OneClick-3.11.2.63.dmg	ctet-stream	en
Aspera Connect for Mac HTML Doc → https://www.ibm.com/docs/en/a → documentation	spera-connect/3.11?topic=as	 spera-connect-user-guide-mad	cos en
Aspera Connect for Mac Release → https://www.ibm.com/docs/en/a → release-notes		 otes-release-aspera-connect-	-3112 en
÷			+

ascli config ascp connect version 'Aspera Connect for Mac Intel' download enclosure --to-folder=.

Downloaded: IBMAsperaConnectInstaller-3.11.2.63.dmg

4.22 Transfer Clients: Agents

Some actions on Aspera Applications lead to file transfers (upload and download) using the FASP protocol (ascp). Transfers will be executed by a transfer client, here called Transfer Agent.

The following agents are supported and selected with option transfer:

transfer	location	Description of agent
direct	local	direct execution of ascp
transferd	local	Aspera Transfer Daemon
connect	local	Aspera Connect Client
desktop	local	Aspera for Desktop
node	remote	Aspera Transfer Node
httpgw	remote	Aspera HTTP Gateway

• Note

All transfer operations are seen from the point of view of the agent. For example, an agent executing an <u>upload</u>, or <u>package send</u> operation will effectively push files to the related server from the system where the agent runs.

All above agents (including direct) receive transfer parameters as a <u>transfer-spec</u>. Parameters in transfer-spec can be modified with option ts.

Specific options for agents are provided with option transfer_info.

Note

Parameters in transfer_info are specific for each agent type and are described in the agents respective sections.

4.22.1 Agent: Direct

The direct agent directly executes a local ascp in ascli. This is the default agent for ascli (option --transfer=direct). ascli will locally search installed Aspera products, including SDK, and use ascp from that component. Refer to section FASP.

The transfer_info option accepts the following optional parameters to control multi-session, Web Socket Session, Resume policy and add any argument to ascp:

Name	Type	Description
WSS	Bool	Web Socket Session Enable use of web socket session in case it is available Default: true
quiet	Bool	If true, then ascp progress bar is not shown. Default: false
trusted_certs client_ssh_key	Array String	List of repositories for trusted certificates. SSH Keys to use for token-based transfers.
offene_son_key	String	One of: dsa_rsa , rsa , per_client . Default: rsa
ascp_args	Array	Array of strings with native ascp arguments. Default: []
spawn_timeout_sec	Float	Multi session Verification time that ascp is running Default: 3

Name	Туре	Description
spawn_delay_sec	Float	Multi session Delay between startup of sessions Default: 2
multi_incr_udp	Bool	Multi Session Increment UDP port on multi-session If true, each session will have a different UDP port starting at fasp_port (or default 33001) Else, each session will use fasp_port (or ascp default) Default: true on Windows, else false
resume	Hash	Resume parameters. See below.
resume.iter_max	Integer	Max number of retry on error Default: 7
resume.sleep_initial	Integer	First Sleep before retry Default: 2
resume.sleep_factor	Integer	Multiplier of sleep period between attempts Default: 2
resume.sleep_max	Integer	Default: 60
monitor	Bool	Use management port. Default: true

In case of transfer interruption, the agent will <u>resume</u> a transfer up to <u>iter_max</u> time. Sleep between iterations is given by the following formula where <u>iter_index</u> is the current iteration index, starting at 0:

```
max( sleep_max, sleep_initial * sleep_factor ^ iter_index )
```

To display the native progress bar of ascp, use:

```
--progress-bar=no --transfer-info=@json:'{"quiet":false}'`
```

To skip usage of management port (which disables custom progress bar), set option monitor to false . In that, use the native progress bar:

```
--transfer-info=@json:'{"monitor":false,"quiet":false}'`
```

By default, Ruby's root CA store is used to validate any HTTPS endpoint used by ascp (e.g. WSS). In order to use a custom certificate store, use the trusted_certs option of direct agent's option transfer_info . To use ascp 's default, use option:

```
--transfer-info=@json:'{"trusted_certs":null}'`.
```

Some transfer errors are considered <u>retry-able</u> (e.g. timeout) and some other not (e.g. wrong password). The list of known protocol errors and retry level can be listed:

```
ascli config ascp errors
```

Examples:

```
ascli ... --transfer-info=@json:'{"wss":true,"resume":{"iter_max":20}}'
ascli ... --transfer-info=@json:'{"spawn_delay_sec":2.5,"multi_incr_udp":false}'
```

This can be useful to activate logging using option -L of ascp. For example, to activate debug level 2 for ascp. (DD), and display those logs on the terminal (-):

```
--transfer-info=@json:'{"ascp_args":["-DDL-"]}'
```

This is useful to debug if a transfer fails.

To store ascp logs in file aspera-scp-transfer.log in a folder, use --transfer-info=@json:'{"ascp_args":["-l

• Note

When transfer agent direct is used, the list of files to transfer is provided to ascp using either --file-list or --file-pair-list and a file list (or pair) file generated in a temporary folder. (unless --file-list or --file-pair-list is provided using transfer_info parameter ascp_args).

In addition to standard methods described in section File List, it is possible to specify the list of file using those additional methods:

• Using option transfer_info parameter ascp_args

```
--sources=@ts --transfer-info=@json:'{"ascp_args":["--file-list","myfilelist"]}'
```

Note

File lists is shown here, there are also similar options for file pair lists.

Note

Those 2 additional methods avoid the creation of a copy of the file list: if the standard options --sources=@lines:@file:... --src-type=... are used, then the file is list read and parsed, and a new file list is created in a temporary folder.

1 Note

Those methods have limitations: they apply <u>only</u> to the <u>direct</u> transfer agent (i.e. local <u>ascp</u>) and not for Aspera on Cloud.

4.22.1.1 Agent: Direct: aspera.conf: Virtual Links

This agent supports a local configuration file: aspera.conf where Virtual links can be configured:

On a server (HSTS), the following commands can be used to set a global virtual link:

```
asconfigurator -x 'set_trunk_data;id,1;trunk_name,in;trunk_capacity,45000;trunk_on,true'
asconfigurator -x 'set_trunk_data;id,2;trunk_name,out;trunk_capacity,45000;trunk_on,true'
asconfigurator -x 'set_node_data;transfer_in_bandwidth_aggregate_trunk_id,1'
asconfigurator -x 'set_node_data;transfer_out_bandwidth_aggregate_trunk_id,2'
```

But this command is not available on clients, so edit the file aspera.conf, you can find the location with: ascli config ascp info --fields=aspera_conf and modify the sections default and trunks like this for a global 100 Mbps virtual link:

```
<?xml version='1.0' encoding='UTF-8'?>
<CONF version="2">
    <default>
        <transfer>
            <in>
                 <bandwidth>
                     <aggregate>
                         <<u>trunk_id</u>>1</<u>trunk_id</u>>
                     </aggregate>
                 </bandwidth>
            </in>
            <0ut>
                 <bandwidth>
                     <aggregate>
                         <trunk_id>2</trunk_id>
                     </aggregate>
                 </bandwidth>
            </out>
        </transfer>
    </default>
    <trunks>
        <trunk>
            <id>1</id>
            <name>in</name>
            <on>true</on>
            <capacity>
                 <schedule format="ranges">1000000</schedule>
            </capacity>
        </trunk>
```

It is also possible to set a schedule with different time and days, for example for the value of schedule:

start=08 end=19 days=mon, tue, wed, thu capacity=900000;1000000

4.22.1.2 Agent: Direct: aspera.conf: File name with special characters

By default, ascp replaces file system disallowed characters in transferred file names with ... On Windows, it concerns: * : | < > " ' ? .

Replacing illegal characters in transferred file names is a built-in feature of Aspera transfers. This behavior can be customized with the aspera.conf configuration file. For example, let's assume we want to replace illegal character: with an underscore ___.

1. First, locate the configuration file with:

```
ascli conf ascp info --fields=aspera_conf
```

Typically, it is located at \$HOME/sdk/aspera.conf

1. Edit this file, and add the following line inside the XML section CONF.default.file_system:

```
<replace_illegal_chars>_|</replace_illegal_chars>
```

The result should look like this:

1. According to the documentation

The parameter works as follows:

- The first character in the value is the replacement character.
- All other characters listed after it are the illegal ones to be replaced.

So in this example: | will be replaced with _...

If the mounted target file system is <u>Windows</u> but <u>ascli</u> runs on Linux, there are several additional illegal characters you need to handle. In that case, you can set the parameter like this (note that XML special characters such as \leq , \geq , and | must be escaped properly):

```
<replace_illegal_chars>_*:|&lt;&gt;&quot;&apos;?</replace_illegal_chars>
```

In this example:

- _ is the replacement character.
- The list of characters that will be replaced is: *: |<>" '?

So, for example:

- report|final?.txt → report_final_.txt
- data*backup"2025".csv → data_backup_2025_.csv

4.22.2 Agent: Connect Client

By specifying option: --transfer=connect, ascli will start transfers using the locally installed <u>IBM Aspera</u> Connect Client. There are no option for transfer info.

4.22.3 Agent: Desktop Client

By specifying option: --transfer=desktop, ascli will start transfers using the locally installed <u>IBM Aspera</u> Desktop Client. There are no option for transfer_info.

4.22.4 Agent: Node API

By specifying option: --transfer=node, ascli starts transfers in an Aspera Transfer Server using the Node API, either on a local or remote node. This is especially useful for direct node-to-node transfers.

Parameters provided in option transfer_info are:

Parameter	Туре	Description
url	String	URL of the Node API Mandatory
username	String	Node API user or access key Mandatory
password	String	Password, secret or bearer token Mandatory
root_id	String	Root file ID Mandatory only for bearer token

Like any other option, transfer_info can get its value from a pre-configured Option Preset:

--transfer-info=@preset:_name_here_

or be specified using the extended value syntax:

--transfer-info=@json:'{"url":"https://...","username":"_user_here_","password":"my_password_here"}'

If transfer_info is not specified and a default node has been configured (name in node for section default) then this node is used by default.

If the password value begins with Bearer then the username is expected to be an access key and the parameter root_id is mandatory and specifies the file ID of the top folder to use on the node using this access key. It can be either the access key's root file ID, or any authorized file ID underneath it.

4.22.5 Agent: HTTP Gateway

The Aspera HTTP Gateway is a service that allows to send and receive files using HTTPS.

By specifying option: --transfer=httpgw, ascli will start transfers using the Aspera HTTP Gateway.

Parameters provided in option transfer_info are:

Name	Туре	Description
url	String	URL of the HTTP GW Mandatory
upload_chunk_size	Integer	Size in bytes of chunks for upload Default: 64000
api_version	String	Force use of version (v1, v2) Default: v2
synchronous	Bool	Wait for each message acknowledgment Default: false

Example:

ascli faspex package recv 323 --transfer=httpgw
--transfer-info=@json:'{"url":"https://asperagw.example.com:9443/aspera/http-gwy"}'

• Note

The gateway only supports transfers authorized with a token.

If the application, e.g. AoC or Faspex 5, is configured to use the HTTP Gateway, then ascli will automatically use the gateway URL if --transfer=httpgw is specified, so transfer_info becomes optional.

4.22.6 Agent: Transfer Daemon

Another possibility is to use the Transfer Daemon (transferd). Set option transfer to transferd. Options for transfer_info are:

Name	Туре	Description
url	String	IP address and port listened by the daemon Mandatory Default: :0
start	Bool	Start a new daemon. Default: true
stop	Bool	Stop daemon when exiting ascli Default: true

• Note

If port zero is specified in the URL, then the daemon will listen on a random available port. If no address is specified, then 127.0.0.1 is used.

For example, to use an external, already running transferd, use option:

```
--transfer-info=@json:'{"url":":55002","start":false,"stop":false}'
```

The gem grpc is not part of default dependencies, as it requires compilation of a native part. So, to use the Transfer Daemon you should install this gem:

gem install grpc

If the execution complains about incompatible libraries, then force recompilation of the native part:

```
gem uninstall grpc
gem install grpc --platform ruby
```

On Windows the compilation may fail for various reasons (3.1.1):

- cannot find -lx64-ucrt-ruby310
 - \rightarrow copy the file [Ruby main dir]\lib\libx64-ucrt-ruby310.dll.a to (remove the dll extension) [Ruby main dir]\lib\libx64-ucrt-ruby310.a
- conflicting types for 'gettimeofday'
 - → edit the file [Ruby main dir]/include/ruby-[version]/ruby/win32.h and change the signature of gettimeofday to gettimeofday(struct timeval *, void *), i.e. change struct timezone to void

4.23 Transfer Specification

Some commands lead to file transfer (upload/download). All parameters necessary for this transfer are described in a transfer-spec (Transfer Specification), such as:

- Server address
- Transfer username

- Credentials
- File list
- Etc...

ascli builds the <u>transfer-spec</u> internally as a <u>Hash</u>. It is not necessary to provide additional parameters on the command line for a transfer.

It is possible to modify or add any of the supported <u>transfer-spec</u> parameter using the <u>ts</u> option. The <u>ts</u> option accepts a <u>Hash</u> <u>Extended Value</u> containing one or several <u>transfer-spec</u> parameters. Multiple <u>ts</u> options on command line are cumulative, and the <u>Hash</u> value is deeply merged. To remove a (deep) key from transfer spec, set the value to <u>null</u>.

• Note

Default transfer spec values can be displayed with command:

```
ascli config ascp info --fields=ts --flat-hash=no
```

It is possible to specify ascp options when the transfer option is set to direct using transfer_info option parameter: ascp_args . Example: --transfer-info=@json:'{"ascp_args":["-l","100m"]}' . This is especially useful for ascp command line parameters not supported in the transfer spec.

The use of a transfer-spec instead of ascp command line arguments has the advantage of:

- Common to all Transfer Agent
- Not dependent on command line limitations (special characters...)

4.24 Transfer Parameters

All standard <u>transfer-spec</u> parameters can be specified. A <u>transfer-spec</u> can also be saved/overridden in the configuration file.

References:

- Aspera Node API Documentation → /opt/transfers
- Aspera Transfer Daemon Documentation → Guides → API Ref → Transfer Spec V1
- Aspera Connect SDK → search The parameters for starting a transfer.

Parameters can be displayed with commands:

```
ascli config ascp spec
ascli config ascp spec --select=@json:'{"d":"Y"}' --fields=-d,n,c
```

A JSON Schema can be generated with command:

```
ascli config ascp schema --format=jsonpp
```

An optional parameter can be specified to display the schema for a specific transfer agent:

```
ascli config ascp schema transferd --format=jsonpp
```

ascp argument or environment variable is provided in description.

ID	Name
Α	Direct
С	Connect
D	Desktop
Н	Httpgw
Ν	Node
Т	Transferd

Field	Type	Description
apply_local_docroot	boolean	Apply local docroot to source paths. (A, T) (apply-local-docroot)

Field	Type	Description			
authentication	string	Set to token for SSH bypass keys, else password asked if not provided. (C)			
cipher	string	In transit encryption algorithms. Allowed values: none, aes-128, aes-192, aes-256, aes-128-cfb, aes-192-cfb, aes-256-cfb, aes-128-gcm, aes-192-gcm, aes-256-gcm (-c (conversion){enum})			
cipher_allowed	string	Returned by node API. Valid literals include aes-128 and none. (C)			
content_protection	string	Enable client-side encryption at rest (CSEAR). (content protection) Allowed values: encrypt, decrypt (file-crypt={enum})			
content_protection_password	string	Specifies CSEAR password. (content protection) (env: ASPERA_SCP_FILEPASS)			
cookie	string	Metadata for transfer specified by application. (env: ASPERA_SCP_COOKIE)			
create_dir	boolean				
delete_before_transfer	boolean				
delete_source	boolean				
destination_root	string	Destination root directory.			
destination_root_id	string	The file ID of the destination root directory. Required when using Bearer token auth for the destination node. (T)			
dgram_size	integer	UDP datagram size in bytes. (-Z {integer})			
direction	string	Direction of transfer (on client side). Allowed values: send, receive (mode=(conversion) {enum})			
exclude_newer_than	string	Exclude files, but not directories, from the transfer if they are newer than the specified number of seconds added to the source computer's epoch. e.g86400 for newer than a day back. (exclude-newer-than={string})			
exclude_older_than	string	Exclude files, but not directories, from the transfer if they are older than the specified number of seconds added to the source computer's epoch. e.g86400 for older than a day back. (exclude-older-than={string})			
fail_bad_filepass	boolean	Fail on bad file decryption passphrase. (A, T) (fail-bad-filepass)			
fasp_port	integer	Specifies fasp (UDP) port. (-0 {integer})			
fasp_proxy	object	Proxy for communications between the remote server and the (local) client. (T)			
file_checksum	string	Enable checksum reporting for transferred files by specifying the hash to use. (A, N) Allowed values: sha-512, sha-384, sha-256, sha1, md5, none (file-checksum={enum})			

Field	Туре	Description	
http_fallback	boolean string	When true(1), attempts to perform an HTTP transfer if a FASP transfer cannot be performed. (-y (conversion) {boolean \ string })	
http_fallback_port	integer	Specifies HTTP port when no cipher is used. (-t {integer})	
https_fallback_port	integer	Specifies HTTPS port when cipher is used. (-t {integer})	
icos	object	Configuration parameters for IBM Cloud Object Storage (ICOS). (T)	
keepalive	boolean	The session is running in persistent session mode. (A, T) (keepalive)	
lock_min_rate	boolean	TODO: remove ? (C)	
lock_min_rate_kbps	boolean	If true, lock the minimum transfer rate to the value set for min_rate_kbps. If false, users can adjust the transfer rate up to the value set for target_rate_cap_kbps. (C, T)	
lock_rate_policy	boolean	If true, lock the rate policy to the default value. (C, T)	
lock_target_rate	boolean	TODO: remove ? (C)	
lock_target_rate_kbps	boolean	If true, lock the target transfer rate to the default value set for target_rate_kbps. If false, users can adjust the transfer rate up to the value set for target_rate_cap_kbps. (C, T)	
min_rate_cap_kbps	integer	The highest minimum rate that an incoming transfer can request, in kilobits per second. Client minimum rate requests that exceed the minimum rate cap are ignored. The default value of unlimited applies no cap to the minimum rate. (Default: 0) (C, T)	
min_rate_kbps	integer	Set the minimum transfer rate in kilobits per second. (-m {integer})	
move_after_transfer	string	Move source files to the specified archive-dir directory after they are transferred correctly. Available as of 3.8.0. Details in ascp manual. Requires write permissions on the source. If src_base is specified, files are moved to archive-dir / path-relative-to-srcbase . archive-dir must be in the same file system (or cloud storage account) as the source files being transferred. archive-dir is subject to the same docroot restrictions as source files. move_after_transfer and remove_after_transfer are mutually exclusive options. After files have been moved to the archive, the original source directory structure is left in place. Empty directories are not saved to archive-dir. To remove empty source directories after a successful move operation, also set remove_empty_directories to true. When using remove_empty_directories to true. When using remove_empty_directories , empty directory removal examination starts at the srcbase and proceeds down any subdirectories. If no srcbase is used and a file path (as opposed to a directory path) is specified, then only the immediate parent directory is examined and removed if it is empty following the move of the source file. (A, N, T) (move-after-transfer={string})	

Field	Туре	Description			
multi_session	integer	Use multi-session transfer. max 128. Each participant on one host needs an independent UDP (-0) port. Large files are split between sessions only when transferring with resume_policy = none.			
multi_session_threshold	integer	Split files across multiple ascp sessions if their size in bytes is greater than or equal to the specified value. (0=no file is split) (A, N, T) (multi-session-threshold={integer})			
obfuscate_file_names	boolean	HTTP Gateway obfuscates file names when set to true . (H)			
overwrite	string	Overwrite files at the destination with source files of the same name based on the policy: - always: Always overwrite the file never: Never overwrite the file. If the destination contains partial files that are older or the same as the source files and resume is enabled, the partial files resume transfer. Partial files with checksums or sizes that differ from the source files are not overwritten. - diff: (default) Overwrite the file if it is different from the source, depending on the compare method (default is size). If the destination is object storage, diff has the same effect as always. If resume is not enabled, partial files are overwritten if they are different from the source, otherwise they are skipped. If resume is enabled, only partial files with different sizes or checksums from the source are overwritten; otherwise, files resume. - diff+older: Overwrite the file if it is older and different from the source, depending on the compare method (default is size). If resume is not enabled, partial files are overwritten if they are older and different from the source, otherwise they are skipped. If resume is enabled, only partial files that are different and older than the source are overwritten, otherwise they are resumed. - older: Overwrite the file if its timestamp is older than the source timestamp. If you set an overwrite policy of diff or diff+older, difference is determined by the value set for resume policy: none: The source and destination files are always considered different and the destination file is always overwritten attributes: The source and destination files are compared based on sparse checksum; (currently file size) full_checksum: The source and destination files are compared based on full checksums Allowed values: never, always, diff, older, diff+older			
password	string	(overwrite={enum}) Password for local Windows user when transfer user associated with node API user is not the same as the one running asperanoded . Allows impersonating the transfer user and have access to resources (e.g. network shares). Windows only, node API only. (N)			
paths	array	Array of path to the source (required) and a path to the destination (optional).			
precalculate_job_size		Specifies whether to precalculate the job size. (precalculate-job-size)			
preserve_access_time		Preserve the source-file access timestamps at the destination. Because source access times are updated by the transfer operation, the timestamp that is preserved is the one just before to the transfer. (preserve-access-time)			
preserve_acls	string	Preserve access control lists. (A, T) Allowed values: none, native, metafile (preserve-acls={enum})			

Field	Type	Description		
preserve_creation_time	boolean	Preserve source-file creation timestamps at the destination. Only Windows systems retain information about creation time. If the destination is not a Windows computer, this option is ignored. (preserve-creation-time)		
preserve_extended_attrs	string	Preserve the extended attributes. (T, A) Allowed values: none, native, metafile (preserve-xattrs={enum})		
preserve_file_owner_gid	boolean	Preserve the group ID for a file owner. (A, T) (preserve-file-owner-gid)		
preserve_file_owner_uid	boolean	Preserve the user ID for a file owner. (A, T) (preserve-file-owner-uid)		
preserve_modification_time	boolean	of a transferred file to the modification of the source file or directory. Preserve source-file modification timestamps at the destination. (preserve-modification-time)		
preserve_remote_acls	string	Preserve remote access control lists. (A, T) Allowed values: none, native, metafile (remote-preserve-acls={enum})		
preserve_remote_ex- tended_attrs	string	Preserve remote extended attributes. (A, T) Allowed values: none, native, metafile (remote-preserve-xattrs={enum})		
preserve_source_access_time	boolean	Preserve the time logged for when the source file was accessed. (A, T) (preserve-source-access-time)		
preserve_times	boolean	Preserve file timestamps. (A, N, T) (-p {boolean})		
proxy	string	Specify the address of the Aspera high-speed proxy server. dnat(s)://[user[:password]@]server:port Default ports for DNAT and DNATS protocols are 9091 and 9092. Password, if specified here, overrides the value of environment variable ASPERA_PROXY_PASS . (A) (proxy={string})		
rate_policy	string	The transfer rate policy to use when sharing bandwidth. Allowable values: - high: When sharing bandwidth, transfer at twice the rate of a transfer using a fair policy fair: (Default) Share bandwidth equally with other traffic low: Use only unused bandwidth fixed: Transfer at the target rate, regardless of the actual network capacity. Do not share bandwidth. Aspera recommends that you do not use this setting except under special circumstances, otherwise the destination storage can be damaged. Allowed values: low, fair, high, fixed (policy={enum})		
rate_policy_allowed	string	Specifies most aggressive rate policy that is allowed. Returned by node API. (C) Allowed values: low, fair, high, fixed		
remote_access_key	string	The access key ID of the access key that was used to construct the bearer token that is used to authenticate to the remote node. (T)		
remote_host	string	IP or fully qualified domain name (FQDN) of the remote server. (host={string})		

Field	Туре	Description		
remote_password	string	SSH session password. (env: ASPERA_SCP_PASS)		
remote_user	string	Remote user. Default value is xfer on node or connect. (user={string})		
remove_after_transfer	boolean	Remove SRC files after transfer success. (A, N, T) (remove-after-transfer)		
remove_empty_directories	boolean	Specifies whether to remove empty directories. (A, N, T) (remove-empty-directories)		
remove_empty_source_dir	boolean	Remove empty source subdirectories and remove the source directory itself, if empty. (T)		
remove_empty_source_directory	boolean	Remove empty source subdirectories and remove the source directory itself, if empty. (A) (remove-empty-source-directory)		
remove_skipped	boolean	Must also have remove_after_transfer set to true. Defaults to false. If true, skipped files will be removed as well. (A, C, N) (remove-skipped)		
resume_policy	string	If a transfer is interrupted or fails to finish, this policy directs the transfer to resume without retransferring the files. Allowable values: - none: Always re-transfer the entire file. - attrs: Compare file attributes and resume if they match, and re-transfer if they do not. - sparse_csum: Compare file attributes and the sparse file checksums; resume if they match, and re-transfer if they do not. - full_csum: Compare file attributes and the full file checksums; resume if they match, and re-transfer if they do not. Note: transferd uses values: attributes, sparse_checksum, full_checksum. Allowed values: none, attrs, sparse_csum, full_csum(-k (conversion){enum})		
retry_duration	integer string	Specifies how long to wait before retrying transfer (e.g. 5min). (T)		
save_before_overwrite		If a transfer would result in an existing file . being overwritten, move that file to .yyyy.mm.dd.hh.mm.ss.index. (where index is set to 1 at the beginning of each new second and incremented for each file saved in this manner during the same second) in the same directory before writing the new file. File attributes are maintained in the renamed file. (A, N, T) (save-before-overwrite)		
skip_duplicate_check	boolean	Don't check for duplicate files at the destination. (A, T) (skip-dir-traversal-dupes)		
skip_special_files	boolean	All assets other than files, directories and symbolic links are considered special. A transfer will fail if the user attempts to transfer special assets. If true, ascp skips special assets and proceeds with the transfer of all other assets. (A, T) (skip-special-files)		
source_root	string	Path to be prepended to each source path. This is either a conventional path or it can be a URI but only if there is no root defined. (source-prefix64=(conversion) {string})		
source_root_id	string	The file ID of the source root directory. Required when using Bearer token auth for the source node. (N, T)		

Field	Type	Description			
src_base	string	Specify the prefix to be stripped off from each source object. The remaining portion of the source path is kept intact at the destination. Special care must be taken when used with cloud storage. (A, N, T) (src-base64=(conversion) {string})			
src_base64	string	The folder name below which the directory structure is preserved (base64 encoded). (A, T) (src-base64={string})			
ssh_args	array	Array of arguments to pass to SSH. Use with caution. (T, A) (ssh-args={array})			
ssh_port	integer	Specifies SSH (TCP) port. (-P {integer})			
ssh_private_key	string	Private key used for SSH authentication. Shall look like:BEGIN RSA PRIV4TE KEY\nMII Note the JSON encoding: \n for newlines. (A, T) (env: ASPERA_SCP_KEY)			
ssh_private_key_passphrase	string	The passphrase associated with the transfer user's SSH private key. Available as of 3.7.2. (A, T) (env: ASPERA_SCP_PASS)			
ssh_private_key_path	string	Path to private key for SSH. (A, T) (-i {string})			
sshfp	string	Check it against server SSH host key fingerprint. (check-sshfp={string})			
symlink_policy	string	Handle source side symbolic links. Allowed values: follow, copy, copy+force, skip (symbolic-links={enum})			
tags	object	Metadata for transfer as JSON. Key aspera is reserved. Key aspera.xfer_retry specifies a retry timeout for node API initiated transfers. (tags64=(conversion){object})			
tags64	string	Metadata for transfer as JSON. Key aspera is reserved. Key aspera.xfer_retry specifies a retry timeout for node API initiated transfers. (A, T) (tags64={string})			
target_rate_cap_kbps	integer	Maximum target rate for incoming transfers, in kilobits per second. Returned by upload/download_setup node API. (C, T)			
target_rate_kbps	integer	Specifies desired speed for the transfer. (-1 {integer})			
title	string	Title of the transfer. (C, N, T)			
token	string	Authorization token. Type: Bearer, Basic or ATM. (Also arg -W) (env: ASPERA_SCP_TOKEN)			
use_ascp4	boolean				
use_system_ssh	string	TODO, comment (A, T) (-SSH {string})			
wss_enabled	boolean	Server has Web Socket service enabled.			
wss_port	integer	TCP port used for websocket service feed.			
xfer_max_retries	integer	Maximum number of retries, for node API initiated transfers. Shall not exceed aspera.conf transfer_manager_max_retries (default 5). (N)			

4.24.1 Destination folder for transfers

The destination folder is set by ascli by default to:

- for downloads
- / for uploads

It is specified by the <u>transfer-spec</u> parameter <u>destination_root</u>. As such, it can be modified with option: --ts=@json:'{"destination_root":"<path>"}' . The option to_folder provides an equivalent and convenient way to change this parameter: --to-folder=<path> .

4.24.2 List of files for transfers

When uploading, downloading or sending files, the user must specify the list of files to transfer.

By default, the list of files to transfer is simply provided on the command line.

The list of (source) files to transfer is specified by (extended value) option sources (default: @args). The list is either simply the list of source files, or a combined source/destination list (see below) depending on value of option src_type (default: list).

In ascli, all transfer parameters, including file list, are provided to the transfer agent in a transfer is independent of the transfer agent (direct, connect, node, transfer daemon...). So, eventually, the list of files to transfer is provided to the transfer agent using the transfer-spec field: "paths" which is a list (array) of pairs of "source" (mandatory) and "destination" (optional). The sources and source-type options provide convenient ways to populate the transfer spec with the source file list.

Possible values for option sources are:

• @args : (default) the list of files (or file pair) is directly provided on the command line (after commands): unused arguments (not starting with) are considered as source files. So, by default, the list of files to transfer will be simply specified on the command line. Example:

```
ascli server upload ~/first.file secondfile
```

This is the same as (with default values):

ascli server upload --sources=@args --src-type=list ~/mysample.file secondfile

An Extended Value with type Array of String

9 Tip

Extended values can be tested with the command config echo

Examples:

· Using extended value

Create the file list:

```
echo ~/mysample.file > myfilelist.txt
echo secondfile >> myfilelist.txt
```

Use the file list: one path per line:

```
--sources=@lines<u>:@file</u>:myfilelist.txt
```

Using JSON array

```
--sources=@json:'["file1","file2"]'
```

• Using STDIN, one path per line

```
--sources=@lines:@stdin:
```

• Using Ruby code (one path per line in file)

```
--sources=@ruby: 'File.read("myfilelist.txt").split("\n")'
```

- Qts: the user provides the list of files directly in the paths field of transfer spec (option ts). Examples:
 - Using transfer spec

```
--sources=@ts --ts=@json:'{"paths":[{"source":"file1"},{"source":"file2"}]}'
```

The option src_type allows specifying if the list specified in option sources is a simple file list or if it is a file pair list.

• Note

Option src_type is not used if option sources is set to @ts

Supported values for src_type are:

- list : (default) the path of destination is the same as source and each entry is a source file path
- pair: the first element is the first source, the second element is the first destination, and so on.

Example: Source file 200KB.1 is renamed sample1 on destination:

ascli server upload --src-type=pair ~/Documents/Samples/200KB.1 /Upload/sample1

• Note

There are some specific rules to specify a file list when using Aspera on Cloud, refer to the AoC plugin section.

4.24.3 Source directory structure on destination

This section is not specific to ascli it is ascp behavior.

The transfer destination is normally expected to designate a destination folder.

But there is one exception: The destination specifies the new item name when the following are met:

- There is a single source item (file or folder)
- Transfer spec create_dir is not set to true (ascp option -d not provided)
- Destination is not an existing folder
- The dirname of destination is an existing folder

For this reason it is recommended to set create_dir to true for consistent behavior between single and multiple
items transfer, this is the default in ascli.

If a simple source file list is provided (no destination in paths, i.e. no file_pair_list provided), the destination folder is used as destination folder for each source file, and source file folder names are not preserved.

The inner structure of source items that are folder is preserved on destination.

A leading / on destination is ignored (relative to docroot) unless docroot is not set (relative to home).

In the following table source folder d3 contains 2 files: f1 and d4/f2.

Source files	Destina- tion	Folders on Dest.	create_d	Destination Files
f1	d/f	-	false	Error: d does not exist.
f1	d/f	d	false	d/f (renamed)
f1	d/f/.	d	false	d/f (renamed)
f1	d/f	d/f	false	d/f/f1
f1 f2	d	d	false	d/f1 $d/f2$
d3	d	-	false	d/f1 d/f2 (renamed)
f1	d	-	true	d/f1
f1 f2	d	-	true	d/f1 d/f2
d1/f1 d2/f2	d	-	true	d/f1 d/f2
d3	d	-	true	d/d3/f1 d/d3/d4/f2

If a file par list is provided then it is possible to rename or specify a different destination folder for each source (relative to the destination).

If transfer spec has a src_base, it has the side effect that the simple source file list is considered as a file pair list, and so the lower structure of source folders is preserved on destination.

Source files	Destination	<pre>src_base</pre>	Destination Files
d1/d2/f2 d1/d3/f3	d	d1	d/d2/f2 d/d3/f3

Advanced Example: Send files ./file1 and ./folder2/files2 to server (e.g. /Upload) and keep the original file names and folders, i.e. send file1 to /Upload/file1 and files2 to /Upload/folder2/files2.

- If files are specified as ./file1 ./folder2/files2,
 - then destination will be: /Upload/file1 /Upload/files2
- One possibility is to specify a file pair list: --src-type=pair file1 file1 folder2/files2 folder2/files2
- Another possibility is to specify a source base: --src-base=\$PWD \$PWD/file1 \$PWD/folder2/files2 (note that _ cannot be used as source base)
- Similarly, create a temporary soft link (Linux): In -s . tmp_base and use --src-base=tmp_base tmp_base/file1 t
- One can also similarly use --sources=@ts and specify the list of files in the paths field of transfer spec with both source and destination for each file.

4.24.4 Multi-session transfer

Multi session, i.e. starting a transfer of a file set using multiple sessions (one ascp process per session) is supported on direct and node agents, not yet on connect.

• --transfer=node

--ts=@json:'{"multi_session":10,"multi_session_threshold":1}'

Multi-session is directly supported by the node daemon.

• --transfer=direct

--ts=@json:'{"multi_session":5,"multi_session_threshold":1,"resume_policy":"none"}'

Mote

resume_policy set to attr may cause problems: none or sparse_csum shall be preferred.

ascli starts multiple ascp for Multi-session using direct agent.

When multi-session is used, one separate UDP port is used per session (refer to ascp manual page).

4.24.5 Content protection

Also known as Client-side encryption at rest (CSEAR), content protection allows a client to send files to a server which will store them encrypted (upload), and decrypt files as they are being downloaded from a server, both using a passphrase, only known by users sharing files. Files stay encrypted on server side.

Activating CSEAR consists in using transfer spec parameters:

- content_protection : activate encryption (encrypt for upload) or decryption (decrypt for download)
- content_protection_password : the passphrase to be used.

Example: parameter to download a Faspex package and decrypt on the fly

--ts=@json:'{"content_protection":"decrypt","content_protection_password":"my_password_here"}'

4.24.6 Transfer Spec Examples

Change target rate

--ts=@json:'{"target_rate_kbps":500000}'

Override the FASP SSH port to a specific TCP port:

--ts=@json:'{"ssh_port":33002}'

• Force HTTP fallback mode:

```
--ts=@json:'{"http_fallback":"force"}'
```

· Activate progress when not activated by default on server

```
--ts=@json:'{"precalculate_job_size":true}'
```

4.25 Transfer progress bar

File transfer operations are monitored, and a progress bar is displayed on the terminal if option progress_bar (Bool) is set to yes (default if the output is a terminal).

The same progress bar is used for any type of transfer, using ascp, server to server, using HTTPS, etc...

4.26 Scheduler

It is useful to configure automated scheduled execution. ascli does not provide an internal scheduler. Instead, use the service provided by the Operating system:

4.26.1 Windows Scheduler

Windows provides the Task Scheduler. It can be configured:

- Using utility schtasks.exe
- Using Powershell function <u>scheduletasks</u>
- Using taskschd.msc (UI)

4.26.2 Unix-like Scheduler

Unix-like systems (Linux, ...) provide cron, configured using a crontab

Linux also provides anacron, if tasks are hourly or daily.

For example, on Linux it is convenient to create a wrapping script, e.g. cron_ascli that will set up the environment (e.g. Ruby) to properly start ascli:

```
#!/bin/bash
# load the Ruby environment
. /etc/profile.d/rvm.sh
rvm use 2.6 --quiet
# set a timeout protection, just in case ascli is frozen
tmout=30m
# forward arguments to ascli
exec timeout ${tmout} ascli "${@}"
```

Example of cronjob created for user xfer.

```
crontab<<EOF
0 **** /home/xfer/cron_ascli preview scan --logger=syslog --display=error
2-59 *** /home/xfer/cron_ascli preview trev --logger=syslog --display=error
EOF</pre>
```

• Note

Logging options are kept here in the cron file instead of configuration file to allow execution on command line with output on command line.

4.27 Running as service

Some commands result in ascli running as a server, listening on a port. In this case, it is usually desirable to run ascli as a service. On Linux, typically, systemd is used.

A convenient way is to write a startup script, and run this script as a service.

Let's give a base name for our service: my_ascli_svc

The startup script can be simply the ascli command line, for example: /usr/local/bin/start_my_ascli_svc.sh

#!/bin/bash
set -e
echo "Starting my_ascli_svc at \$(date)"
set PATH to find ascli, and other environment setup
exec ascli

And make this script executable:

chmod a+x /usr/local/bin/start_my_ascli_svc.sh

Create a startup file: /etc/systemd/system/my_ascli_svc.service :

```
[Unit]
Description=My ascli daemon
After=network.target

[Service]
ExecStart=/usr/local/bin/start_my_ascli_svc.sh
Restart=on-failure
RestartSec=15
User=xfer
# Optional, specify a working directory
# WorkingDirectory=/path/to/working/dir
# Optional, redirect output
StandardOutput=journal
StandardError=journal
[Install]
WantedBy=multi-user.target
```

Then enable and start with:

```
sudo systemctl daemon-reload
sudo systemctl enable --now my_ascli_svc.service
systemctl status my_ascli_svc.service
journalctl -u my_ascli_svc.service
```

4.28 Locking for exclusive execution

In some cases one needs to ensure that ascli is not executed several times in parallel.

When ascli is executed automatically on a schedule basis, one generally desires that a new execution is not started if a previous execution is still running because an ongoing operation may last longer than the scheduling period:

- Executing instances may pile-up and kill the system
- The same file may be transferred by multiple instances at the same time.
- preview may generate the same files in multiple instances.

Usually the OS native scheduler already provides some sort of protection against parallel execution:

- The Windows scheduler does this by default
- Linux cron can leverage the utility flock to do the same:

```
/usr/bin/flock -w 0 /var/cron.lock ascli ...
```

ascli natively supports a locking mechanism with option lock_port. Technically, this opens a local TCP server port, and fails if this port is already used, providing a local lock. Lock is released when process exits. When using ascli in a container, this does not work with other containers, as each container have its own network.

Testing ascli locking:

Run this same command in two separate terminals within less than 30 seconds:

```
ascli config echo @ruby:'sleep(30)' --lock-port=12345
```

The first instance will sleep 30 seconds, the second one will immediately exit like this:

WARN -- : Another instance is already running (Address already in use - bind(2) for "127.0.0.1" port → 12345).

4.29 "Provençal"

ascp, the underlying executable implementing Aspera file transfer using FASP, has a capability to not only access the local file system (using system's open, read, write, close primitives), but also to do the same operations on other data storage such as S3, Hadoop and others. This mechanism is called PVCL (from Provençal, a restaurant located in Sophia Antipolis). Several PVCL adapters are available, one is embedded in ascp, the others are provided in shared libraries and must be activated.

The list of supported PVCL adapters can be retrieved with command:

```
ascli config ascp info --fields=@re:'^pvcl'
```

```
process v1
shares v1
noded v1
faux v1
file v1
stdio v1
stdio-tar v1
```

Here we can see the adapters: process, shares, noded, faux, file, stdio, stdio-tar.

Those adapters can be used wherever a file path is used in ascp including configuration. They act as a pseudo drive.

The simplified format is:

```
<adapter>:///<sub file path>?<arg1>=<val1>&...
```

One of the adapters, used in this manual, for testing, is faux. It is a pseudo file system allowing generation of file data without actual storage (on source or destination).

4.30 faux: for testing

This adapter can be used to simulate a file or a directory. This is a feature of ascp, not ascli. The following is an extract of the man page of ascp.

To discard data at the destination, the destination argument is set to faux://.

To send uninitialized data in place of an actual source file, the source file is replaced with an argument of the form:

faux:///filename?filesize

where:

- filename is the name that will be assigned to the file on the destination
- filesize is the number of bytes that will be sent (in decimal).

• Note

Characters ? and & are shell special characters (wildcard and background), so faux file specification on command line should be protected (using quotes or \). If not, the shell may give error: no matches found or equivalent.

For all sizes, a suffix can be added (case-insensitive) to the size: k, m, g, t, p, e (values are power of 2, e.g. 1M is 220, i.e. 1 mebibyte, not megabyte). The maximum allowed value is 8*260. Very large faux file sizes (petabyte range and above) will likely fail due to lack of destination storage unless destination is faux://.

To send uninitialized data in place of a source directory, the source argument is replaced with an argument of the form:

faux:///dirname?<arg1>=<val1>&...

where:

- dirname is the folder name and can contain / to specify a subfolder.
- Supported arguments are:

Name	Type	Description
count	int	Number of files Mandatory
file	string	Basename for files Default: file
size	int	Size of first file. Default: 0
inc	int	Increment applied to determine next file size Default: 0
seq	enum	Sequence in determining next file size Values: random, sequential Default: sequential
buf_init	enum	How source data is initialized Option none is not allowed for downloads. Values: none, zero, random Default: zero

The sequence parameter is applied as follows:

- If seq is random then each file size is:
 - size +/- (inc * rand())
 - Where rand is a random number between 0 and 1
 - Note that file size must not be negative, inc will be set to size if it is greater than size
 - Similarly, overall file size must be less than 8*260. If size + inc is greater, inc will be reduced to limit size + inc to 7*260.
- If seq is sequential then each file size is:
 - size + ((file_index 1) * inc)
 - Where first file is index 1
 - So file1 is size bytes, file2 is size + inc bytes, file3 is size + inc * 2 bytes, etc.
 - As with random, inc will be adjusted if size + (count * inc) is not less than 8*260.

Filenames generated are of the form: <file>_<00000 ... count>_<filesize>

Examples:

• Upload 20 gibibyte of random data to file myfile to directory /Upload

```
ascli server upload faux:///myfile\?20g --to-folder=/Upload
```

• Upload a file /tmp/sample but do not save results to disk (no docroot on destination)

```
ascli server upload /tmp/sample --to-folder=faux://
```

• Upload a faux directory mydir containing 1 million files, sequentially with sizes ranging from 0 to 2 Mebibyte - 2 bytes, with the base name of each file being testfile to /Upload

```
ascli server upload "faux:///mydir?file=testfile&count=1m&size=0&inc=2&seq=sequential"

---to-folder=/Upload
```

• Upload a faux directory mydir containing 1000 files, of size 1 byte, with the base name of each file being testfile to /Upload

```
ascli server upload "faux:///mydir?file=testfile&count=1000&size=1" --to-folder=/Upload
```

4.31 Usage

```
ascli -h
NAME
ascli -- a command line tool for Aspera Applications (v4.25.0.pre)
SYNOPSIS
ascli COMMANDS [OPTIONS] [ARGS]
```

```
DESCRIPTION
       Use Aspera application to perform operations on command line.
       Documentation and examples: https://rubygems.org/gems/aspera-cli
        execute: ascli conf doc
        or visit: https://www.rubydoc.info/gems/aspera-cli
        source repo: https://github.com/IBM/aspera-cli
ENVIRONMENT VARIABLES
       Any option can be set as an environment variable, refer to the manual
COMMANDS
       To list first level commands, execute: ascli
       Note that commands can be written shortened (provided it is unique).
OPTIONS
       Options begin with a '-' (minus), and value is provided on command line.
       Special values are supported beginning with special prefix @pfx:, where pfx is one of:
       val, base64, csvt, env, file, uri, json, lines, list, none, path, re, ruby, secret, stdin,
   stdbin, yaml, zlib, extend, preset, vault
       Dates format is 'DD-MM-YY HH:MM:SS', or 'now' or '-<num>h'
ARGS
       Some commands require mandatory arguments, e.g. a path.
OPTIONS: global
                                     Use interactive input of missing params: [no], yes
        --interactive=ENUM
       --ask-options=ENUM
                                     Ask even optional options: [no], yes
        --struct-parser=ENUM
                                     Default parser when expected value is a struct: json, ruby
        --format=ENUM
                                     Output format: text, nagios, ruby, json, jsonpp, yaml, [table],
   csv, image
        --output=VALUE
                                     Destination for results (String)
                                     Output only some information: [info], data, error
        --display=ENUM
        --fields=VALUE
                                     Comma separated list of: fields, or ALL, or DEF (String, Array,
   Regexp, Proc)
                                     Select only some items in lists: column, value (Hash, Proc)
       --select=VALUE
        --table-style=VALUE
                                     (Table) Display style (Hash)
        --flat-hash=ENUM
                                     (Table) Display deep values as additional keys: no, [yes]
        --multi-single=ENUM
                                     (Table) Control how object list is displayed as single table, or
   multiple objects: [no], yes, single
                                     Show secrets on command output: [no], yes
        --show-secrets=ENUM
        --image=VALUE
                                     Options for image display (Hash)
                                     Show this message
   -h, --help
                                     Generate bash completion for command
        --bash-comp
                                     Display parameters used for the provided action
        --show-config
   -v, --version
                                     Display version
        --ui=ENUM
                                     Method to start browser: text, [graphical]
                                     Replacement character and invalid filename characters
        --invalid-characters=VALUE
        --log-level=ENUM
                                     Log level: trace2, trace1, debug, info, [warn], error, fatal,
   unknown
        --log-format=VALUE
                                     Log formatter (Proc, Logger::Formatter, String)
        --logger=ENUM
                                     Logging method: [stderr], stdout, syslog
        --lock-port=VALUE
                                     Prevent dual execution of a command, e.g. in cron (Integer)
        --once-only=ENUM
                                     Process only new items (some commands): [no], yes
                                     Show passwords in logs: [no], yes
       --log-secrets=ENUM
                                     Cleanup temporary files on exit: no, [yes]
       --clean-temp=ENUM
        --pid-file=VALUE
                                     Write process identifier to file, delete on exit (String)
        --home=VALUE
                                     Home folder for tool (String)
        --config-file=VALUE
                                     Path to YAML file with preset configuration
                                     Secret for access keys
       --secret=VALUE
                                     Vault for secrets (Hash)
        --vault=VALUE
        --vault-password=VALUE
                                     Vault password
        --query=VALUE
                                     Additional filter for for some commands (list/delete) (Hash,
   Array)
        --property=VALUE
                                     Name of property to set (modify operation)
        --bulk=ENUM
                                     Bulk operation (only some): [no], yes
        --bfail=ENUM
                                     Bulk operation error handling: no, [yes]
    -N, --no-default
                                     Do not load default configuration for plugin
```

```
-P, --presetVALUE
                                     Load the named option preset from current config file
                                     Period in days to check new version (zero to disable)
        --version-check-days=VALUE
        --plugin-folder=VALUE
                                     Folder where to find additional plugins
        --override=ENUM
                                     Wizard: override existing value: [no], yes
        --default=ENUM
                                     Wizard: set as default configuration for specified plugin (also:
   update): no, [yes]
       --key-path=VALUE
                                     Wizard: path to private key for JWT
                                     Ascp: Path to ascp
        --ascp-path=VALUE
        --use-product=VALUE
                                     Ascp: Use ascp from specified product
        --sdk-url=VALUE
                                     Ascp: URL to get Aspera Transfer Executables
        --locations-url=VALUE
                                     Ascp: URL to get locations of Aspera Transfer Daemon
        --sdk-folder=VALUE
                                     Ascp: SDK folder path
                                     Display progress bar: [no], yes
        --progress-bar=ENUM
        --smtp=VALUE
                                     Email: SMTP configuration (Hash)
       --notify-to=VALUE
                                     Email: Recipient for notification of transfers
        --notify-template=VALUE
                                     Email: ERB template for notification of transfers
       --insecure=ENUM
                                     HTTP/S: Do not validate any certificate: [no], yes
                                     HTTP/S: Do not validate certificate for these URLs (Array)
       --ignore-certificate=VALUE
        --warn-insecure=ENUM
                                     HTTP/S: Issue a warning if certificate is ignored: no, [yes]
                                     HTTP/S: List of folder with trusted certificates (Array, String)
        --cert-stores=VALUE
       --http-options=VALUE
                                     HTTP/S: Options for HTTP/S socket (Hash)
       --http-proxy=VALUE
                                     HTTP/S: URL for proxy with optional credentials (String)
       --cache-tokens=ENUM
                                     Save and reuse OAuth tokens: no, [yes]
       --fpac=VALUE
                                     Proxy auto configuration script
                                     HTTP proxy credentials for fpac: user, password (Array)
       --proxy-credentials=VALUE
        --ts=VALUE
                                     Override transfer spec values (Hash)
       --to-folder=VALUE
                                     Destination folder for transferred files
        --sources=VALUE
                                     How list of transferred files is provided (@args,@ts,Array)
        --src-type=ENUM
                                     Type of file list: [list], pair
        --transfer=ENUM
                                     Type of transfer agent: desktop, node, [direct], transferd,
   httpgw, connect
        --transfer-info=VALUE
                                     Parameters for transfer agent (Hash)
COMMAND: config
SUBCOMMANDS: ascp check_update coffee detect documentation download echo email_test file folder gem
   genkey image initdemo open platform plugins preset proxy_check pubkey remote_certificate
   smtp_settings test tokens transferd vault wizard
COMMAND: shares
SUBCOMMANDS: admin files health
OPTIONS:
       --url=VALUE
                                     URL of application, e.g. https://app.example.com/aspera/app
                                     User's identifier
        --username=VALUE
       --password=VALUE
                                    User's password
COMMAND: node
SUBCOMMANDS: access_keys api_details asperabrowser async basic_token bearer_token browse cat central
→ delete download events health info license mkdir mkfile mklink rename search service simulator
   slash space ssync stream sync telemetry transfer transport upload watch_folder
OPTIONS:
                                     URL of application, e.g. https://app.example.com/aspera/app
        --url=VALUE
       --username=VALUE
                                     User's identifier
                                     User's password
        --password=VALUE
        --validator=VALUE
                                     Identifier of validator (optional for central)
       --asperabrowserurl=VALUE
                                     URL for simple aspera web ui
        --default-ports=ENUM
                                     Gen4: Use standard FASP ports (true) or get from node API
   (false): no, [yes]
        --node-cache=ENUM
                                     Gen4: Set to no to force actual file system read: no, [yes]
        --root-id=VALUE
                                     Gen4: File id of top folder when using access key (override AK
   root id)
        --dynamic-key=VALUE
                                     Private key PEM to use for dynamic key auth
COMMAND: faspio
SUBCOMMANDS: bridges health
```

```
OPTIONS:
                                     URL of application, e.g. https://app.example.com/aspera/app
        --url=VALUE
        --username=VALUE
                                     User's identifier
        --password=VALUE
                                     User's password
        --auth=ENUM
                                     OAuth type of authentication: jwt, basic
        --client-id=VALUE
                                     OAuth client identifier
                                     OAuth JWT RSA private key PEM value (prefix file path with
        --private-key=VALUE
   @file:)
        --passphrase=VALUE
                                     OAuth JWT RSA private key passphrase
COMMAND: orchestrator
SUBCOMMANDS: health info plugins processes workflow
OPTIONS:
        --url=VALUF
                                     URL of application, e.g. https://app.example.com/aspera/app
       --username=VALUE
                                     User's identifier
       --password=VALUE
                                     User's password
                                     Specify result value as: 'work_step:parameter'
       --result=VALUE
       --synchronous=ENUM
                                     Wait for completion: [no], yes
        --ret-style=ENUM
                                     How return type is requested in api: header, [arg], ext
        --auth-style=ENUM
                                     Authentication type: arg_pass, [head_basic], apikey
COMMAND: alee
SUBCOMMANDS: entitlement health
OPTIONS:
        --url=VALUE
                                     URL of application, e.g. https://app.example.com/aspera/app
       --username=VALUE
                                     User's identifier
                                     User's password
        --password=VALUE
COMMAND: ats
SUBCOMMANDS: access_key api_key aws_trust_policy cluster
OPTTONS:
        --ibm-api-key=VALUE
                                     IBM API key, see https://cloud.ibm.com/iam/apikeys
        --instance=VALUE
                                     ATS instance in ibm cloud
        --ats-key=VALUE
                                     ATS key identifier (ats_xxx)
        --ats-secret=VALUE
                                     ATS key secret
        --params=VALUE
                                     Parameters access key creation (@json:)
        --cloud=VALUE
                                     Cloud provider
        --region=VALUE
                                     Cloud region
COMMAND: faspex5
SUBCOMMANDS: admin bearer_token gateway health invitations packages postprocessing shared_folders

→ user version

OPTIONS:
                                     URL of application, e.g. https://app.example.com/aspera/app
       --url=VALUE
                                     User's identifier
       --username=VALUE
       --password=VALUE
                                     User's password
                                     OAuth client identifier
       --client-id=VALUE
       --client-secret=VALUE
                                     OAuth client secret
                                     OAuth redirect URI for web authentication
       --redirect-uri=VALUE
        --auth=ENUM
                                     OAuth type of authentication: web, [jwt], boot
        --private-key=VALUE
                                     OAuth JWT RSA private key PEM value (prefix file path with
   @file:)
        --passphrase=VALUE
                                     OAuth JWT RSA private key passphrase
        --box=VALUE
                                     Package inbox, either shared inbox name or one of: inbox,
   inbox_history, inbox_all, inbox_all_history, outbox, outbox_history, pending, pending_history,
   all or ALL
        --shared-folder=VALUE
                                     Send package with files from shared folder
        --group-type=ENUM
                                     Type of shared box: [shared_inboxes], workgroups
COMMAND: cos
SUBCOMMANDS: node
OPTIONS:
```

```
--bucket=VALUE
                                     Bucket name
        --endpoint=VALUE
                                     Storage endpoint (URL)
       --apikey=VALUE
                                     Storage API key
        --crn=VALUE
                                     Resource instance id (CRN)
        --service-credentials=VALUE IBM Cloud service credentials (Hash)
        --region=VALUE
                                     Storage region
        --identity=VALUE
                                     Authentication URL (https://iam.cloud.ibm.com/identity)
COMMAND: httpgw
SUBCOMMANDS: health info
OPTIONS:
        --url=VALUE
                                     URL of application, e.g. https://app.example.com/aspera/app
COMMAND: faspex
SUBCOMMANDS: address_book dropbox health login_methods me package source v4
OPTIONS:
                                     URL of application, e.g. https://app.example.com/aspera/app
        --url=VALUE
                                     User's identifier
       --username=VALUE
                                     User's password
       --password=VALUE
        --link=VALUE
                                     Public link for specific operation
       --delivery-info=VALUE
                                     Package delivery information (Hash)
       --remote-source=VALUE
                                     Remote source for package send (id or %name:)
       --storage=VALUE
                                     Faspex local storage definition (for browsing source)
        --recipient=VALUE
                                     Use if recipient is a dropbox (with *)
        --box=ENUM
                                     Package box: [inbox], archive, sent
COMMAND: preview
SUBCOMMANDS: check events scan show test trevents
OPTIONS:
        --url=VALUE
                                     URL of application, e.g. https://app.example.com/aspera/app
        --username=VALUE
                                     User's identifier
                                     User's password
        --password=VALUE
        --skip-format=ENUM
                                     Skip this preview format (multiple possible): png, mp4
        --folder-reset-cache=ENUM
                                     Force detection of generated preview by refresh cache: [no],
   header, read
       --skip-types=VALUE
                                     Skip types in comma separated list
       --previews-folder=VALUE
                                     Preview folder in storage root
                                     Path to temp folder
        --temp-folder=VALUE
        --skip-folders=VALUE
                                     List of folder to skip
        --base=VALUE
                                     Basename of output for for test
        --scan-path=VALUE
                                     Subpath in folder id to start scan in (default=/)
        --scan-id=VALUE
                                     Folder id in storage to start scan in, default is access key
   main folder id
       --mimemagic=ENUM
                                     Use Mime type detection of gem mimemagic: [no], yes
       --overwrite=ENUM
                                     When to overwrite result file: always, never, [mtime]
        --file-access=ENUM
                                     How to read and write files in repository: [local], remote
        --max-size=VALUE
                                     Maximum size (in bytes) of preview file
        --thumb-vid-scale=VALUE
                                     Png: video: size (ffmpeg scale argument)
        --thumb-vid-fraction=VALUE
                                     Png: video: time percent position of snapshot
        --thumb-img-size=VALUE
                                     Png: non-video: height (and width)
        --thumb-text-font=VALUE
                                     Png: plaintext: font for text rendering: `magick identify -list
   font`
                                     Mp4: method for preview generation: [reencode], blend, clips
        --video-conversion=ENUM
        --video-png-conv=ENUM
                                     Mp4: method for thumbnail generation: [fixed], animated
       --video-scale=VALUE
                                     Mp4: all: video scale (ffmpeg scale argument)
       --video-start-sec=VALUE
                                     Mp4: all: start offset (seconds) of video preview
        --reencode-ffmpeg=VALUE
                                     Mp4: reencode: options to ffmpeg
                                     Mp4: blend: # key frames
        --blend-keyframes=VALUE
        --blend-pauseframes=VALUE
                                     Mp4: blend: # pause frames
        --blend-transframes=VALUE
                                     Mp4: blend: # transition blend frames
                                     Mp4: blend: frame per second
       --blend-fps=VALUE
        --clips-count=VALUE
                                     Mp4: clips: number of clips
        --clips-length=VALUE
                                     Mp4: clips: length in seconds of each clips
```

```
COMMAND: aoc
SUBCOMMANDS: admin automation bearer_token files gateway organization packages reminder servers

→ tier restrictions user

OPTIONS:
        --url=VALUE
                                     URL of application, e.g. https://app.example.com/aspera/app
        --username=VALUE
                                     User's identifier
                                     User's password
        --password=VALUE
        --auth=ENUM
                                     OAuth type of authentication: web, [jwt]
       --client-id=VALUE
                                     OAuth API client identifier
        --client-secret=VALUE
                                     OAuth API client secret
       --scope=VALUE
                                     OAuth scope for AoC API calls
                                     OAuth API client redirect URI
       --redirect-uri=VALUE
       --private-key=VALUE
                                     OAuth JWT RSA private key PEM value (prefix file path with
   @file:)
       --passphrase=VALUE
                                     RSA private key passphrase (String)
       --workspace=VALUE
                                     Name of workspace (String, NilClass)
       --new-user-option=VALUE
                                     New user creation option for unknown package recipients (Hash)
                                     Validate shared inbox metadata: no, [yes]
        --validate-metadata=ENUM
        --package-folder=VALUE
                                     Field of package to use as folder name, or @none: (String,
   NilClass)
COMMAND: server
SUBCOMMANDS: browse cp delete df download du health info ls md5sum mkdir mv rename rm sync upload
OPTIONS:
        --url=VALUE
                                     URL of application, e.g. https://app.example.com/aspera/app
       --username=VALUE
                                     User's identifier
                                     User's password
        --password=VALUE
        --ssh-keys=VALUE
                                     SSH key path list (Array or single)
       --passphrase=VALUE
                                     SSH private key passphrase
        --ssh-options=VALUE
                                     SSH options (Hash)
COMMAND: console
SUBCOMMANDS: health transfer
OPTIONS:
        --url=VALUE
                                     URL of application, e.g. https://app.example.com/aspera/app
        --username=VALUE
                                     User's identifier
        --password=VALUE
                                     User's password
```

• Note

Commands and parameter values can be written in short form.

4.32 Bulk creation and deletion of resources

Bulk creation and deletion of resources are possible using option bulk (yes, no (default)). In that case, the operation expects an Array of Hash instead of a simple Hash using the Extended Value Syntax. This option is available only for some resources: if you need it: try and see if the entities you try to create or delete support this option.

4.33 Option: query

The query option can generally be used to add URL parameters to commands that list resources. It takes either a Hash or an Array, corresponding to key/value pairs that appear in the query part of request.

```
For example: --query=@json:'{"p1":"v1","p2":"v2"}' leads to query: ?p1=v1&p2=v2.
```

If the same parameter needs to be provided several times, then it's possible as well to provide an Array or 2-element Array: --query=@json:'[["p1":,"v1"],["p2":"v2"]]' leads to the same result as previously.

If PHP's style array is used, then one can use either:

```
--query=@json:'{"a":["[]","v1","v2"]}'--query=@json:'[["a[]","v1"],["a[]","v2"]]'
```

Both result in: a[]=v1&a[]=v2.

4.34 Plugins

ascli uses a plugin mechanism. The first level command (just after ascli on the command line) is the name of the concerned plugin which will execute the command. Each plugin usually represents commands sent to a specific application. For instance, the plugin faspex allows operations on Aspera Faspex.

Available plugins can be found using command:

ascli config plugin list				
+ plugin +	+ detect	+ wizard	+	
shares node 	Y Y	Y Y	/aspera-cli/lib/aspera/cli/plugins/shares.rb /aspera-cli/lib/aspera/cli/plugins/node.rb	+

Most plugins will take the URL option: url to identify their location.

REST APIs of Aspera legacy applications (Aspera Node, Faspex 4, Shares, Console, Orchestrator) use simple user-name/password authentication: HTTP Basic Authentication using options: username and password.

Aspera on Cloud and Faspex 5 rely on OAuth.

By default, plugins are looked-up in folders specified by (multi-value) option plugin_folder:

```
ascli --show-config --fields=plugin_folder
```

You can create the skeleton of a new plugin like this:

```
ascli config plugin create foo .
Created ./foo.rb
ascli --plugin-folder=. foo
```

4.35 Plugins vs Transfer Agents

Plugins typically represent a specific remote application on which ascli can operate, while transfer agents are the underlying components that handle the actual data transfer.

A given remote application can sometimes be both a plugin and a transfer agent. For example: node and httpgw are both plugins and transfer agents.

A plugin is invoked as the first positional argument in a command line. A Transfer Agent is used by setting the option transfer (e.g. --transfer=node).

Chapter 5

Plugin: aoc: IBM Aspera on Cloud

Aspera on Cloud API requires the use of OAuth v2 mechanism for authentication (HTTP Basic authentication is not supported).

It is recommended to use the wizard to set it up, although manual configuration is also possible.

5.1 AoC configuration: Using Wizard

ascli provides a configuration wizard.

The wizard guides you through the steps to create a new configuration preset for Aspera on Cloud.

The first optional argument is the URL of your Aspera on Cloud instance or simply the organization name, i.e. one of those :

- _your_organization_
- _your_organization_.ibmaspera.com
- https://_your_organization_.ibmaspera.com

The second optional argument can also be provided to specify the plugin name, e.g. acc for Aspera on Cloud. If optional arguments are not provided, the wizard will ask interactively and try to detect the application.

Here is a sample invocation:

```
$ ascli config wizard
Using: Aspera on Cloud at https://_my_org_.ibmaspera.com
Path to private RSA key (leave empty to generate):
option: key_path>
Using existing key:
/home/john/.aspera/ascli/my_private_key.pem
Please Log in as user laurent.martin.aspera@fr.ibm.com at: https://_my_org_.ibmaspera.com
Navigate to: (User) → Account Settings → Profile → Public Key
Check or update the value to (including BEGIN/END lines):
   --BEGIN PUBLIC KEY--
MIICIjANBg....
....kCAwEAAQ==
----END PUBLIC KEY----
Once updated or validated, press [Enter].
Using global client_id.
Preparing preset: aoc_my_org_ibmaspera_com_john_example_com
Setting config preset as default for aoc
You can test with:
ascli aoc user profile show
Saving config file.
```

Note

In above example, replace https://_my_org_.ibmaspera.com with your actual AoC URL.

Optionally, it is possible to create a new organization-specific integration, i.e. client application identification. For this, specify the option: --use-generic-client=no.

If you already know the application, and want to limit the detection to it, provide URL and plugin name:

ascli config wizard _my_org_ aoc

• Note

In above example, replace _my_org_ with the first part of your actual AoC URL: https://_my_org_.ibmaspera.com.

After successful completion of the wizard, a new configuration preset is created, and set as default for the acc plugin. This can be verified with command:

ascli config preset over

5.2 AoC configuration: Using manual setup

• Note

If you used the wizard (recommended): skip this section.

5.2.1 AoC manual configuration: Details

Several types of OAuth authentication are supported:

- JSON Web Token (JWT): authentication is secured by a private key (recommended)
- Web based authentication: authentication is made by user using a browser
- URL Token: external user's authentication with URL tokens (public links)

The authentication method is controlled by option auth.

For a <u>quick start</u>, follow the mandatory and sufficient section: API Client Registration (auth=web) as well as [Option Preset] (#option-preset) for Aspera on Cloud.

For a more convenient, browser-less, experience follow the JWT section (auth=jwt) in addition to Client Registration.

In OAuth, a <u>Bearer</u> token is generated to authenticate REST calls. Bearer tokens are valid for a period of time defined (by the AoC app, configurable by admin) at its creation. <u>ascli</u> saves generated tokens in its configuration folder, tries to re-use them or regenerates them when they have expired.

5.2.2 API Client Registration

Optional

If you use the built-in client id and client secret, skip this and do not set them in next section.

Else you can use a specific OAuth API client_id, the first step is to declare ascli in Aspera on Cloud using the admin interface.

(AoC documentation: Registering an API Client).

Let's start by a registration with web based authentication (auth=web):

- Open a web browser, log to your instance: e.g. https://_my_org_.ibmaspera.com/ (use your actual AoC instance URL)
- Go to Apps → Admin → Organization → Integrations
- Click Create New
 - Client Name: ascli

- Redirect URIs: http://localhost:12345
- Origins: localhost
- uncheck Prompt users to allow client to access
- leave the JWT part for now
- Save

• Note

For web based authentication, ascli listens on a local port (e.g. specified by the redirect_uri, in this example: 12345), and the browser will provide the OAuth code there. For ascli, HTTP is required, and 12345 is the default port.

Once the client is registered, a Client ID and Secret are created, these values will be used in the next step.

5.2.3 Configuration for Aspera on Cloud

If you did not use the wizard, you can also manually create an Option Preset for ascli in its configuration file.

Let's create an Option Preset called: my_aoc_org using ask for interactive input (client info from previous step):

```
ascli config preset ask my_aoc_org url client_id client_secret
option: url> https://_my_org_.ibmaspera.com/
option: client_id> my_client_id_here
option: client_secret> my_client_secret_here
updated: my_aoc_org
```

• Note

In above example, replace https://_my_org_.ibmaspera.com with your actual AoC URL.

(This can also be done in one line using the command config preset update my_aoc_org --url=...)

Define this Option Preset as default configuration for the aspera plugin:

ascli config preset set default aoc my_aoc_org

Note

Default auth method is web and default redirect_uri is http://localhost:12345 . Leave those default values.

5.2.4 Authentication with private key

For a Browser-less, Private Key-based authentication, use the following steps.

In order to use JSON Web Token (JWT) for Aspera on Cloud API client authentication, a private/public key pair must be used.

5.2.4.1 API Client JWT activation

If you are not using the built-in client_id and secret, JWT needs to be authorized in Aspera on Cloud. This can be done in two manners:

- Graphically
 - Open a web browser, log to your instance: https://_my_org_.ibmaspera.com/ (Use your actual AoC instance URL)
 - Go to Apps → Admin → Organization → Integrations
 - Click on the previously created application
 - select tab: JSON Web Token Auth
 - Modify options if necessary, for instance: activate both options in section Settings
 - Save
- Using command line

ascli aoc admin client list

Items: 4/4

id	name
oXPUyJ7JpQ TaoAmAG8Rg TDN12bLZqw VTh92i50fQ	PRI Sydney ascli_test_web ascli_web shannon

```
ascli aoc admin client modify my_BJbQiFw

→ @json:'{"jwt_grant_enabled":true,"explicit_authorization_required":false}'
```

modified

5.2.5 User key registration

The public key must be assigned to your user. This can be done in two manners:

5.2.5.1 Graphically

Open the previously generated public key located here: \$HOME/.aspera/ascli/my_private_key.pub

- Open a web browser, log to your instance: https://_my_org_.ibmaspera.com/ (Use your actual AoC instance URL)
- Click on the user's icon (top right)
- Select Account Settings
- Paste the Public Key PEM value in the Public Key section
- · Click on Submit

5.2.5.2 Using command line

ascli aoc admin user list

id	name	email
1234567	John Doe	john@example.com
7654321	Alice Saprich	alice@example.com
1234321	Sponge Bob	bob@example.com

```
ascli aoc user profile modify

→ @ruby:'{"public_key"=>File.read(File.expand_path("~/.aspera/ascli/my_private_key.pub"))}'
```

modified



The aspera user info show command can be used to verify modifications.

5.2.6 Option Preset modification for JWT

To activate default use of JWT authentication for ascli using the Option Preset, do the following:

- · Change auth method to JWT
- Provide location of private key
- Provide username to login as (OAuth subject)

Execute:

```
ascli config preset update my_aoc_org --auth=jwt
---private-key=@val:@file:~/.aspera/ascli/my_private_key --username=someuser@example.com
```

Note

The private key argument represents the actual PEM string. In order to read the content from a file, use the <code>@file:</code> prefix. But if the <code>@file:</code> argument is used as is, it will read the file and set in the configuration file. So, to keep the <code>@file:</code> tag in the configuration file, the <code>@val:</code> prefix is added.

After this last step, commands do not require web login anymore.

5.2.7 Public and private links

AoC gives the possibility to generate public links for both the Files and Packages modules. Public links embed the authorization of access. Provide the public link using option url alone.

In addition, the Files application supports private links. Private links require the user to authenticate. So, provide the same options as for regular authentication, and provide the private link using option url.

A user may not be part of any workspace, but still have access to shared folders (using private links). In that case, it is possible to list those shared folder by using a value for option workspace equal to @none: or @json:null or @ruby:nil.

5.2.8 AoC: First Use

Once client has been registered and Option Preset created: ascli can be used:

```
ascli aoc files br /
Current Workspace: Default Workspace (default)
empty
```

5.3 Calling AoC APIs from command line

The command ascli aoc bearer can be used to generate an OAuth token suitable to call any AoC API (use the scope option to change the scope, default is user:all). This can be useful when a command is not yet available.

Example:

It is also possible to get the bearer token for node, as user or as admin using:

```
ascli aoc files bearer_token_node /
ascli aoc admin node v4 1234 --secret=_ak_secret_here_ bearer_token_node /
```

5.4 Administration

The admin command allows several administrative tasks (and require admin privilege).

It allows actions (create, update, delete) on <u>resources</u>: users, group, nodes, workspace, etc... with the admin resource command.

5.4.1 Listing resources

The command aoc admin <type> list lists all entities of given type. It uses paging and multiple requests if necessary.

The option query can be optionally used. It expects a Hash using Extended Value Syntax, generally provided using: --query=@json: {...} . Values are directly sent to the API call and used as a filter on server side.

The following parameters are supported:

- q: a filter on name of resource (case-insensitive, matches if value is contained in name)
- sort : name of fields to sort results, prefix with for reverse order.

- max : maximum number of items to retrieve (stop pages when the maximum is passed)
- pmax : maximum number of pages to request (stop pages when the maximum is passed)
- page: native API parameter, in general do not use (added by ascli)
- per_page: native API parameter, number of items par API call, in general do not use
- Other specific parameters depending on resource type.

Both max and pmax are processed internally in ascli, not included in actual API call and limit the number of successive pages requested to API. ascli will return all values using paging if not provided.

Other parameters are directly sent as parameters to the GET request on API.

page and per_page are normally added by ascli to build successive API calls to get all values if there are more than 1000. (AoC allows a maximum page size of 1000).

q and sort are available on most resource types.

Other parameters depend on the type of entity (refer to AoC API).

Examples:

• List users with laurent in name:

```
ascli aoc admin user list --query=@json:'{"q":"laurent"}'
```

• List users who logged-in before a date:

```
ascli aoc admin user list --query=@json:'{"q":"last_login_at:<2018-05-28"}'
```

• List external users and sort in reverse alphabetical order using name:

```
ascli aoc admin user list --query=@json:'{"member_of_any_workspace":false,"sort":"-name"}'
```

Refer to the AoC API for full list of query parameters, or use the browser in developer mode with the web UI.

```
Tip
```

The option select can also be used to further refine selection, refer to section earlier.

5.4.2 Selecting a resource

Resources are identified by a unique id, as well as a unique name (case-insensitive).

To execute an action on a specific resource, select it using one of those methods:

- recommended: give ID directly on command line after the action: aoc admin node show 123
- Give name on command line after the action: aoc admin node show name abc
- Provide option id : aoc admin node show 123
- Provide option name : aoc admin node show --name=abc

5.4.3 Creating a resource

New resources (users, groups, workspaces, etc...) can be created using a command like:

```
ascli aoc admin create <resource type> @json:'{<...parameters...>}'
```

Some API endpoints are described in IBM API Hub. Sadly, not all.

Nevertheless, it is possible to guess the structure of the creation value by simply dumping an existing resource, and use the same parameters for the creation.

```
ascli aoc admin group show 12345 --format=json
```

```
{"created_at":"2018-07-24T21:46:39.000Z","description":<u>null</u>,"id":"12345","manager":<u>false</u>,"name":

A8Demo

WS1","owner":<u>false</u>,"queued_operation_count":0,"running_operation_count":0,

"stopped_operation_count":0,"updated_at":"2018-07-24T21:46:39.000Z","saml_group":<u>false</u>,

"saml_group_dn":<u>null</u>,"system_group":<u>true</u>,"system_group_type":"workspace_members"}
```

Remove the parameters that are either obviously added by the system: id, created_at, updated_at or optional. And then craft your command:

```
ascli aoc admin group create @json:'{"wrong":"param"}'
```

If the command returns an error, example:

```
ERROR: Rest: found unpermitted parameter: :wrong code: unpermitted_parameters request_id: 2a487dbc-bc5c-41ab-86c8-3b9972dfd4c4 api.ibmaspera.com 422 Unprocessable Entity
```

Well, remove the offending parameters and try again.

• Note

Some properties that are shown in the web UI, such as membership, are not listed directly in the resource, but instead another resource is created to link a user and its group: group_membership

5.4.4 Access Key secrets

In order to access some administrative actions on <u>nodes</u> (in fact, access keys), the associated secret is required. The secret is provided using the <u>secret</u> option. For example in a command like:

```
ascli aoc admin node 123 --secret="my_secret_here" v3 info
```

It is also possible to store secrets in the secret vault and then automatically find the related secret using the config finder.

5.4.5 Activity

The activity app can be queried with:

```
ascli aoc admin analytics transfers
```

It can also support filters and send notification using option notify_to. A template is defined using option
notify_template :

mytemplate.erb:

```
From: <%=from_name%> <<%=from_email%>>
To: <<%=ev['user_email']%>>
Subject: <%=ev['files_completed']%> files received

Dear <%=ev[:user_email.to_s]%>,
We received <%=ev['files_completed']%> files for a total of <%=ev['transferred_bytes']%> bytes,

starting with file:
<%=ev['content']%>

Thank you.
```

The environment provided contains the following additional variable:

• ev : all details on the transfer event

Example:

```
ascli aoc admin analytics transfers --once-only=yes --lock-port=12345
--query=@json:'{"status":"completed","direction":"receive"}' --notify-to=active
--notify-template=@file:mytemplate.erb
```

Options:

- once_only keep track of last date it was called, so next call will get only new events
- query filter (on API call)
- notify send an email as specified by template, this could be places in a file with the @file modifier.

Note

This must not be executed in less than 5 minutes because the analytics interface accepts only a period of time between 5 minutes and 6 months. The period is [date of previous execution]..[now].

5.4.6 Transfer: Using specific transfer ports

By default, transfer nodes are expected to use ports TCP/UDP 33001. The web UI enforces that. The option default_ports ([yes]/no) allows ascli to retrieve the server ports from an API call (download_setup) which reads the information from aspera.conf on the server.

5.4.7 Using ATS

Refer to section Examples of ATS and substitute command ats with acc admin ats.

5.4.8 Files with type link

Aspera on Cloud Shared folders are implemented through a special type of file: link. A link is the equivalent of a symbolic link on a file system: it points to another folder (not file).

Listing a link (in terminal position of path) will information on the link itself, not the content of the folder it points to. To list the target folder content, add a / at the end of the path.

Example:

5.4.9 Example: Bulk creation of users

5.4.10 Example: Find with filter and delete

```
| 98399 | deleted |
+-----+
```

5.4.11 Example: Find deactivated users since more than 2 years

```
ascli aoc admin user list --query=@ruby:'{"deactivated"=>true,"q"=>"last_login_at:<#{(DateTime.now. | optime.utc-2*365*86400).iso8601}"}'
```

To delete them use the same method as before

5.4.12 Example: Display current user's workspaces

ascli aoc user workspaces list

+ id	+ name	
16 17 18	Engineering Marketing Sales +	

5.4.13 Example: Create a sub access key in a node

Creation of a sub-access key is like creation of access key with the following difference: authentication to Node API is made with access key (master access key) and only the path parameter is provided: it is relative to the storage root of the master key. (id and secret are optional)

```
ascli aoc admin resource node --name=_node_name_ --secret=_secret_ v4 access_key create

Graph == --secret=_secret_ v4 access_key create
```

5.4.14 Example: Display transfer events (ops/transfer)

```
ascli aoc admin node --secret=_secret_ v3 transfer list --query=@json:'[["q","*"],["count",5]]'

Examples of query:
```

```
{"q":"type(file_upload OR file_delete OR file_download OR file_rename OR folder_create OR

→ folder_delete OR folder_share OR folder_share_via_public_link)","sort":"-date"}
```

{"tag": aspera.files.package_id=LA80U3p8w"}

5.4.15 Example: Display node events (events)

ascli aoc admin node --secret=_secret_ v3 events

5.4.16 Example: Display members of a workspace

```
ascli aoc admin workspace_membership list --fields=member_type,manager,member.email
---query=@json:'{"embed":"member","inherited":false,"workspace_id":11363,"sort":"name"}'
```

member_type
user user user user group user

Other query parameters:

```
{"workspace_membership_through":true,"include_indirect":true}
```

5.4.17 Example: Add all members of a workspace to another workspace

a- Get ID of first workspace

```
WS1='First Workspace'
WS1ID=$(ascli aoc admin workspace list --query=@json:'{"q":"'"$WS1"'"}'

→ --select=@json:'{"name":"'"$WS1"'"}' --fields=id --format=csv)
```

b- Get ID of second workspace

```
WS2='Second Workspace'
WS2ID=$(ascli aoc admin workspace list --query=@json:'{"q":"'"$WS2"'"}'

→ --select=@json:'{"name":"'"$WS2"'"}' --fields=id --format=csv)
```

c- Extract membership information

```
ascli aoc admin workspace_membership list --fields=manager,member_id,member_type,workspace_id
--query=@json:'{"workspace_id":'"$WS1ID"'}' --format=jsonpp --output=ws1_members.json
```

d- Convert to creation data for second workspace:

or, using jq:

e- Add members to second workspace

ascli aoc admin workspace_membership create --bulk=yes @json:@file:ws2_members.json

5.4.18 Example: Get users who did not log since a date

ascli aoc admin user list --fields=email --query=@json:'{"q":"last_login_at:<2018-05-28"}'

5.4.19 Example: List Limited users

ascli aoc admin user list --fields=email --select=@json:'{"member_of_any_workspace":false}'

5.4.20 Example: Create a group, add to workspace and add user to group

Create the group and take note of id

```
ascli aoc admin group create @json:'{"name":"group 1","description":"my super group"}'
```

Group: 11111

Get the workspace ID

```
ascli aoc admin workspace list --query=@json:'{"q":"myworkspace"}' --fields=id --format=csv

→ --display=data
```

Workspace: 22222

· Add group to workspace

```
ascli aoc admin workspace_membership create

Gison:'{"workspace_id":22222,"member_type":"user","member_id":11111}'
```

· Get a user's ID

```
ascli aoc admin user list --query=@json:'{"q":"manu.macron@example.com"}' --fields=id --format=csv
--display=data
```

User: 33333

Add user to group

```
ascli aoc admin group_membership create

Graph of the companies of the com
```

5.4.21 Example: Perform a multi Gbps transfer between two remote shared folders

In this example, a user has access to a workspace where two shared folders are located on different sites, e.g. different cloud regions.

First, set up the environment (skip if already done)

```
ascli config wizard --url=https://sedemo.ibmaspera.com --username=someuser@example.com
Detected: Aspera on Cloud
Preparing preset: aoc_sedemo
Using existing key:
/Users/laurent/.aspera/ascli/aspera_aoc_key
Using global client_id.
Please Login to your Aspera on Cloud instance.
Navigate to your "Account Settings"
Check or update the value of "Public Key" to be:
----BEGIN PUBLIC KEY----
SOME PUBLIC KEY PEM DATA HERE
----END PUBLIC KEY----
Once updated or validated, press enter.
creating new config preset: aoc_sedemo
Setting config preset as default for aspera
saving configuration file
You can test with:
```

This creates the option preset <code>aoc_[org name]</code> to allow seamless command line access and sets it as default for Aspera on Cloud.

Then, create two shared folders located in two regions, in your files home, in a workspace.

Then, transfer between those:

ascli aoc user profile show

5.4.22 Example: Create registration key to register a tethered node

The following command will create and display a secret token to register a self-managed Aspera Transfer Server:

```
ascli aoc admin client_registration_token create @json:'{"data":{"name":"laurentnode",<sub>|</sub>

--fields=token
--show-secrets=yes
```

5.4.23 Example: Delete all registration keys

```
+----+
| id | status |
+----+
| 99 | deleted |
| 100 | deleted |
| 101 | deleted |
| 102 | deleted |
+----+
```

5.4.24 Example: Create a Node

AoC nodes as actually composed with two related entities:

- An access key created on the Transfer Server (HSTS/ATS)
- A node resource in the AoC application.

The web UI allows creation of both entities in one shot. For more flexibility, ascli allows this in two separate steps.

Note

When selecting Use existing access key in the web UI, this actually skips access key creation (first step).

So, for example, the creation of a node using ATS in IBM Cloud looks like (see other example in this manual):

· Create the access key on ATS

The creation options are the ones of ATS API, refer to the section on ATS for more details and examples.

Once executed, the access key id and secret, randomly generated by the Node API, is displayed.

Note

Once returned by the API, the secret will not be available anymore, so store this preciously. ATS secrets can only be reset by asking IBM support.

· Create the AoC node entity

First, Retrieve the ATS node address

```
ascli aoc admin ats cluster show --cloud=softlayer --region=eu-de --fields=transfer_setup_url
--format=csv
```

Then use the returned address for the url key to actually create the AoC Node entity:

Creation of a node with a self-managed node is similar, but the command acc admin ats access_key create is replaced with node access_key create on the private node itself.

5.5 List of files to transfer

Source files are provided as a list with the sources option. By default, simply the list of files on the command line. Refer to section File list.

5.6 Packages app

The web-mail-like application.

5.6.1 Send a Package

General syntax:

```
ascli aoc packages send [package extended value] [other parameters such as options and file list]
```

Package creation parameter are sent as <u>Command Parameter</u>. Refer to the AoC package creation API, or display an existing package in JSON to list attributes.

List allowed shared inbox destinations with:

```
ascli aoc packages shared_inboxes list
```

Use fields: recipients and/or bcc_recipients to provide the list of recipients: user or shared inbox:

- Provide either IDs as expected by API: "recipients":[{"type":"dropbox","id":"1234"}]
- or just names: "recipients":[{"The Dest"}].

ascli will resolve the list of email addresses and dropbox names to the expected type/ID list, based on case-insensitive partial match.

If a user recipient (email) is not already registered and the workspace allows external users, then the package is sent to an external user, and:

- if the option new_user_option is @json:{"package_contact":true} (default), then a public link is sent and the external user does not need to create an account
- if the option new_user_option is @json: {}, then external users are invited to join the workspace

5.6.1.1 Example: Send a package with one file to two users, using their email

```
ascli aoc packages send @json:'{"name":"my title","note":"my

output on the contempt of the co
```

5.6.1.2 Example: Send a package to a shared inbox with metadata

```
ascli aoc packages send --workspace="my ws" @json:'{"name":"my pack title","recipients":["Shared

→ Inbox With Meta"],"metadata":{"Project

→ Id":"123","Type":"Opt2","CheckThose":["Check1","Check2"],"Optional

→ Date":"2021-01-13T15:02:00.000Z"}}' ~/Documents/Samples/200KB.1
```

It is also possible to use identifiers and API parameters:

```
ascli aoc packages send --workspace="my ws" @json:'{"name":"my pack title","recipients":[{"type":]

dropbox","id":"12345"}],"metadata":[{"input_type":"single-text","name":"Project

Id","values":["123"]},{"input_type":"single-dropdown","name":"Type","values":["0pt2"]},

{"input_type":"multiple-checkbox","name":"CheckThose","values":["Check1","Check2"]},

{"input_type":"date","name":"Optional Date","values":["2021-01-13T15:02:00.000Z"]}]}'

~/Documents/Samples/200KB.1
```

5.6.1.3 Example: Send a package with files from the Files app

Find files in Files app:

```
ascli aoc files browse /src_folder
                         | recursive_size | size
                                                           | modified_time
                                                                                  | access_level |
 name
                | type
 sample_video | link
                                                           | 2020-11-29T22:49:09Z | edit
                                                            2021-04-21T18:19:25Z
 100G
                  file
                                             107374182400 |
                                                                                    edit
                  file
                                                            2021-05-18T08:22:39Z
 10M.dat
                                             10485760
                                                                                  l edit
 Test.pdf
                  file
                                             1265103
                                                            2022-06-16T12:49:55Z | edit
```

Let's send a package with the file 10M.dat from subfolder /src_folder in a package:

```
ascli aoc files node_info /src_folder --format=json --display=data | ascli aoc packages send

→ @json:'{"name":"test","recipients":["someuser@example.com"]}' 10M.dat --transfer=node

→ --transfer-info=@json:@stdin:
```

5.6.2 Receive packages

The command to receive one or multiple packages is:

```
ascli aoc packages recv <package id> [<file> ...]
```

Where <package id> is the identifier of the package to receive or ALL to receive all packages matching the query. Option once_only is supported, see below.

To download only some files from the package, just add the path of the files on the command line: [<file> ...], see option sources. By default, all files in the package are downloaded, i.e. • is used as the file list.

Option package_folder defines the attribute of folder used as destination sub folder in the to_folder path (see description earlier). The following syntax is supported

Syntax	Description
@none:	No subfolder is created, files are downloaded directly into the specified to_folder.
<field></field>	A subfolder named after the package's specified field is created inside to_folder .
<field1>+<field2></field2></field1>	A subfolder named after the combination of two package fields with a _ is created inside to_folder .
<field1>+<field2>?</field2></field1>	A subfolder named after the package's specified field1 is created, unless it already exists. Else it falls back to the combination of both fields with

The special value seq for <field> will append an incrementing number to the folder name starting at 1. If ? is used, then the sequence number is used only if the folder already exists.

Examples:

- id : Subfolder named after package ID. If the same package is downloaded several times, it will always be placed in the same folder.
- name: Subfolder named after package name. If two packages with the same name are downloaded, they will be combined in the same folder.
- name+id : Subfolder named after the combination of package name and ID.
- name+id? : Subfolder named after the package's name is created, unless it already exists. Else it falls back to the combination of both fields with . .
- name+seq? : Subfolder named after the package's name is created, unless it already exists. Else it falls back to the combination of name and sequence number.

Mote

When <field1>+<field2>? is used, if two packages are downloaded and have the same fields, they will be downloaded in the same folder. If name+seq? is used, if the same package is downloaded multiple times, it will be placed in different folders with a sequence number.

5.6.2.1 Example: Receive all packages from a given shared inbox

```
ascli aoc packages recv ALL --workspace=_workspace_ --once-only=yes --lock-port=12345
--query=@json:'{"dropbox_name":"_shared_inbox_name_","archived":false,"received":true,
--tase@json:'{"resume_policy":"sparse_csum","target_rate_kbps":50000}'
```

To list packages that would be downloaded, without actually downloading them, replace recv ALL with list (keep options once_only and query)

5.6.2.2 Receive new packages only (Cargo)

It is possible to automatically download new packages, like using Aspera Cargo:

ascli aoc packages recv ALL --once-only=yes --lock-port=12345

- ALL (case-sensitive) will download all packages
- --once-only=yes keeps memory of any downloaded package in persistency files located in the configuration folder
- --lock-port=12345 ensures that only one instance is started at the same time, to avoid running two downloads in parallel

Typically, one would execute this command on a regular basis, using the method of your choice: see Scheduler.

5.6.2.3 Example: Content of a received Package

Some node operations are available for a package, such as browse and find.

To list the content of a package, use command packages browse <package id> <folder>:

ascli aoc package browse my5CnbeWng /

Use command find to list recursively.

For advanced users, it's also possible to pipe node information for the package and use node operations:

```
ascli aoc package node_info <package ID here> / --format=json --show-secrets=yes --display=data |

access_key do self browse /
```

5.6.3 List packages

By default, when using aoc packages list or aoc packages receive ALL, the following query is performed:

Query parameter	Value
archived	false
has_content	true
received	true
completed	true
workspace_id	Set based on current workspace.
dropbox_id	Set according to dropbox_name, if provided.
exclude_dropbox_packages	true unless dropbox_id is provided.

Parameters provided using option query override this query. To remove a parameter, set it to null.

5.6.3.1 Example: List packages in a given shared inbox

When user packages are listed, the following query is used:

```
{"archived":<u>false</u>,"exclude_dropbox_packages":<u>true</u>,"has_content":<u>true</u>,"received":<u>true</u>}
```

To list packages in a shared inbox, the query has to be specified with the shared inbox by name or its identifier. Additional parameters can be specified, as supported by the API (to find out available filters, consult the API definition, or use the web interface in developer mode). The current workspace is added unless specified in the query.

Note

By default, exclude_dropbox_packages is set to true for user packages, and to false for shared inbox packages. This can be overridden in the query.

Using shared inbox name:

```
ascli aoc packages list --query=@json:'{"dropbox_name":"My Shared

Inbox","archived":false,"received":true,"has_content":true,"exclude_dropbox_packages":false,

"include_draft":false,"sort":"-received_at"}'
```

Using shared inbox identifier: first retrieve the ID of the shared inbox, and then list packages with the appropriate filter.

```
shared_box_id=$(ascli aoc packages shared_inboxes show --name='My Shared Inbox' --format=csv
--display=data --fields=id)

ascli aoc packages list
--query=@json:'{"dropbox_id":"'$shared_box_id'","archived":false,"received":true,"has_content":
--true,"exclude_dropbox_packages":false,"include_draft":false,"sort":"-received_at"}'
```

5.7 Files app

The Files application presents a <u>Home</u> folder to users in a given workspace. Files located here are either user's files, or shared folders.

• Note

All commands under files are the same as under access_keys do self for plugin node, i.e. gen4/access key operations.

5.7.1 Download Files

The general download command is:

```
ascli aoc files download <source folder path> <source filename 1> ...
```

I.e. the first argument is the source folder, and the following arguments are the source file names in this folder.

If a single file or folder is to be downloaded, then a single argument can be provided.

```
ascli aoc files download <single file path>
```

5.7.2 Shared folders

Like in AoC web UI, "Shared Folders" can be created and shared with either <u>Private</u> or <u>Public</u> links. <u>Private</u> links require the collaborator to log in to access the shared folder. <u>Public</u> links include a passcode that enables the user to access the shared folder without login-in.

Shared folders can be created either:

- by users in a workspace: they can share personal folders with other users in the same workspace: aoc files perm
- by administrators: they can share a folder with users in any workspace: aoc admin node do <node ID> perm

Technically (API), shared folder are managed through permissions on node and an event is sent to AoC to create a <u>link</u> in the user's home folder to the shared folder. In both cases, it is necessary to specify a workspace.

The basic payload to create a permission, i.e. a Shared Folder (last argument at creation usually specified with <code>@json:</code>) is:

```
"file_id": "50",
   "access_levels": ["list","read","write","delete","mkdir","rename","preview"],
   "access_type": "user",
   "access_id": "john@example.com",
   "tags": {...},
}
```

ascli expects the same payload for creation. ascli automatically populates some payload fields and provides convenient additional fields that generate native fields:

Field	Туре	Description
file_id	Native Auto	ID of the folder to share, as specified in the command line by path.
access_levels	Native Optional	List of access levels to set for the shared folder. Defaults to full access.
tags	Native Auto	Set with expected values for AoC: username who creates, and workspace in which the shared folder is created.
access_type	Native Required	Type of access, such as user, group, or workspace. Can be set with parameter with.
access_id	Native Required	ID of the user, group, or workspace (see with)
with	ascli	Recipient of shared folder. Can be a username, a group name, or a workspace name. ascli will resolve the name to the proper type and ID in fields access_type and access_id. If the value is the empty string, then it declares the shared folder in the workspace
		(first action to do, see below).
link_name	ascli	Name of the link file created in the user's home folder for private links.
as	ascli	Name of the link file created in the user's home folder for admin shared folders.

In order to declare/create the shared folder in the workspace, a special value for access_id is used: ASPERA_ACCESS_KEY_ADMIN_WS_[workspace ID]] . This is conveniently set by ascli using an empty string for field with . In order to share a folder with a different, special tags are set, but this is conveniently done by ascli using the as field.

5.7.2.1 User Shared Folders

Personal shared folders, created by users in a workspace follow the syntax:

ascli aoc files permission --workspace=<workspace name> <path to folder> ...



The workspace is identified by name, and folder by path, relative to the user's home. To use an identifier instead, one can use the percent selector, like %id:1234

5.7.2.2 Admin Shared Folders

Admin shared folders, created by administrators in a workspace follow the syntax:

ascli aoc admin node do <node ID> permission --workspace=<workspace name> <path to folder>

🥊 Tip

The node is identified by identifier. To use an name instead, one can use the percent selector, like %name: "my node"

5.7.2.3 Example: List permissions on a shared folder

ascli aoc files permission /shared_folder_test1 list

5.7.2.4 Example: Share a personal folder with other users

ascli aoc files permission /shared_folder_test1 create @json:'{"with":"laurent"}'

5.7.2.5 Example: Revoke shared access

ascli aoc files permission /shared folder test1 delete 6161

5.7.2.6 Example: Public and Private short links

They can be managed with commands:

```
ascli aoc files short_link <path to folder> private create
ascli aoc files short_link <path to folder> private list
ascli aoc files short_link <path to folder> public list
ascli aoc files short_link public delete <id>
ascli aoc files short_link public modify <id>@json:'{...}'
```

Only public short links can be modified. An optional payload can be provided at creation, for example to protect with a password, or set an expiry date. A password can be provided on create and modify for public links:

{"password": "my password here"}

To remove a password:

{"password_enabled":false}

Note

Access level cannot be customized in this version.

An expiration date can be set with parameter expires_at, using ISO 8601 format. E.g. 2025-08-29T08:10:31.000Z. If only a date is provided, it will be set to midnight UTC of that date.

5.7.2.7 Example: Create a workspace admin shared folder

First, identify the node ID where the shared folder will be created.

To get the node ID of the default node for workspace my ws, use the command:

ascli aoc admin workspace show %name:'my ws' --fields=node_id

Alternatively (longer):

ascli aoc admin workspace list --select=@json:'{"name":"my ws"}' --fields=node_id

Or select a node identifier manually from the list of nodes:

ascli aoc admin node list --fields=id,name

In the following commands, replace:

- 1234 with the node ID
- my ws with the workspace name
- /folder_on_node with the name of the folder on the node: it can also be a folder deeper than level 1.

The node can also be conveniently identified using the percent selector instead of numerical ID: <code>%name:"my node".</code>

If the shared folder does not exist, then create it:

```
ascli aoc admin node do 1234 mkdir /folder_on_node
```

Create the shared folder in workspace my ws (set with to empty string, or do not specify it). Optionally use as to set the name of the shared folder if different from the folder name on the node. For other options, refer to the previous section on shared folders.

```
ascli aoc admin node do 1234 permission /folder_on_node create

Graph of the description ```

#### 1 Note

The previous command only declares the shared folder in the workspace, but does not share it with anybody.

To share with a user, group, or workspace, use the with parameter with the name of a entity to share with (non-empty value). The "with" parameter will perform a lookup, and set fields access\_type and access\_id accordingly. The native fields access\_type and access\_id can also be used, instead of with.

```
ascli aoc admin node do 1234 permission /folder_on_node create

→ @json:'{"with":"john@example.com","as":"folder_for_one_user"}' --workspace="my ws"

ascli aoc admin node do 1234 permission /folder_on_node create @json:'{"with":"group

→ 1","as":"folder_for_a_group"}' --workspace="my ws"

ascli aoc admin node do 1234 permission /folder_on_node create @json:'{"with":"my

→ ws","as":"folder_for_all_workspace"}' --workspace="my ws"
```

#### Mote

In the previous commands, field as is optional.

#### 5.7.3 Cross Organization transfers

It is possible to transfer files directly between organizations without having to first download locally and then upload...

Although optional, the creation of Option Preset is recommended to avoid placing all parameters in the command line.

Procedure to send a file from org1 to org2:

- Get access to Organization 1 and create an Option Preset: e.g. org1 , for instance, use the Wizard
- Check that access works and locate the source file e.g. mysourcefile, e.g. using command files browse
- Get access to Organization 2 and create an Option Preset: e.g. org2
- Check that access works and locate the destination folder mydestfolder
- Execute the following:

ascli -Porg1 aoc files node\_info /mydestfolder --format=json --display=data | ascli -Porg2 aoc files upload mysourcefile --transfer=node --transfer-info=@json:@stdin:

Explanation:

- ascli is the command to execute by the shell
- -Porg1 load options for preset org1 (url and credentials)

- aoc use Aspera on Cloud plugin
- files node\_info /mydestfolder generate transfer information including Node API credential and root ID, suitable for the next command
- --format=json format the output in JSON (instead of default text table)
- --display=data display only the result, and remove other information, such as workspace name
- I the standard output of the first command is fed into the second one
- -Porg2 aoc use Aspera on Cloud plugin and load credentials for org2
- files upload mysourcefile upload the file named mysourcefile (located in org2) to org1
- --transfer=node use transfer agent type node instead of default direct
- --transfer-info=@json:@stdin: provide node transfer agent information, i.e. Node API credentials, those are expected in JSON format and read from standard input

#### 5.7.4 Find Files

The command aoc files find allows searching for files in a given workspace.

It works also on node resource using the v4 command:

```
ascli aoc admin node --name='my node name' --secret='my_secret_here' v4 find ...
```

For instructions, refer to section find for plugin node.

## 5.8 Tested commands for aoc

#### 1 Note

Add ascli aoc in front of the following commands:

```
admin analytics transfers nodes
admin analytics transfers organization
→ --query=@ison:'{"status":"completed","direction":"receive","limit":2}'
 --notify-to=my_email_external --notify-template=@ruby:'%Q{From: <%=from_name%>
 <<%=from email%>>\nTo: <<%=to%>>\nSubject: <%=ev["files completed"]%> files
received\n\n<%=ev.to_yaml%>}'
admin analytics transfers users --once-only=yes
admin application list
admin ats access_key create --cloud=aws --region=my_region --params=@json:'{"id":"ak_aws","name":"my
 test key AWS","storage":{"type":"aws_s3","bucket":"my_bucket","credentials":{"access_key_id":_|
"my_access_key", "secret_access_key": "my_secret_key" }, "path": "/" } }'
admin ats access_key create --cloud=softlayer --region=my_region
--params=@json:'{"id":"ak1ibmcloud", "secret": "my_secret_here", "name": "my test
 key","storage":{"type":"ibm-s3","bucket":"my_bucket","credentials":{"access_key_id":_|
 "my_access_key", "secret_access_key": "my_secret_key"}, "path": "/"}}'
admin ats access_key delete ak1ibmcloud
admin ats access_key list --fields=name,id
admin ats access_key node ak1ibmcloud --secret=my_secret_here browse /
admin ats cluster clouds
admin ats cluster list
admin ats cluster show --cloud=aws --region=eu-west-1
admin ats cluster show 1f412ae7-869a-445c-9c05-02ad16813be2
admin auth providers list
admin client list
admin client_access_key list
admin client_registration_token create @json:'{"data":{"name":"test_client_reg1",|
→ "client subject scopes":["alee", "aejd"], "client subject enabled":true}}'
admin client registration token delete client reg id
admin client_registration_token list
admin contact list
admin dropbox list
admin dropbox_membership list
admin group list
admin kms_profile list
admin node do %name:my_node_name --secret=my_ak_secret browse /
```

```
admin node do %name:my_node_name --secret=my_ak_secret browse /folder_sub --node-cache=no
admin node do %name:my_node_name --secret=my_ak_secret delete /folder1
admin node do %name:my_node_name --secret=my_ak_secret delete /folder_sub
admin node do %name:my_node_name --secret=my_ak_secret mkdir /folder1
admin node do %name:my_node_name --secret=my_ak_secret mkdir /folder_sub
admin node do %name:my_node_name --secret=my_ak_secret v3 access_key create

 @json:'{"id":"testsub1","storage":{"path":"/folder_sub"}}

admin node do %name:my_node_name --secret=my_ak_secret v3 access_key delete testsub1
admin node do %name:my_node_name --secret=my_ak_secret v3 events
admin node do %name:my_node_name delete test_shared_folder
admin node do %name:my_node_name mkdir test_shared_folder
admin node do %name:my_node_name perm test_shared_folder create

→ @json:'{"with":"", "as":"other_name_shared"}' --workspace=my_workspace_shared_inbox
admin node do %name:my_node_name perm test_shared_folder create
→ @json:'{"with":"my_user_email", "as":"other_name_shared"}' --workspace=my_workspace_shared_inbox
admin node do %name:my_node_name perm test_shared_folder create
Gison:'{"with":"my_user_group","as":"other_name_shared"}' --workspace=my_workspace_shared_inbox
admin node list
admin operation list
admin organization show
admin package list --http-options=@json:'{"read_timeout":120.0}'
admin saml_configuration list
admin self show
admin short_link list
admin subscription account
admin subscription usage
admin subscription usage MONTH
admin user list
admin user modify %name:my_user_email @json:'{"deactivated":false}'
admin workspace list
admin workspace_membership list
admin workspace_membership list --fields=ALL --query=@json:'{"page":1,"per_page":50,"embed":_|
 "member", "inherited":false, "workspace_id":11363, "sort": "name"}'
automation workflow action wf_id create @json:'{"name":"toto"}' \
automation workflow create @json:'{"name":"test_workflow"}'
automation workflow delete wf_id
automation workflow list
automation workflow list --query=@json:'{"show_org_workflows":"true"}' --scope=admin:all
automation workflow list --select=@json:'{"name":"test_workflow"}' --fields=id --format=csv
→ --display=data --output=test
bearer_token --display=data --scope=user:all
files bearer /
files bearer_token_node / --cache-tokens=no
files browse /
files browse / --url=my_private_link
files browse / --url=my_public_link_folder_no_pass
files browse / --url=my_public_link_folder_pass --password=my_public_link_password
files browse my_remote_file
files browse my_remote_folder
files browse my_remote_folder/
files cat testdst/test_file.bin
files delete /testsrc
files down --to-folder=. testdst/test_file.bin testdst/test_file.bin
files download --transfer=connect testdst/test_file.bin
files download --transfer=desktop testdst/test_file.bin
files find /
files find / '\.partial$'
files find / @ruby:'->(f){f["type"].eql?("file")}'
files mkdir /testsrc
files modify /some_folder @json:'{"mount_point":false}'
files modify my_test_folder
files permission my_test_folder list
files rename /some_folder testdst
files short_link /testdst private create
files short_link /testdst private list
files short_link /testdst public create
files show %id:aoc_file_id
```

```
files show /
files show testdst/test file.bin
files sync admin status /data/local_sync
files sync pull /testdst --to-folder=/data/local_sync
files thumbnail my_test_folder/video_file.mpg
files thumbnail my_test_folder/video_file.mpg --query=@json:'{"text":true,"double":true}'
files transfer push /testsrc --to-folder=/testdst test_file.bin
files upload --to-folder=/ test_file.bin --url=my_public_link_folder_no_pass
files upload --to-folder=/testsrc test_file.bin
files upload --to-folder=/testsrc test_file.bin test_file.bin
files upload --workspace=my_other_workspace --to-folder=my_other_folder test_file.bin --transfer=node
→ --transfer-info=@json:@stdin:
files v3 info
gateway --pid-file=pid_aocfxgw @json:'{"url":"https://localhost:12345/aspera/faspex"}' &
organization
organization --format=image --fields=background_image_url --ui=text
organization --url=my_public_link_recv_from_aoc_user
packages browse package_id3 /
packages list
packages list --query=@json:'{"dropbox_name":"my_shared_inbox_name","sort":"-received_at",|
 "archived":false, "received":true, "has_content":true, "exclude_dropbox_packages":false}'
packages receive ALL --once-only=yes --to-folder=. --lock-port=12345 packages receive ALL --once-only=yes --to-folder=. --lock-port=12345
 --query=@json:'{"dropbox_name":"my_shared_inbox_name","archived":false,"received":true,_|
 "has_content":true,"exclude_dropbox_packages":false,"include_draft":false}'
 --ts=@json:'{"resume_policy":"sparse_csum","target_rate_kbps":50000}'
packages receive INIT --once-only=yes --query=@json:'{"dropbox_name":"my_shared_inbox_name"}'
packages receive package_id3 --to-folder=.
packages receive package_id3 --to-folder=. / --package-folder=name
packages send --workspace=my_workspace_shared_inbox --validate-metadata=yes @json:'{"name":"$(notdir
 test) PACKAGE_TITLE_BASE", "recipients": ["my_shared_inbox_meta"], "metadata": [{"input_type": |
 "single-text", "name": "Project
 Id","values":["123"]},{"input_type":"single-dropdown","name":"Type","values":["Opt2"]},
 {"input_type":"multiple-checkbox","name":"CheckThose","values":["Check1","Check2"]},
 {"input_type":"date","name":"Optional Date","values":["2021-01-13T15:02:00.000Z"]}}}
 test_file.bin
packages send --workspace=my_workspace_shared_inbox --validate-metadata=yes @json:'{"name":"$(notdir
 test) PACKAGE_TITLE_BASE","recipients":["my_shared_inbox_meta"],"metadata":{"Project
 Id":"456","Type":"Opt2","CheckThose":["Check1","Check2"],"Optional
→ Date":"2021-01-13T15:02:00.000Z"}}' test_file.bin
packages send --workspace=my_workspace_shared_inbox --validate-metadata=yes @json:'{"name":"$(notdir
 test) PACKAGE_TITLE_BASE", "recipients": ["my_shared_inbox_meta"], "metadata": {"Type": "Opt2",
 "CheckThose":["Check1","Check2"],"Optional Date":"2021-01-13T15:02:00.000Z"}}' test_file.bin
packages send --workspace=my_workspace_shared_inbox @json:'{"name":"$(notdir test)
 PACKAGE_TITLE_BASE", "recipients": ["my_shared_inbox_name"]}' test_file.bin
packages send @json:'{"name":"$(notdir test) PACKAGE_TITLE_BASE","recipients":["my_email_external"]}'
→ --new-user-option=@json:'{"package_contact":true}' test_file.bin
packages send @json:'{"name":"$(notdir test)
→ PACKAGE_TITLE_BASE", "recipients": ["my_email_internal"], "note": "my_note"}' test_file.bin
packages send @json:'{"name":"$(notdir test) PACKAGE_TITLE_BASE"}' test_file.bin
--url=my_public_link_send_aoc_user --password=my_public_link_send_use_pass
packages send @json:'{"name":"$(notdir test) PACKAGE_TITLE_BASE"}' test_file.bin
→ --url=my_public_link_send_shared_inbox
packages shared_inboxes list
packages shared_inboxes show %name:my_shared_inbox_name
remind --username=my_user_email
servers
tier_restrictions
user contacts list
user pref modify @json:'{"default_language":"en-us"}'
user pref show
user profile modify @json:'{"name":"dummy change"}'
user profile show
user workspaces current
user workspaces list
```

# **Chapter 6**

# Plugin: ats: IBM Aspera Transfer Service

#### ATS is usable either:

- From an AoC subscription: ascli aoc admin ats : use AoC authentication
- Or from an IBM Cloud subscription: ascli ats: use IBM Cloud API key authentication

## 6.1 IBM Cloud ATS: Creation of API key

This section is about using ATS with an IBM cloud subscription. If you are using ATS as part of AoC, then authentication is through AoC, not IBM Cloud.

First get your IBM Cloud API key. For instance, it can be created using the IBM Cloud web interface, or using command line:

```
ibmcloud iam api-key-create mykeyname -d 'my sample key'

OK
API key mykeyname was created

Please preserve the API key! It cannot be retrieved after it's created.

Name mykeyname

Description my sample key

Created At 2019-09-30T12:17+0000

API Key my_secret_api_key_here

Locked false

UUID ApiKey-05b8fadf-e7fe-abcd-93a9-6fd348c5ab1f
```

#### References:

- IBM Cloud Managing user API keys
- IBM Aspera on Cloud regions

ascli ats api\_key create

Then, to register the key by default for the ats plugin, create a preset. Execute:

ascli config preset update my\_ibm\_ats --instance=aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeee

## 6.2 ATS Access key creation parameters

When creating an ATS access key, the option params must contain an extended value with the creation parameters. Those are directly the parameters expected by the ATS API.

# 6.3 Misc. Examples

Example: create access key on IBM Cloud (Softlayer):

Example: create access key on AWS:

Example: create access key on Azure SAS:

#### Note

The blob name is mandatory after server address and before parameters, and that parameter srec is mandatory.

Example: create access key on Azure:

delete all my access keys:

```
ascli ats access_key list --field=id --format=csv | ascli ats access_key delete @lines:@stdin:
```

The parameters provided to ATS for access key creation are the ones of ATS API for the POST /access\_keys endpoint.

## 6.4 Tested commands for ats

#### 1 Note

Add ascli ats in front of the following commands:

```
access_key create --cloud=softlayer --region=my_region
--params=@json:'{"id":"ak2ibmcloud","secret":"my_secret_here","name":"my test key","storage":{"type":"ibm-s3","bucket":"my_bucket","credentials":{"access_key_id":
"my_access_key", "secret_access_key": "my_secret_key"}, "path": "/"}}'
access_key delete ak2ibmcloud
access_key delete ak_aws
access_key entitlement ak2ibmcloud
access_key list --fields=name,id
access_key node ak2ibmcloud browse / --secret=my_secret_here
access_key show ak2ibmcloud
api_key create
api_key instances
api_key list
cluster clouds
cluster list
cluster show --cloud=aws --region=eu-west-1
cluster show 1f412ae7-869a-445c-9c05-02ad16813be2
```

# **Chapter 7**

# Plugin: server: IBM Aspera High Speed Transfer Server (SSH)

The server plugin is used for operations on Aspera HSTS using SSH authentication. It is the original way of accessing an Aspera Server, often used for server to server transfers. An SSH session is established, authenticated with either a password or an SSH private key, then commands ascp (for transfers) and ascmd (for file operations) are executed.

The URL to be provided with option url shall be like ssh://\_server\_address\_:33001, then option username is used to specify the transfer user, and finally either option password or ssh\_keys (with one or several paths) for the authentication.

Typically:

```
ascli server --url=ssh://hsts.example.com:33001 --username=john --password=_something_here_ ...
ascli server --url=ssh://hsts.example.com:33001 --username=john --ssh-keys=~/.ssh/id_rsa ...
```

#### 7.1 Tested commands for server

Note

Add ascli server in front of the following commands:

```
browse /
browse / --password=@none: --ssh-options=@json:'{"number_of_password_prompts":0}'
→ --ssh-keys=$aspera_key_path
browse my_inside_folder/test_file.bin
browse my_upload_folder/target_hot
cp my_inside_folder/test_file.bin my_upload_folder/200KB.2
delete my_inside_folder
delete my_upload_folder/to.delete
download my_inside_folder/test_file.bin --to-folder=.
→ --transfer-info=@json:'{"wss":false,"resume":{"iter_max":1}}'
download my_large_file --to-folder=my_upload_folder --transfer=node --ts.resume_policy=none
du /
health transfer --to-folder=my_upload_folder --format=nagios
md5sum my_inside_folder/test_file.bin
mkdir my_inside_folder --logger=stdout
mkdir my_upload_folder/target_hot
mv my_upload_folder/200KB.2 my_upload_folder/to.delete
sync admin status /data/local_sync
sync pull my_inside_folder --to-folder=/data/local_sync @json:'{"name":"serv_sync_pull_conf"}'
upload 'faux:///test1?100m' 'faux:///test2?100m' --to-folder=/Upload
--ts=@json:'{"target_rate_kbps":1000000,"resume_policy":"none","precalculate_job_size":true}'
```

```
upload 'faux:///test1?100m' 'faux:///test2?100m' --to-folder=/Upload
--ts=@json:'{"target_rate_kbps":1000000,"resume_policy":"none","precalculate_job_size":true}'
 --transfer-info=@json:'{"quiet":false}' --progress=no
"multi_session_threshold":1,"resume_policy":"none","target_rate_kbps":100000}'
--transfer-info=@json:'{"spawn_delay_sec":2.5,"multi_incr_udp":false}' --progress-bar=yes
upload --sources=@ts --transfer-info=@json:'{"ascp_args":["--file-list","filelist.txt"]}'

→ --to-folder=my_inside_folder

upload --sources=@ts --transfer-info=@json:'{"ascp args":["--file-pair-list","file pair list.txt"]}'
upload --sources=@ts
--ts=@json:'{"paths":[{"source":"test_file.bin","destination":"my_inside_folder/other_name_4"}]}
→ --transfer=transferd
upload --src-type=pair 'test_file.bin' my_inside_folder/other_name_2 --notify-to=my_email_external
→ --transfer-info=@json:'{"ascp_args":["-1","100m"]}'
upload --src-type=pair --sources=@json:'["test_file.bin","my_inside_folder/other_name_3"]'
→ --transfer-info.quiet=false --progress=no
upload --src-type=pair test_file.bin my_upload_folder/other_name_5 --ts=@json:'{"cipher":"aes-192-
 gcm","content_protection":"encrypt","content_protection_password":"my_secret_here","cookie":
 "biscuit","create_dir":true,"delete_before_transfer":false,"delete_source":false,
 "exclude_newer_than":"-1","exclude_older_than":"-10000","fasp_port":33001,"http_fallback":false,
 "multi_session":0,"overwrite":"diff+older","precalculate_job_size":true,"preserve_access_time":_|
 true, "preserve_creation_time":true, "rate_policy": "fair", "resume_policy": "sparse_csum", |
 "symlink_policy":"follow"}'
upload --to-folder=my_upload_folder/target_hot --lock-port=12345
 --transfer-info=@json:'{"ascp_args":["--remove-after-transfer","--remove-empty-directories","--
 exclude-newer-than=-8","--src-base","source_hot"]}' source_hot
```

#### 7.2 Authentication on Server with SSH session

If SSH is the session protocol (by default i.e. not WSS), then following session authentication methods are supported:

- password : SSH password
- ssh keys : SSH keys (Multiple SSH key paths can be provided.)

If username is not provided then the default transfer user xfer is used.

If neither SSH password nor key is provided and a transfer token is provided in transfer spec (option ts), then standard SSH bypass key(s) is used.

Example:

```
ascli server --url=ssh://_server_address_:33001 ... --ts=@json:'{"token":"Basic _token_here_"}'
```

#### 1 Note

If you need to use the Aspera public keys, then specify an empty token: --ts=@json:'{"token":""}' : Aspera public SSH keys will be used, but the protocol will ignore the empty token.

The value of the  $ssh_{keys}$  option can be a single value or an Array. Each value is a path to a private key and is expanded ( $\sim$  is replaced with the user's home folder).

Examples:

```
ascli server --ssh-keys=~/.ssh/id_rsa
ascli server --ssh-keys=@list:,~/.ssh/id_rsa
ascli server --ssh-keys=@json:'["~/.ssh/id_rsa"]'
```

For file operation command (browse, delete), the Ruby SSH client library Net::SSH is used and provides several options settable using option ssh\_options (additive option like ts).

For a list of SSH client options, refer to the Ruby documentation of Net::SSH.

Among the 50 available SSH options:

- verbose
- use\_agent
- passphrase

By default, the SSH library will check if a local ssh-agent is running.

On Linux, if you get an error message such as:

ERROR -- net.ssh.authentication.agent: could not connect to ssh-agent: Agent not configured or on Windows:

ERROR -- net.ssh.authentication.agent: could not connect to ssh-agent: pageant process not running

This means that your environment suggests using an agent, but you don't have such an SSH agent running, then:

- Check env var: SSH AGENT SOCK
- Check your file: \$HOME/.ssh/config
- Check if the SSH key is protected with a passphrase (then, use the passphrase SSH option)
- · Check the Ruby SSH options in start method
- To disable the use of ssh-agent, use the option ssh\_options like this:

ascli server --ssh-options=@json:'{"use agent": false}' ...



This can also be set using a preset.

If one of the SSH private keys is passphrase-protected, then option passphrase can be used. It is equivalent to setting both options ssh\_options.passphrase and ts.ssh\_private\_key\_passphrase.

#### 7.3 Other session channels for server

URL schemes local and https are also supported (mainly for testing purpose). ( --url=local: , --url=https://...)

- local will execute ascmd locally, instead of using an SSH connection.
- https will use Web Socket Session: This requires the use of a transfer token. For example a Basic token can be used.

As, most of the time, SSH is used, if a http scheme is provided without token, the plugin will fallback to SSH and port 33001.

# 7.4 Examples: server

One can test the server application using the well known demo server:

```
ascli config initdemo
ascli server browse /aspera-test-dir-large
ascli server download /aspera-test-dir-large/200MB
```

initdemo creates an Option Preset demoserver and set it as default for plugin server.

If an SSH private key is used for authentication with a passphrase, the passphrase needs to be provided to both options: ssh\_options (for browsing) and ts (for transfers):

```
ascli server --url=ssh://_server_address_here_:33001 --username=_user_here_

--ssh_keys=_private_key_path_here_ --passphrase=_passphrase_here_
```

# Plugin: **node**: IBM Aspera High Speed Transfer Server Node

This plugin gives access to capabilities provided by the HSTS Node API.

The authentication is username and password or access\_key and secret through options: username and password.

#### Note

Capabilities of this plugin are used in other plugins which access to the Node API, such as aoc, ats, shares.

#### • Note

This plugin can be used with any type of <u>Aspera Node</u>, either on premise or ATS, provided that you have Node API credentials. Those credentials can be either Node API user or Access Key (e.g. on ATS).

## 8.1 File Operations

It is possible to do gen3/node user operations:

- browse
- Transfer (upload / download / sync)
- delete
- ...

When using an access key, so called gen4/access key API is also supported through sub commands using access\_keys do self.

#### Example:

- ascli node browse / : list files with gen3/node user API
- ascli node access\_key do self browse / : list files with gen4/access key API

#### **8.1.1** Browse

Native API parameters can be placed in option query.

Special parameters can be placed in option |query | for "gen3" browse:

| Parameter | Description                      |
|-----------|----------------------------------|
| recursive | Recursively list files.          |
| max       | Maximum number of files to list. |
| self      | Offset in the list.              |

Option node\_cache can be set to no to avoid use of folder cache (Redis) and force actual read of file system.

## 8.2 Operation find on gen4/access key

The command find <folder> [filter\_expr] is available for gen4/access key, under access\_keys do self

The argument <folder> is mandatory and is the root from which search is performed. The argument [filter\_expr] is optional and represent the matching criteria.

It recursively scans storage to find files/folders matching criteria and then returns a list of matching entries.

[filter expr] is either:

- Optional (default): All files and folder are selected
- Type String: The expression is similar to shell globing, refer to Ruby function: File.fnmatch
- Type Proc: The expression is a Ruby lambda that takes one argument: a Hash that contains the current folder entry to test. Refer to the following examples.

Examples of expressions:

Find all files and folders under /

ascli node access\_keys do self find

• Find all text files /Documents

ascli node access\_keys do self find /Documents '\*.txt'

The following are examples of Ruby lambda code to be provided in the following template command:

ascli node access\_keys do self find / @ruby:'->(f){[code here]}'

#### Tip

Single quotes are used here above to protect the whole <u>Ruby</u> expression from the shell. Then double quotes are used for strings in the <u>Ruby</u> expression to not mix with the shell.

• Find files more recent than 100 days

```
->(f){f["type"].eql?("file") and (DateTime.now-DateTime.parse(f["modified_time"]))<100}
```

Find files older than 1 year

```
->(f){f["type"].eql?("file") and (DateTime.now-DateTime.parse(f["modified_time"]))>365}
```

• Find files larger than 1 MB

```
->(f){f["type"].eql?("file")} and f["size"].to_i>1000000}
```

Filter out files beginning with \_\_\_ or named .DS\_Store :

```
->(f){!(f["name"].start_with?("._") or f["name"].eql?(".DS_Store"))}
```

Match files using a Ruby Regex: \.gif\$

```
->(f){f["name"].match?(/\.gif$/)}
```

ascli commands can be piped in order to combine operations, such as find and delete:

```
ascli node access_keys do self find / @ruby:'->(f){f["type"].eql?("file") and

OuteTime.now-DateTime.parse(f["modified_time"]))>365}' --fields=path --format=csv | ascli node

``

1 Note

The pipe | character on the last line is used to chain commands.

8.3 Listing transfer events

When a transfer is run, its information is stored (typically, 1 day) in the HSTS database (Redis). This information can be retrieved with command: transfer list.

If the number of transfers is too large, then the list will be retrieved using several API calls.

In addition, it is possible to list "only new information" using option once_only.

```
ascli node transfer list --once-only=yes
```

The iteration_token that keeps memory of the latest event is stored in the persistence repository of ascli. To reset it, add option: --query=@json:'{"reset": true}'. To list only a number of events, use the max parameter in query. Other parameters are directly transmitted to the underlying API (GET /ops/transfers).

8.4 Central

The central sub-command uses the <u>reliable query</u> API (session and file). It allows listing transfer sessions and transferred files.

Filtering can be applied:

ascli node central file list

By providing the validator option, offline transfer validation can be done.

• Note

See later in this doc, refer to HSTS doc.

8.5 Sync

There are three commands related to file synchronisation:

Com-		shares		server	
mand	node		aoc		Description
sync	Yes	Yes	Yes	Yes	Perform a local sync, by executing async locally.
async	Yes				Uses API /async .
					Get status on sync operation on server side, like Aspera Console.
ssync	Yes				Uses API /asyncs .
					It can start a sync operation on the server side, and monitor only those.

For details on the sync action, refer to IBM Aspera Sync.

async subcommands: show and delete accept special identifier ALL.

8.6 FASP Stream

It is possible to start a faspstream session using the Node API:

Use the command ascli node stream create --ts=@json:<value>, with transfer-spec:

```
{"direction": "send", "source": "udp://233.3.3.4:3000?loopback=1&ttl=2", "destination": "udp://233.3.3.3: ]

3001/", "remote_host": "localhost", "remote_user": "stream", "remote_password": "my_pass_here"}
```

8.7 Watchfolder

Refer to Aspera Server documentation, or Aspera Watchfolder API Documentation for watch folder creation.

ascli supports remote operations through the Node API. Operations are:

- Start watchd and watchfolderd services running as a system user having access to files
- · Configure a Watchfolder to define automated transfers

```
ascli node service create @json:'{"id":"mywatchd","type":"WATCHD","run_as":{"user":"user1"}}'
ascli node service create

Gison:'{"id":"mywatchfolderd","type":"WATCHFOLDERD","run_as":{"user":"user1"}}'
ascli node watch_folder create @json:'{"id":"mywfolder","source_dir":"/watch1","target_dir":"/",
Head of the content of
```

8.8 Out of Transfer File Validation

Follow the Aspera Transfer Server configuration to activate this feature.

The following command lists one file that requires validation, and assign it to the unique validator identifier provided:

To update the status of the file, use the following command:

8.9 Example: SHOD to ATS

Scenario: Access to a <u>Shares on Demand</u> (SHOD) server on AWS is provided by a partner. We need to transfer files from this third party SHOD instance into our Azure BLOB storage. Simply create an <u>Aspera Transfer Service</u> instance, which provides access to the Node API. Then create a configuration for the <u>SHOD</u> instance in the configuration file: in section <u>shares</u>, a configuration named: <u>aws_shod</u>. Create another configuration for the Azure ATS instance: in section <u>node</u>, named <u>azure_ats</u>. Then execute the following command:

```
ascli node download /share/sourcefile --to-folder=/destination_folder --preset=aws_shod
--transfer=node --transfer-info=@preset:azure_ats
```

This will get transfer information from the SHOD instance and tell the Azure ATS instance to download files.

8.10 Node file information

When Node API is used with an Access key, extra information can be retrieved, such as preview.

• Note

Display of preview on terminal requires installation of extra gem: rmagick

```
dnf install -y ImageMagick-devel
gem install rmagick rainbow
```

For example, it is possible to display the preview of a file, if it exists, using an access key on node:

ascli node access_key do self thumbnail /preview_samples/Aspera.mpg

Previews are mainly used in AoC, this also works with AoC:

ascli aoc files thumbnail /preview_samples/Aspera.mpg

🕊 Tip

To specify the file by its file ID, use the selector syntax: "id:_file_id_here_

1 Note

To force textual display of the preview on iTerm, prefix command with: env -u TERM_PROGRAM -u LC_TERMINAL

8.11 Create access key

9 Tip

The id and secret fields are optional. If not provided, they will be generated and returned into the result. In that case, provide option --show-secrets=yes to get the generated secret.

Access keys support extra overriding parameters using parameter: configuration and sub keys transfer and server. For example, an access key can be modified or created with the following options:

```
{"configuration":{"transfer":{"target_rate_cap_kbps":500000}}}
```

The list of supported options can be displayed using command:

ascli node info --field=@ruby:'/^access_key_configuration_capabilities.*/'

8.12 Generate and use bearer token

Bearer tokens are part of the <u>gen4/access key</u> API. It follows the model of OAuth 2. For example, they are used in Aspera on Cloud. This is also available for developers for any application integrating Aspera. In this API, files, users and groups are identified by an ID (a String, e.g. "125", not necessarily numerical).

Bearer tokens are typically generated by the authenticating application and then recognized by the Node API. A bearer token is authorized on the node by creating permissions on a folder.

Bearer tokens can be generated using ascli command bearer_token: it takes two arguments:

- The private key used to sign the token.
- The token information, which is a Hash containing the following elements:

Parameter	Default	Туре	Description
_scope _validity user_id	user:all 86400 - node. <access_key>:<_scope</access_key>	Special Special Mandatory	Either user:all or admin:all Validity in seconds from now. Identifier of user API scope
scope	node. Vaccess_key/. V_scope	Manuatory	e.g. node. <access_key>:<node scope=""></node></access_key>
expires_at	now+<_validity>	Mandatory	Format: %Y-%m-%dT%H:%M:%SZ e.g. 2021-12-31T23:59:59Z
auth_type	access_key	Optional	access_key , node_user
group_ids	-	Optional	List of group IDs
organization_id	-	Optional	Organization ID
watermarking_json_ba	-	Optional	Watermarking information (not used)

Note

For convenience, ascli provides additional parameters _scope and _validity . They are not part of the API and are removed from the final payload. They are used respectively to easily set a value for scope and expires_at

8.12.1 Bearer token: Environment

An access key shall be created to grant access for transfers to its storage. The access key and its secret represent administrative access to the storage as it has access rights to the whole storage of the access key.

They way to create access keys depend slightly on the type of HSTS:

- If a self-managed Aspera node is used, then a <u>node user admin</u> must be created: It has no <u>docroot</u> but has at least one file restriction (for testing, one can use ★ to accept creation of an access key with any storage root path). Refer to the Aspera HSTS documentation.
- If Cloud Pak for integration is used, then the node admin is created automatically.
- If Aspera on Cloud or ATS is used, then the SaaS API for access key creation is used.

Note

Refer to HSTS manual: Access key authentication section for more details on access key creation.

In the next sections, we will assume that an access key has been created and that ascli is configured to use this access key by default using node.

8.12.2 Bearer token: Preparation

Let's assume that the access key was created, and a default configuration is set to use this <u>access key</u>. Using <u>ascli</u>, an access key can be created using the <u>access_key</u> create on the node (using main node credentials) or ATS.

Create a private key (organization key) that will be used to sign bearer tokens:

my_private_pem=./myorgkey.pem
ascli config genkey \$my_private_pem

• Note

This private key is not used for authentication. It is used to sign bearer tokens. Refer to section private key for more details on generation.

The corresponding public key shall be placed as an attribute of the <u>access key</u> (done with PUT /access_keys/<id>

ascli node access_key set_bearer_key self @file:\$my_private_pem

1 Note

Either the public or private key can be provided, and only the public key is used. This will enable to check the signature of the bearer token. Above command is executed with access key credentials.

Alternatively, use the following equivalent command, as ascli kindly extracts the public key with extension .pub:

ascli node access_key modify %id:self @ruby:'{token_verification_key:
 File.read("'\$my_private_pem'.pub")}'

8.12.3 Bearer token: Configuration for user

• Select a folder for which we want to grant access to a user, and get its identifier:

my_folder_id=\$(ascli node access_key do self show / --fields=id)

Note

Here we simply select /, but any folder can be selected in the access key storage.

Let's designate a user by its ID:

my_user_id=777

Note

This is an arbitrary identifier, typically managed by the web application. Not related to Linux user IDs or anything else.

Grant this user access to the selected folder:

```
ascli node access_key do self permission %id:$my_folder_id create

→ @json:'{"access_type":"user","access_id":"'$my_user_id'"}'
```

· Create a Bearer token for the user:

```
ascli node bearer_token @file:./myorgkey.pem

→ @json:'{"user_id":"'$my_user_id'","_validity":3600}' --output=bearer.txt
```

1 Note

The Bearer token can also be created using command as node admin on HSTS. Refer to the HSTS manual: Bearer tokens section. Code for token generation is provided in lib/aspera/api/node.rb

8.12.4 Bearer token: User side

Now, let's assume we are the user, the only information received are:

- · The URL of the Node API
- A Bearer token
- A file ID for which we have access

Let's use it:

```
ascli node -N --url=https://... --password="Bearer $(cat bearer.txt)" --root-id=$my_folder_id

→ access_key do self br /
```

8.13 Tested commands for node

Note

Add ascli node in front of the following commands:

```
--url=https://tst.example.com/path --password="Bearer bearer_666" --root-id=root_id access_key do
⇔ self br /
access_key create
→ @json:'{"id":"my_username","secret":"my_password_here","storage":{"type":"local","path":"/"}}'
access_key delete my_username
access_key do my_ak_name browse /
access_key do my_ak_name delete /test_nd_ak2
access_key do my_ak_name delete test_nd_ak3
access_key do my_ak_name download test_nd_ak3 --to-folder=.
access_key do my_ak_name find my_test_folder
access_key do my_ak_name find my_test_folder @re:'\.jpg$'
access_key do my_ak_name find my_test_folder @ruby:'->(f){f["name"].end_with?(".jpg")}'
access_key do my_ak_name mkdir /tst_nd_ak
access_key do my_ak_name mkfile /mkfile.txt "hello world"
access_key do my_ak_name mklink /mklink.txt
   --query=@json:'{"target":"/mkfile.txt","target_node_id":"123"}'
access_key do my_ak_name node_info /
access_key do my_ak_name rename /tst_nd_ak test_nd_ak2
access_key do my_ak_name show %id:1
access_key do my_ak_name show /test_nd_ak3
access_key do my_ak_name upload 'faux:///test_nd_ak3?100k' --default-ports=no
access_key do self permission %id:root_id create @json:'{"access_type":"user","access_id":"666"}'
access_key do self permission / show 1
access_key do self show / --fields=id --output=root_id
access key list
access_key set_bearer_key self @file:my_private_key
```

```
access_key show %id:self
api details
asperabrowser
async bandwidth %name:SYNC_NAME
async counters %name:SYNC NAME
async delete ALL
async files %name:SYNC_NAME
async files %name:SYNC_NAME --once-only=yes
async list
async show %name:SYNC_NAME
async show ALL
basic_token
bearer_token @file:my_private_key @json:'{"user_id":"666"}' --output=bearer_666
browse / --log-level=trace2
cat my_upload_folder/test_file.bin
central file list
central file modify --validator=1 @json:'{"files":[]}'
central session list
delete @list:,my_upload_folder/a_folder,my_upload_folder/tdlink,my_upload_folder/a_file
delete my_upload_folder/test_file.bin
download my_upload_folder/test_file.bin --to-folder=.
info --fpac='function FindProxyForURL(url,host){return "DIRECT"}'
license
mkdir my_upload_folder/a_folder
mkfile my_upload_folder/a_file1 "hello world"
mklink my_upload_folder/a_folder my_upload_folder/tdlink
rename my_upload_folder a_file1 a_file
search / --query=@json:'{"sort":"mtime"}'
service create @json:'{"id":"service1","type":"WATCHD","run_as":{"user":"user1"}}'
service delete service1
service list
slash
space /
ssync bandwidth %name:my_node_sync
ssync counters %name:my_node_sync
ssync create @json:'{"configuration":{"name":"my_node_sync","local":{"path":"my_local_path_real"},<sub>|</sub>
   "remote":{"host":"my_host","port":my_port,"user":"my_username","pass":"my_password_here","path<sup>†</sup>:
   "my_remote_path"}}}'
ssync delete %name:my node sync
ssync files %name:my_node_sync
ssync list
ssync show %name:my_node_sync
ssync start %name:my_node_sync
ssync state %name:my_node_sync
ssync stop %name:my_node_sync
ssync summary %name:my_node_sync
stream list
sync admin status /data/local_sync
sync pull /aspera-test-dir-tiny --to-folder=/data/local_sync @json:'{"reset":true}'
transfer list --once-only=yes
transfer list --query=@json:'{"active_only":true}'
transfer list --query=@json:'{"reset":true}' --once-only=yes
transfer sessions
transport
upload --to-folder=my_upload_folder --sources=@ts
   --ts=@json:'{"paths":[{"source":"/aspera-test-dir-small/10MB.2"}],"precalculate_job_size":true}'
   --transfer=node --transfer-info=@json:'{"url":"https://node.example.com/path@","username":
   "my_username","password":"my_password_here"}'
upload --username=my_ak_name --password=my_ak_secret test_file.bin
upload test_file.bin --to-folder=my_upload_folder --ts=@json:'{"target_rate_cap_kbps":10000}'
watch_folder list
```

8.14 Open Telemetry

The node plugin supports Open Telemetry (OTel) for monitoring and tracing.

Note

This is an experimental feature and currently only available for the node plugin and Instana backend.

ascli polls the Node API for transfer events and sends them to an OTel collector.

The command expects the following parameters provided as a Hash positional parameter:

Parameter	Type	Default	Description
url key interval	String String Float	- - 10	URL of the Instana HTTPS backend for OTel. Agent key for the backend. Polling interval in seconds.
			o for single shot.

To retrieve OTel backend information: Go to the Instana web interface, $\underline{\mathsf{More}} \to \underline{\mathsf{Agents}} \to \underline{\mathsf{Docker}}$ and identify the agent endpoint and key, e.g. endpoint=ingress-blue-saas.instana.io . Identify the region and the endpoint URL will be https://otlp-[region]-saas.instana.io , i.e. replace ingress with otlp .

For convenience, those parameters can be provided in a preset, e.g. named otel_default.

Then it is invoked like this (assuming a default node is configured):

ascli node telemetry @preset:otel_default

In Instana, create a custom Dashboard to visualize the OTel data:

- · Add Widget: Histogram
- · Data Source: Infrastructure and Platforms
- Metric: search transfer

Plugin: faspex5: IBM Aspera Faspex v5

IBM Aspera's newer self-managed application.

3 authentication methods are supported (option auth):

Method	Description
jwt web	General purpose, private-key based authentication
web	Requires authentication with web browser
<pre>public_link</pre>	Public link authentication (set when option url is a public link)
boot	Use authentication token copied from browser (experimental)



If you have a Faspex 5 public link, provide it, as-is, through the option url.

9.1 Faspex 5 quick start with wizard

For a quick start, one can use the wizard, which will help to create an Option Preset:

ascli config wizard

Then, answer questions interactively:

argument: url> faspex5.example.com

Potentially, multiple applications may be detected, or if only Faspex is detected, it would skip this step:

When Faspex is detected, it would ask for the path to a private key. If you don't have a private key, then leave that field blank, and it will generate one or use one that was previously generated.

```
Using: Faspex at https://faspex5.example.com/aspera/faspex
Please provide the path to your private RSA key, or nothing to generate one:
option: key_path>
Using existing key:
/Users/someuser/.aspera/ascli/my_key
```

Then, the email of the user shall be provided:

option: username> someuser@example.com

The administrator of Faspex shall provide you with a client id, as specified below.

```
Ask the ascli client ID and secret to your Administrator.
Admin should login to: https://faspex5.example.com/aspera/faspex
Navigate to: :: → Admin → Configurations → API clients
Create an API client with:
 name: ascli
 JWT: enabled
Then, logged in as someuser@example.com go to your profile:
() → Account Settings → Preferences -> Public Key in PEM:
----BEGIN PUBLIC KEY----
redacted
----END PUBLIC KEY----
Once set, fill in the parameters:
option: client_id> _my_key_here_
option: client_secret> ****
Preparing preset: faspex5_example_com_user
Setting config preset as default for faspex5
You can test with:
ascli faspex5 user profile show
Saving configuration file.
```

• Note

Paste the entire public key, including the BEGIN and END lines, into the user's profile.

For more information on the JWT method, refer to the section below.

If you have generated a private key with the wizard and lost the public key, you can retrieve the public key like this:

```
ascli faspex5 --show-config --show-secrets=yes --fields=private_key | ascli conf pubkey @stdin:
```

9.2 Faspex 5 JWT authentication

This is the general purpose and recommended method to use.

Activation is in two steps:

The administrator must create an API client in Faspex with JWT support

This operation is generally done only once:

- As Admin, Navigate to the web UI: Admin → Configurations → API Clients → Create
- Give a name, like ascli
- Activate JWT
- There is an option to set a global public key allowing the owner of the private key to impersonate any user. Unless
 you want to do this, leave this field empty.
- Click on Create Button
- Take note of Client ID (and Client Secret, but not used in current version)
- The user will authenticate with a private key and set the public key in his Faspex 5 profile.

This operation is done by each user using the CLI.

- As a user, click the user logo located to the left of the app switcher in the top-right corner.
- Select Account Settings.
- At the bottom, in the Public key in PEM format field, paste the <u>public key</u> that corresponds to the private key assigned to your account.

🕊 Tip

If you don't have a private key, see Private Key to generate one.

Then use these options:

```
--auth=jwt
--client-id=_client_id_here_
--client-secret=my_secret_here
--username=_username_here_
--private-key=@file:.../path/to/key.pem
```

• Note

Use the private_key option to provide the PEM content (not the file path). To load from a file, prefix the path with @file:, e.g. @file:/path/to/key.pem.

Typically, users create a preset so they don't have to enter these options each time.

Example:

```
ascli config preset update myf5 --auth=jwt --client-id=_client_id_here_

--client-secret=my_secret_here --username=_username_here_ --private-key=@file:.../path/to/key.pem
ascli config preset set default faspx5 myf5
ascli faspex5 user profile show
```

9.3 Faspex 5 web authentication

For web-based authentication, the administrator must create an API client in Faspex for an external web app support:

- As Admin, Navigate to the web UI: Admin → Configurations → API Clients → Create
- · Do not Activate JWT
- Set Redirect URI to https://127.0.0.1:8888
- Click on Create Button
- Take note of the Client Id (and Client Secret, but not used in current version)

The user will use the following options:

```
--auth=web
--client-id=_client_id_here_
--client-secret=my_secret_here
--redirect-uri=https://127.0.0.1:8888
```

9.4 Faspex 5 public link authentication

If all you have is a public link received by email or other, you can still do authorized actins with it.

For example, for a public link to post a package:

ascli faspex5 packages send --url='https://faspex5.example.com/?context=_some_long_string_here_'

9.5 Faspex 5 bootstrap authentication

For boot method: (will be removed in future)

- As user: Open a Web Browser
- · Start developer mode
- Login to Faspex 5
- Find the first API call with Authorization header, and copy the value of the token (series of base64 values with dots)

Use this token as password and use --auth=boot.

```
ascli config preset update f5boot --url=https://localhost/aspera/faspex --auth=boot
-- --password=_token_here_
```

9.6 Tested commands for faspex5

• Note

Add ascli faspex5 in front of the following commands:

```
admin accounts list
admin alternate addresses list
admin clean_deleted
admin configuration modify @json:'{"mfa_required":false}'
admin configuration show
admin contacts list
admin distribution_lists create @json:'{"name":"test4","contacts":[{"name":"john@example.com"}]}'
admin distribution_lists delete %name:test4
admin distribution_lists list --query=@json:'{"type":"global"}'
admin email notifications list
admin email_notifications show welcome_email
admin event app --query=@json:'{"max":20}
admin event web
admin jobs list --query=@json:'{"job_type":"email","status":"failed"}' --fields=id,error_desc
admin metadata_profiles list
admin node browse %name:Local
admin node list
admin node shared_folders %name:Local list
admin node shared_folders %name:Local show %name:Main
admin node shared folders %name:Local user %name:Main list
admin node show %name:Local
admin oauth_clients list
admin registrations list
admin saml_configs list
admin shared_inboxes invite %name:my_shared_box_name johnny@example.com
admin shared_inboxes list
admin shared_inboxes list --query=@json:'{"all":true}'
admin shared_inboxes members %name:my_shared_box_name create %name:john@example.com
admin shared_inboxes members %name:my_shared_box_name delete %name:john@example.com
admin shared_inboxes members %name:my_shared_box_name delete %name:johnny@example.com
admin shared_inboxes members %name:my_shared_box_name list
admin smtp create @json:'{"auth_type":"open","server_address":"smtp.gmail.com","server_port":587, |
→ "domain": "gmail.com", "tls_enabled": true, "packages_recipient_from": "sender"}'
admin smtp modify @json:'{"default_time_zone_offset":"0"}'
admin smtp show
admin smtp test my_email_external
admin workgroups list
bearer_token
gateway --pid-file=pid_f5_fxgw @json:'{"url":"https://localhost:12346/aspera/faspex"}' &
health --url=https://faspex5.example.com/path
invitation list
invitations create @json:'{"email_address":"aspera.user1+u@gmail.com"}'
packages browse f5_pack_id --query=@json:'{"recursive":true}'
packages delete f5_pack_id
packages list --box=ALL
packages list --box=my_shared_box_name
packages list --box=my_workgroup --group-type=workgroups
packages list --box=outbox --fields=DEF,sender.email,recipients.0.recipient_type
packages list --query=@json:'{"mailbox":"inbox","status":"completed"}
packages receive --box=my_shared_box_name package_box_id1 --to-folder=.
packages receive --box=my_workgroup --group-type=workgroups workgroup_package_id1 --to-folder=.
packages receive ALL --once-only=yes --to-folder=.
packages receive INIT --once-only=yes
packages receive f5_pack_id --to-folder=.
--ts=@json:'{"content_protection_password":"my_secret_here"}'
packages send --shared-folder=%name:my_shared_folder_name @json:'{"title":"test

→ title", "recipients": ["my_email_internal"] | my_shared_folder_file --fields=id --display=data
  --output=f5_pack_id
packages send --url=my_public_link_send_f5_user @json:'{"title":"test title"}' test_file.bin
```

```
packages send --url=my_public_link_send_shared_box @json:'{"title":"test title"}' test_file.bin
packages send @json:'{"title":"test
→ title", "recipients": ["my_shared_box_name"], "metadata": {"Options": "Opt1", "TextInput": "example

    text"}}' test file.bin

packages send @json:'{"title":"test title","recipients":["my_workgroup"]}' test_file.bin
packages send @json:'{"title":"test title","recipients":[{"name":"my_username"}]my_meta}'
+ test_file.bin --ts=@json:'{"content_protection_password":"my_secret_here"}
packages show --box=my_shared_box_name package_box_id1
packages show --box=my_workgroup --group-type=workgroups workgroup_package_id1
packages show f5_pack_id
packages status f5_pack_id
postprocessing --pid-file=pid_f5_postproc @json:'{"url":"https://localhost:8553/asclihook", |

    "script_folder":"","cert":"localhost.p12","key":"changeit"}' &
shared browse %name:my_src
shared list
shared_folders browse %name:my_shared_folder_name
shared_folders list
user account
user account --query.expand=true
user profile modify @json:'{"preference":{"connect_disabled":false}}'
user profile show
version
```

Most commands correspond directly to REST API calls. Parameters to commands are carried through option query, as extended value, for list, or through Command Parameter for creation. One can conveniently use the JSON format with prefix <code>@json:</code>.



The API is listed in Faspex 5 API Reference under IBM Aspera Faspex API.

9.7 Faspex 5: Inbox selection

By default, package operations (send , receive , list) are performed on the user's inbox.

To select another inbox, use option box with one of the following values:

box	Comment
inbox	Default
<pre>inbox_history</pre>	
inbox_all	
<pre>inbox_all_history</pre>	
outbox	
<pre>outbox_history</pre>	
pending	
pending_history	
all	
ALL	All inboxes of all users. <u>admin only</u> .
Open value	If group_type is shared_inboxes : name of a shared inbox (default) If group_type is workgroups : workgroup name

Note

In case the name of the box is an open value, use option <code>group_type</code> set to either <code>shared_inboxes</code> or <code>workgroups</code> .

9.8 Faspex 5: Send a package

A package can be sent with the command:

```
ascli faspex5 packages send [extended value: Hash with package info] [files...]
```

The Hash creation <u>Command Parameter</u> provided to command corresponds to the Faspex 5 API: POST /packages (refer to the API reference for a full list of parameters, or look at request in browser).

Required fields are title and recipients.

Example (assuming a default preset is created for the connection information):

Longer example for the payload of @json::

```
{"title":"some title","recipients":[{"recipient_type":"user","name":"user@example.com"}]}
```

recipient_type is one of (Refer to API):

- user
- workgroup
- external_user
- distribution_list
- shared_inbox

ascli adds some convenience: The API expects the field recipients to be an Array of Hash, each with field name and optionally recipient_type. ascli also accepts an Array of String, with simply a recipient name. Then, ascli will look up existing contacts among all possible types, use it if a single match is found, and set the name and recipient_type accordingly. Else an exception is sent.

On the state of
The lookup is case-insensitive and on partial matches.

```
{"title":"some title","recipients":["user@example.com"]}
```

If the lookup needs to be only on certain types, you can specify the field: recipient_types with either a single value or an Array of values (from the list above). e.g.:

```
{"title":"test

→ title","recipient_types":"user","recipients":["user1@example.com","user2@example.com"]}
```

9.9 Faspex 5: Send a package with metadata

It's the same as sending a package, but with an extra field metadata in the package info.

```
{"title":"test title","recipients":["my shared inbox"],"metadata":{"Confidential":"Yes","Drop

→ menu":"Option 1"}}
```

Basically, add the field metadata, with one key per metadata and the value is directly the metadata value. (Refer to API documentation for more details).

9.10 Faspex 5: List packages

Option box can be used to list packages from a specific box (see Inbox Selection above).

Option query can be used to filter the list of packages, based on native API parameters, directly sent to Faspex 5 API GET /packages .

Parameter	Type	Description
offset	Native	Managed by ascli: Offset of first package. Default: 0
limit	Native	Managed by ascli: # of packages per API call. Default: 100
q	Native	General search string case-insensitive, matches if value is contained in several fields
	Native	Other native parameters are supported (Refer to API documentation)
max	Special	Maximum number of items to retrieve (stop pages when the maximum is passed)

Parameter	Туре	Description
pmax	Special	Maximum number of pages to request (stop pages when the maximum is passed)

A <u>Command Parameter</u> in last position, of type <u>Proc</u>, can be used to filter the list of packages. This advantage of this method is that the expression can be any test, even complex, as it is Ruby code. But the disadvantage is that the filtering is done in <u>ascli</u> and not in Faspex 5, so it is less efficient.

Examples:

• List only available packages: (filtering is done in Faspex)

```
ascli faspex5 packages list --query=@json:'{"status":"completed"}'
```

• Similar, using filtering in ascli:

```
ascli faspex5 packages list @ruby:'->(p){p["state"].eql?("released")}'
```

9.11 Faspex 5: Browsing folder content

Several entities support folder browsing: Packages, Nodes, Shared Folders. All support two modes: paging and legacy API. By default, paging is used.

Option query is available with parameters supported by the API and ascli:

Parameter	Evaluation	Default	Description
paging	ascli	true	Use paging API.
recursive	ascli	false	List inside folders.
max	ascli	-	Maximum number of items.
filter	API	{"basenames":[]}	Refer to API doc.
offset	API (legacy)	Θ	Index of first item.
limit	API (legacy)	500	Number of items in one API call result.
per_page	API (paging)	500	Number of items in one API call result.

9.12 Faspex 5: Content of a received Package

• Note

Listing content also applies to sent packages using --box=outbox.

To list the content of a received package, use command faspex5 packages browse <package id> . Optionally, provide a folder path.

9.13 Faspex 5: Receive a package

To receive one, or several packages at once, use command faspex5 packages receive. Provide either a single package ID, or an extended value Array of package IDs, e.g. @list:,1,2,3 as argument.

The same options as for faspex5 packages list can be used to select the box and filter the packages to download. I.e. options box and query, as well as last Command Parameter Proc (filter).

Option --once-only=yes can be used, for "cargo-like" behavior. Special package ID INIT initializes the persistency of already received packages when option --once-only=yes is used.

Special package ID ALL selects all packages (of the selected box). In this case, typically, only completed packages should be downloaded, so use option --query=@json:'{"status":"completed"}'.

If a package is password protected, then the content protection password is asked interactively. To keep the content encrypted, use option: --ts=@json:'{"content_protection":null}', or provide the password instead of null.

Tip: If you use option query and/or positional filter, you can use the list command for a dry run.

9.14 Faspex 5: List all shared inboxes and work groups

If you are a regular user, to list work groups you belong to:

```
ascli faspex5 admin workgroup list
```

If you are admin or manager, add option: --query=@json:'{"all":true}', this will list items you manage, even if you do not belong to them. Example:

```
ascli faspex5 admin shared list --query=@json:'{"all":true}' --fields=id,name
```

Shared inbox members can also be listed, added, removed, and external users can be invited to a shared inbox.

ascli faspex5 admin shared_inboxes invite '%name:the shared inbox' john@example.com

It is equivalent to:

```
ascli faspex5 admin shared_inboxes invite '%name:the shared inbox'

Gison:'{"email_address":"john@example.com"}'
```

Other payload parameters are possible for invite in this last Hash Command Parameter:

```
{"description":"blah", "prevent_http_upload":<u>true</u>, "custom_link_expiration_policy":<u>false</u>, |

"invitation_expires_after_upload":<u>false</u>, "set_invitation_link_expiration":<u>false</u>, |

"invitation_expiration_days":3}
```

9.15 Faspex 5: Create Metadata profile

```
ascli faspex5 admin metadata_profiles create @json:'{"name":"the
    profile","default":false,"title":{"max_length":200,"illegal_chars":[]},"note":{"max_length":400,
    "illegal_chars":[],"enabled":false},"fields":[{"ordering":0,"name":"field1","type":"text_area",
    "require":true,"illegal_chars":[],"max_length":100},{"ordering":1,"name":"fff2","type":
    "option_list","require":false,"choices":["opt1","opt2"]}]}'
```

9.16 Faspex 5: Create a Shared inbox with specific metadata profile

ascli faspex5 admin shared create @json:'{"name":"the shared inbox","metadata_profile_id":1}'

9.17 Faspex 5: List content in Shared folder and send package from remote source

ascli faspex5 shared_folders list --fields=id,name

ascli faspex5 shared_folders br %name:'Server Files' /folder

Tip

The shared folder can be identified by its numerical id or by name using percent selector: %<field>:<value> . e.g. --shared-folder=3

9.18 Faspex 5: Receive all packages (cargo)

To receive all packages, only once, through persistency of already received packages:

ascli faspex5 packages receive ALL --once-only=yes --query=@json:'{"status":"completed"}'

To initialize, and skip all current package so that next time ALL is used, only newer packages are downloaded:

ascli faspex5 packages receive INIT --once-only=yes

9.19 Faspex 5: Invitations

There are two types of invitations of package submission: public or private.

Public invitations are for external users, provide just the email address.

ascli faspex5 invitations create @json:'{"email_address":"john@example.com"}' --fields=access_url

Private invitations are for internal users, provide the user or shared inbox identifier through field recipient_name.

9.20 Faspex 5: Cleanup packages

Warning

Operation requires admin level.

The default automated cleanup period can be displayed with:

ascli faspex5 admin configuration show --fields=days_before_deleting_package_records

This parameter can be modified with:

ascli faspex5 admin configuration modify @json:'{"days_before_deleting_package_records":30}'

To start package purge, i.e. permanently remove packages marked for deletion older than days_before_deleting_package_, use command:

ascli faspex5 admin clean_deleted

Note

The expiration period taken by default is the one from admin configuration show. To use a different period than the default, specify it on command line with: <code>@json:'{"days_before_deleting_package_records":15}'</code>

To delete all packages, one can use the following command:

ascli faspex5 packages list --box=ALL --format=yaml --fields=id | ascli faspex5 packages delete

→ @yaml:@stdin:

• Note

Above command will mark all packages for deletion, and will be permanently removed after the configured period (clean_deleted command). It is possible to add a filter to the list command to only delete packages matching some criteria, e.g. using --select=@ruby:'->(p){...}' on packages list.

9.21 Faspex 5: Admin: Unlock user

To unlock a user, you can deactivate and then re-activate the user:

ascli faspex5 admin accounts modify %name:some.user@example.com @json:'{"account_activated":false}'

ascli faspex5 admin accounts modify %name:some.user@example.com @json:'{"account_activated":true}'



Here we use the convenient percent selector, but the numerical ID can be used as well.

To send a password reset link to a user, use command reset password on the account.

9.22 Faspex 5: Faspex 4-style post-processing

The command ascli faspex5 postprocessing emulates Faspex 4 post-processing script execution in Faspex 5. It implements a web hook for Faspex 5 and calls a script with the same environment variables as set by Faspex 4. Environment variables at set to the values provided by the web hook which are the same as Faspex 4 post-processing.

It allows to quickly migrate workflows from Faspex 4 to Faspex 5 while preserving scripts. Nevertheless, on long term, a native approach shall be considered, such as using Aspera Orchestrator or other workflow engine, using Faspex 5 native web hooks or File Processing.

It is invoked like this:

ascli faspex5 postprocessing

An optional positional parameter can be provided as extended value Hash:

Parameter	Type	Default	Description
server info	-	-	See Web service.
script_folder	String		Prefix added to script path (Default: CWD)
fail_on_error	Bool	false	Fail if true and process exits with non-zero code
<pre>timeout_seconds</pre>	Integer	60	Time out before script is killed

When a request on ascli is received the following happens:

- ascli gets the path of the URL called
- It removes the base path of base URL.
- It prepends it with the value of script_folder
- It executes the script at that path
- Upon success, a success code is returned

For example:

```
ascli faspex5 postprocessing

Graph of the processing of the proce
```

In Faspex 5, the URL of the webhook endpoint shall be reachable from within Faspex containers. For example, if ascli in running in the base host, the URL hostname shall not be localhost, as this refers to the local address inside Faspex container. Instead, one can specify the IP address of the host or host.containers.internal (Check podman manual).

Let's define the web hook:

Webhook endpoint URI: http://host.containers.internal:8080/processing/script1.sh

Then the post-processing script executed will be /opt/scripts/script1.sh.

9.23 Faspex 5: Faspex 4 Gateway

1 Note

This is not a feature for production. It's provided for testing only.

For legacy Faspex client applications that use the send API (only) of Faspex v4, the command gateway provides the capability to present an API compatible with Faspex 4, and it will call the Faspex 5 API.

It takes a single argument which is the URL at which the gateway will be located (locally):

ascli faspex5 gateway @json:'{"url":"https://localhost:12345/aspera/faspex"}'

There are many limitations:

- It's only to emulate the Faspex 4 send API (send package).
- No support for remote sources, only for an actual file transfer by the client.
- The client must use the transfer spec returned by the API (not faspe: URL).
- Tags returned in transfer spec must be used in transfer.
- Only a single authentication is possible (per gateway) on Faspex5.
- No authentication of F4 side (ignored).

Behavior: The API client calls the Faspex 4 API on the gateway, then the gateway transforms this into a Faspex5 API call, which returns a transfer spec, which is returned to the calling client. The calling client uses this to start a transfer to HSTS which is actually managed by Faspex 5.

For other parameters, see Web service.

9.24 Faspex 5: Get Bearer token to use API

If a command is missing, then it is still possible to execute command by calling directly the API on the command line using curl:

curl -H "Authorization: \$(ascli ascli bearer)"

https://faspex5.example.com/aspera/faspex/api/v5/api_endpoint_here

Plugin: faspex: IBM Aspera Faspex v4

Warning

Faspex v4 is end of support since September 30th, 2024. So this plugin for Faspex v4 is deprecated. If you still need to use Faspex4, then use ascli version 4.19.0 or earlier.

1 Note

For full details on Faspex API, refer to: Reference on Developer Site

This plugin uses APIs versions 3 Faspex v4. The v4 command requires the use of API v4, refer to the Faspex Admin manual on how to activate.

10.1 Listing Packages

Command: faspex package list

10.1.1 Option box

By default, it looks in box inbox, but the following boxes are also supported: archive and sent, selected with option box.

10.1.2 Option recipient

A user can receive a package because the recipient is:

- The user himself (default)
- The user is member of a dropbox/workgroup: filter using option recipient set with value *<name of dropbox/workgroup

10.1.3 Option query

As inboxes may be large, it is possible to use the following query parameters:

Parameter	Evaluation	Description
count	API	Number of items in one API call result (default=0, equivalent to 10)
page	API	ID of page in call (default=0)
startIndex	API	Index of item to start (default=0)
max	ascli	Maximum number of items
pmax	ascli	Maximum number of pages

(SQL query is LIMIT <startIndex>, <count>)

The API is listed in Faspex 4 API Reference under Services (API v.3).

If no parameter max or pmax is provided, then all packages will be listed in the inbox, which result in paged API calls (using parameter: count and page). By default, count is 0 (10), it can be increased to issue less HTTP calls.

10.1.4 Example: List packages in dropbox

```
ascli faspex package list --box=inbox --recipient='*my_dropbox'

--query=@json:'{"max":20,"pmax":2,"count":20}'
```

List a maximum of 20 items grouped by pages of 20, with maximum 2 pages in received box (inbox) when received in dropbox *my_dropbox .

10.2 Receiving a Package

The command is package recv, possible methods are:

- Provide a package ID with option id
- Provide a public link with option link
- Provide a faspe: URI with option link

```
ascli faspex package recv 12345
ascli faspex package recv --link=faspe://...
```

If the package is in a specific dropbox/workgroup, add option recipient for both the list and recv commands.

```
ascli faspex package list --recipient='*dropbox_name'
ascli faspex package recv 125 --recipient='*dropbox_name'
```

If id is set to ALL, then all packages are downloaded, and if option once_only is used, then a persistency file is created to keep track of already downloaded packages.

10.3 Sending a Package

The command is faspex package send. Package information (title, note, metadata, options) is provided in option delivery_info. The content of delivery_info is directly the contents of the send v3 API of Faspex 4.

Example:

If the recipient is a dropbox or workgroup: provide the name of the dropbox or workgroup preceded with ★ in the recipients field of the delivery_info option: "recipients":["*MyDropboxName"]

Additional optional parameters in mandatory option delivery info:

- Package Note:: "note": "note this and that"
- Package Metadata: "metadata": {"Meta1": "Val1", "Meta2": "Val2"}

It is possible to send from a remote source using option remote_source , providing either the numerical ID, or the name of the remote source using percent selector: %name:<name>.

Remote source can be browsed if option storage is provided. storage is a Hash extended value. The key is the storage name, as listed in source list command. The value is a Hash with the following keys:

- node is a Hash with keys: url, username, password
- path is the sub-path inside the node, as configured in Faspex

10.4 Email notification on transfer

Like for any transfer, a notification can be sent by email using options: notify_to and notify_template. Example:

```
ascli faspex package send --delivery-info=@json:'{"title":"test pkg

→ 1","recipients":["aspera.user1@gmail.com"]}' ~/Documents/Samples/200KB.1

→ --notify-to=aspera.user1@gmail.com --notify-template=@ruby:'%Q{From: <%=from_name%>

→ <<%=from_email%>>\nTo: <<%=to%>>\nSubject: Package sent:

→ <%=ts["tags"]["aspera"]["faspex"]["metadata"]["_pkg_name"]%> files received\n\nTo user:

→ <%=ts["tags"]["aspera"]["faspex"]["recipients"].first["email"]%>}'
```

In this example the notification template is directly provided on command line. Package information placed in the message are directly taken from the tags in transfer spec. The template can be placed in a file using modifier: <code>@file:</code>

10.5 Operations on dropbox

Example:

```
ascli faspex v4 dropbox create @json:'{"dropbox":{"e_wg_name":"test1","e_wg_desc":"test1"}}'
ascli faspex v4 dropbox list
ascli faspex v4 dropbox delete 36
```

10.6 Remote sources

Faspex lacks an API to list the contents of a remote source (available in web UI). To work around this, the Node API is used, for this it is required to set option: storage that links a storage name to a node configuration and sub path.

Example:

In this example, a Faspex storage named my_storage exists in Faspex, and is located under the docroot in /mydir (this must be the same as configured in Faspex). The node configuration name is my_faspex_node here.

1 Note

The v4 API provides an API for nodes and shares.

10.7 Automated package download (cargo)

It is possible to tell ascli to download newly received packages, much like the official cargo client, or drive. Refer to the same section in the Aspera on Cloud plugin:

ascli faspex packages recv ALL --once-only=yes --lock-port=12345

10.8 Tested commands for faspex

• Note

Add ascli faspex in front of the following commands:

```
address_book
dropbox list --recipient="*my_dbx"
health
```

```
login_methods
package list --box=sent --query.max=1 --fields=package_id --display=data --format=csv
→ --output=f4 prs2
package list --query.max=1 --fields=package id --display=data --format=csv --output=f4 prs1
package list --query.max=5
package list --recipient="*my_dbx" --format=csv --fields=package_id --query.max=1 --output=f4_db_id1 package list --recipient="*my_wkg" --format=csv --fields=package_id --query.max=1 --output=f4_db_id2
package receive --to-folder=. --link=https://app.example.com/recv_from_user_path
package receive ALL --once-only=yes --to-folder=. --query=@json:'{"max":10}
package receive f4_db_id1 --recipient="*my_dbx" --to-folder=.
package receive f4_db_id2 --recipient="*my_wkg" --to-folder=.
package receive f4_pri1 --to-folder=.
package receive f4_prs2 --to-folder=. --box=sent
package send --delivery-info=@json:'{"title":"$(notdir test)
→ PACKAGE_TITLE_BASE", "recipients": ["*my_dbx"]}' test_file.bin
package send --delivery-info=@json:'{"title":"$(notdir test)
→ PACKAGE_TITLE_BASE", "recipients": ["*my_wkg"]}' test_file.bin
package send --delivery-info=@json:'{"title":"$(notdir test)
PACKAGE_TITLE_BASE", "recipients": ["my_email_internal", "my_username"] } ' test_file.bin
package send --delivery-info=@json:'{"title":"$(notdir test)
→ PACKAGE_TITLE_BASE", "recipients": ["my_email_internal"]}' --remote-source=%name:my_src
\rightarrow sample_source.txt
package send --link=https://app.example.com/send_to_dropbox_path
→ --delivery-info=@json:'{"title":"$(notdir test) PACKAGE_TITLE_BASE"}' test_file.bin
package send --link=https://app.example.com/send_to_user_path
→ --delivery-info=@json:'{"title":"$(notdir test) PACKAGE_TITLE_BASE"}' test_file.bin
source info %name:my_src --storage=@preset:faspex4_storage
source list
source node %name:my_src br / --storage=@preset:faspex4_storage
v4 dmembership list
v4 dropbox list
v4 metadata_profile list
v4 user list
v4 wmembership list
v4 workgroup list
```

Plugin: shares: IBM Aspera Shares v1

Aspera Shares supports the Node API for the file transfer part.

Supported commands are listed in Share's API documentation:

https://developer.ibm.com/apis/catalog/aspera--aspera-shares-api/Introduction

The payload for creation is the same as for the API, parameters are provided as positional Hash.

Example: Create a Node: Attributes are like API:

Attribute	Required	Default
name	Yes	
host	Yes	
api_username	Yes	
api_password	Yes	
port		9092
ssl		true
verify_ssl		false
timeout		30s
open_timeout		10s

Example: Create a share and add a user to it.

```
ascli shares admin share create

Gison:'{"node_id":1,"name":"test1","directory":"test1","create_directory":true}'

share_id=$(ascli shares admin share list --select=@json:'{"name":"test1"}' --fields=id)

user_id=$(ascli shares admin user all list --select=@json:'{"username":"username1"}' --fields=id)

ascli shares admin share user_permissions $share_id create

Gjson:'{"user_id":'$user_id',"browse_permission":true, "download_permission":true,

"mkdir_permission":true,"delete_permission":true,"rename_permission":true,

"content_availability_permission":true,"manage_permission":true}'
```

11.1 Tested commands for shares

• Note

Add ascli shares in front of the following commands:

```
admin group all list
admin node list
admin share list --fields=DEF,-status,status_message
admin share user permissions 1 list
```

```
admin user all app_authorizations 1 modify @json:'{"app_login":true}'
admin user all app_authorizations 1 show
admin user all list
admin user all share_permissions 1 list
admin user all share permissions 1 show 1
admin user ldap add the name
admin user local list
admin user saml import @json:'{"id":"the_id","name_id":"the_name"}'
files browse /
files delete my_share_folder/new_folder
files delete my_share_folder/test_file.bin
files download --to-folder=. my_share_folder/test_file.bin
files download --to-folder=. my_share_folder/test_file.bin my_share_folder/test_file.bin
--transfer=httpgw --transfer-info=@json:'{"url":"https://tst.example.com/path@"}'
files mkdir my_share_folder/new_folder
files sync push /data/local_sync --to-folder=my_share_folder/synctst
files sync push /data/local_sync --to-folder=my_share_folder/synctst @json:'{"reset":true}'
files upload --to-folder=my_share_folder 'faux:///testfile?1m' --transfer=httpgw
   --transfer-info=@json:'{"url":"https://tst.example.com/path@","synchronous":true,"api_version":
"v1","upload_chunk_size":100000}'
files upload --to-folder=my_share_folder sendfolder --transfer=httpgw
   --transfer-info=@json:'{"url":"https://tst.example.com/path@","synchronous":true,"api_version":

    "v1", "upload_chunk_size":100000}'
files upload --to-folder=my_share_folder test_file.bin
files upload --to-folder=my_share_folder test_file.bin --transfer=httpgw
   --transfer-info=@json:'{"url":"https://tst.example.com/path@"}'
health
```

Plugin: console: IBM Aspera Console

12.1 Transfer filter

Listing transfers supports the API syntax.

In addition, it is possible to place a single <code>query</code> parameter in the request to filter the results: <code>filter</code>, following the syntax:

(field operator value) and (field operator value) ...

12.2 Tested commands for console

• Note

Add ascli console in front of the following commands:

```
health
transfer current files console_xfer_id
transfer current list --query.filter='(transfer_name contain aoc)'
transfer current list --query=@json:'{"filter1":"transfer_name","comp1":"contain","val1":"aoc"}'
transfer current show console_xfer_id
transfer smart list
transfer smart sub my_smart_id

Gipon:'{"source":{"paths":["my_smart_file"]},"source_type":"user_selected"}'
```

workflow list

workflow status ALL

workflow status my_workflow_id

Plugin: orchestrator :IBM Aspera Orchestrator

13.1 Tested commands for orchestrator

workflow start my_workflow_id @json:'{"Param":"world !"}'

```
Add ascli orchestrator in front of the following commands:

health
info
plugins
processes
workflow details my_workflow_id
workflow export my_workflow_id
workflow inputs my_workflow_id
```

workflow start my_workflow_id @json:'{"Param":"world !"}' --result=ResultStep:Complete_status_message

Plugin: cos: IBM Cloud Object Storage

The IBM Cloud Object Storage provides the possibility to execute transfers using FASP. It uses the same transfer service as Aspera on Cloud, called Aspera Transfer Service (ATS). Available ATS regions: https://status.aspera.io

There are two possibilities to provide credentials. If you already have the endpoint, API key and Resource Instance ID (CRN), use the first method. If you don't have credentials but have access to the IBM Cloud console, then use the second method.

14.1 Using endpoint, API key and Resource Instance ID (CRN)

If you have those parameters already, then following options shall be provided:

Option	Description
bucket	Bucket name
endpoint	Storage endpoint URL
	e.g. https://s3.hkg02.cloud-object-storage.appdomain.cloud
apikey	API Key
crn	Resource instance ID

For example, let us create a default configuration:

```
ascli config preset update mycos --bucket=mybucket

--endpoint=https://s3.us-east.cloud-object-storage.appdomain.cloud --apikey=abcdefgh
--crn=crn:v1:bluemix:public:iam-identity::a/xxxxxxx
ascli config preset set default cos mycos
```

Then, jump to the transfer example.

14.2 Using service credential file

If you are the COS administrator and don't have yet the credential: Service credentials are directly created using the IBM cloud Console (web UI). Navigate to:

- → Navigation Menu
- → Resource List
- → Storage
- → Select your storage instance
- → Service Credentials
- → New credentials (Leave default role: Writer, no special options)
- → Copy to clipboard

Then save the copied value to a file, e.g.: \$HOME/cos_service_creds.json or using the IBM Cloud CLI:

(if you don't have jq installed, extract the structure as follows)

It consists in the following structure:

```
"apikey": "my_api_key_here",
  "cos_hmac_keys": {
    "access_key_id": "my_access_key_here",
    "secret_access_key": "my_secret_here"
},
  "endpoints": "https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints",
  "iam_apikey_description": "my_description_here",
  "iam_apikey_name": "my_key_name_here",
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Writer",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/xxxxxxxx....",
  "resource_instance_id": "crn:v1:bluemix:public:cloud-object-storage:global:a/xxxxxxxx...."
}
```

The field resource_instance_id is for option crn

The field apikey is for option apikey

1 Note

Endpoints for regions can be found by querying the endpoints URL from file or from the IBM Cloud Console.

The required options for this method are:

Option	Description
bucket	Bucket name
region	Bucket region
	e.g. eu-de
service_credentials	JSON information saved from IBM Cloud console.

For example, let us create a default configuration:

```
ascli config preset update mycos --bucket=laurent

---service-credentials=@val:@json:@file:~/service_creds.json --region=us-south
ascli config preset set default cos mycos
```

14.3 Operations, transfers

Let's assume you created a default configuration from one of the two previous steps (else specify the access options on command lines).

A subset of **node** plugin operations are supported, basically Node API:

```
ascli cos node info
ascli cos node upload 'faux:///sample1G?1g'
```

Note

A dummy file sample1G of size 2 GB is generated using the faux PVCL scheme (see previous section and man ascp), but you can, of course, send a real file by specifying a real file path instead.

14.4 Tested commands for cos



Add ascli cos in front of the following commands:

```
node download test_file.bin --to-folder=.
node info --bucket=my_bucket --endpoint=my_endpoint --apikey=my_api_key --crn=my_resource_instance_id
node info --bucket=my_bucket --region=my_region --service-credentials=@json:@file:my_cos_svc_cred
node info --log-level=trace2
node upload test_file.bin
```

Plugin: httpgw: HTTP Gateway

15.1 Tested commands for httpgw



Plugin: faspio: Faspio Gateway

16.1 Tested commands for faspio

Plugin: alee : Aspera License Entitlement Engine

Retrieve information on subscription.

17.1 Tested commands for alee



entitlement health -N

Plugin: preview: Preview generator for AoC

The preview generates thumbnails (office, images, video) and video previews on storage for use primarily in the Aspera on Cloud application. It uses the <u>Node API</u> of Aspera HSTS and requires use of Access Keys and its <u>storage root</u>. Several options can be used to tune several aspects:

- Methods for detection of new files needing generation
- Methods for generation of video preview
- · Parameters for video handling

See also https://github.com/IBM/aspera-on-cloud-file-previews

18.1 Aspera Server configuration

Specify the preview's folder as shown in:

https://ibmaspera.com/help/admin/organization/installing_the_preview_maker

By default, the preview plugin expects previews to be generated in a folder named previews located in the storage root. On the transfer server execute:

PATH=/opt/aspera/bin:\$PATH

asconfigurator -x "server;preview_dir,previews"
asnodeadmin --reload

Note

The configuration <code>preview_dir</code> is <code>relative</code> to the storage root, no need leading or trailing <code>/</code> . In general just set the value to <code>previews</code>

If another folder is configured on the HSTS, then specify it to ascli using the option previews_folder.

The HSTS Node API limits any preview file to a parameter: max_request_file_create_size_kb (1 KB is 1024 Bytes). This size is internally capped to 1<<24 Bytes (16777216), i.e. 16384 KB, i.e. 16 MB.

To change this parameter in aspera.conf, use asconfigurator. To display the value, use asuserdata:

asuserdata -a | grep max_request_file_create_size_kb

max_request_file_create_size_kb: "1024"
asconfigurator -x "server; max_request_file_create_size_kb,16384"

If you use a value different from 16777216, then specify it using option max_size.

Note

The HSTS parameter (max_request_file_create_size_kb) is in kilo Bytes while the generator parameter is in Bytes (factor of 1024).

18.2 External tools: Linux

ascli requires the following external tools available in the PATH:

- ImageMagick v7+: magick (for tools: convert and composite)
- OptiPNG: optipng
- FFmpeg: ffmpeg ffprobe
- LibreOffice: unoconv

Here shown on Red Hat/Rocky Linux.

Other OSes should work as well, but are not tested.

To check if all tools are found properly, execute:

ascli preview check

18.2.1 Image: ImageMagick and optipng

```
dnf install -y ImageMagick optipng
```

You may also install ghostscript which adds fonts to ImageMagick. Available fonts, used to generate PNG for text, can be listed with magick identify -list font. Prefer ImageMagick version >=7.

More info on ImageMagick at https://imagemagick.org/

If your OS has only ImageMagick v6, then you can create a script called magick and add it to your PATH:

```
#!/bin/bash
exec "$@"
```

make it executable:

chmod a+x /usr/local/bin/magick

18.2.2 Video: FFmpeg

The easiest method is to download and install the latest released version of fimpeg with static libraries from https://johnvansickle.com/ffmpeg/

```
curl -s https://johnvansickle.com/ffmpeg/releases/ffmpeg-release-amd64-static.tar.xz|(mkdir -p /opt && cd /opt && rm -f ffmpeg /usr/bin/{ffmpeg,ffprobe} && rm -fr ffmpeg-*-amd64-static && tar xJvf --- && ln -s ffmpeg-* ffmpeg && ln -s /opt/ffmpeg/{ffmpeg,ffprobe} /usr/bin)
```

18.2.3 Office: unoconv and LibreOffice

If you don't want to have preview for office documents or if it is too complex you can skip office document preview generation by using option: --skip-types=office

The generation of preview in based on the use of LibreOffice's unoconv.

RHEL 8/Rocky Linux 8+

dnf install unoconv

· Amazon Linux

```
amazon-linux-extras enable libreoffice
yum clean metadata
yum install libreoffice-core libreoffice-calc libreoffice-opensymbol-fonts libreoffice-ure

→ libreoffice-writer libreoffice-pyuno libreoffice-impress
wget https://raw.githubusercontent.com/unoconv/unoconv/master/unoconv
mv unoconv /usr/bin
chmod a+x /usr/bin/unoconv
```

18.3 Configuration

The preview generator should be executed as a non-user. When using object storage, any user can be used, but when using local storage it is usually better to use the user xfer, as uploaded files are under this identity: this ensures proper access rights. (we will assume this)

Like any ascli commands, options can be passed on command line or using a configuration Option Preset. The configuration file must be created with the same user used to run so that it is properly used on runtime.

The xfer user has a special protected shell: aspshell, so in order to update the configuration, and when changing identity, specify an alternate shell. E.g.:

```
su -s /bin/bash - xfer

ascli config preset update mypreviewconf --url=https://localhost:9092 --username=my_access_key

--password=my_secret --skip-types=office --lock-port=12346

ascli config preset set default preview mypreviewconf
```

Here we assume that Office file generation is disabled, else remove this option. lock_port prevents concurrent execution of generation when using a scheduler.

One can check if the access key is well configured using:

```
ascli -Ppreviewconf node browse /
```

This shall list the contents of the storage root of the access key.

18.4 Options for generated files

When generating preview files, some options are provided by default. Some values for the options can be modified on command line. For video preview, the whole set of options can be overridden with option reencode_ffmpeg: it is a Hash with two keys: in and out, each is an Array of strings with the native options to ffmpeg.

18.5 Execution

ascli intentionally supports only a <u>one shot</u> mode (no infinite loop) in order to avoid having a hanging process or using too many resources (calling REST API too quickly during the scan or event method). It needs to be run on a regular basis to create or update preview files. For that use your best reliable scheduler, see <u>Scheduler</u>.

Typically, for <u>Access key</u> access, the system/transfer user is xfer. So, in order to be consistent, and generate the appropriate access rights, the generation process should be run as user xfer.

Let's do a one shot test, using the configuration previously created:

```
su -s /bin/bash - xfer

or

sudo -u xfer /bin/bash

and then:
```

```
ascli preview scan --overwrite=always
```

When the preview generator is first executed it will create a file: <code>.aspera_access_key</code> in the preview's folder which contains the access key used. On subsequent run it reads this file and check that previews are generated for the same access key, else it fails. This is to prevent clash of different access keys using the same root.

18.6 Configuration for Execution in scheduler

Details are provided in section Scheduler.

Shorter commands can be specified if a configuration preset was created as shown previously.

For example the timeout value can be differentiated depending on the option: event versus scan:

18.7 Candidate detection for creation or update (or deletion)

ascli generates preview files using those commands:

- trevents : only recently uploaded files will be tested (transfer events)
- events: only recently uploaded files will be tested (file events: not working)
- scan: recursively scan all files under the access key's storage root
- test : test using a local file

Once candidate are selected, once candidates are selected, a preview is always generated if it does not exist already, else if a preview already exist, it will be generated using one of three values for the overwrite option:

- always: preview is always generated, even if it already exists and is newer than original
- never : preview is generated only if it does not exist already
- mtime : preview is generated only if the original file is newer than the existing

Deletion of preview for deleted source files: not implemented yet (TODO).

If the scan or events detection method is used, then the option: skip_folders can be used to skip some folders. It expects a list of path relative to the storage root (docroot) starting with slash, use the @json: notation, example:

```
ascli preview scan --skip-folders=@json:'["/not_here"]'
```

The option folder_reset_cache forces the node service to refresh folder contents using various methods.

When scanning the option query has the same behavior as for the node access_keys do self find command. Refer to that section for details.

18.8 Preview File types

Two types of preview can be generated:

- · png: thumbnail
- mp4: video preview (only for video)

Use option skip_format to skip generation of a format.

18.9 Supported input Files types

The preview generator supports rendering of those file categories:

- image
- pdf
- plaintext
- office
- video

To avoid generation for some categories, specify a list using option skip_types.

Each category has a specific rendering method to produce the PNG thumbnail.

The mp4 video preview file is only for category video

File type is primarily based on file extension detected by the Node API and translated info a mime type returned by the Node API.

18.10 mimemagic

By default, the Mime type used for conversion is the one returned by the Node API, based on file name extension.

It is also possible to detect the mime type using option mimemagic . To use it, set option mimemagic to yes: --mimemagic=yes .

This requires to manually install the mimemagic gem: gem install mimemagic.

In this case the preview command will first analyze the file content using mimemagic, and if no match, will try by extension.

If the mimemagic gem complains about missing mime info file:

- Any OS:
 - Examine the error message
 - Download the file: freedesktop.org.xml.in
 - move and rename this file to one of the locations expected by mimemagic as specified in the error message
- · Windows:
 - Download the file: freedesktop.org.xml.in
 - Place this file in the root of Ruby (or elsewhere): C:\RubyVV-x64\freedesktop.org.xml.in
 - Set a global variable using SystemPropertiesAdvanced.exe or using cmd (replace VV with version) to the exact path of this file:

```
SETX FREEDESKTOP_MIME_TYPES_PATH C:\RubyVV-x64\freedesktop.org.xml.in
```

- Close the cmd and restart a new one if needed to get refreshed env vars
- Linux RHFL 8+:

```
dnf install shared-mime-info
```

· macOS:

brew install shared-mime-info

18.11 Generation: Read source files and write preview

Standard open source tools are used to create thumbnails and video previews. Those tools require that original files are accessible in the local file system and also write generated files on the local file system. ascli provides 2 ways to read and write files with the option: file_access

If the preview generator is run on a system that has direct access to the file system, then the value local can be used. In this case, no transfer happen, source files are directly read from the storage, and preview files are directly written to the storage.

If the preview generator does not have access to files on the file system (it is remote, no mount, or is an object storage), then the original file is first downloaded, then the result is uploaded, use method remote.

18.12 Tested commands for preview

1 Note

Add ascli preview in front of the following commands:

test --mimemagic=yes --base=test my_jpg_unk trevents --once-only=yes --skip-types=office --log-level=info

IBM Aspera Sync

An interface for the async utility is provided in the following plugins:

- server sync (SSH Auth)
- node sync (use gen3 token)
- aoc files sync (uses node plugin with bearer token)
- shares files sync (uses node plugin with gen3 token)

The sync command, available in above plugins, performs the following actions:

- Start a local Sync session by executing the async command with the appropriate parameters.
- Get local Sync session information using the asyncadmin command, if available.
- Get local Sync session information accessing directly the Async snap database.

One advantage of using ascli over the async command line is the possibility to use a configuration file, using standard options of ascli. Moreover, ascli supports sync with application requiring token-based authorization.

Some sync parameters are filled by the related plugin using transfer spec parameters (e.g. including token).

A Note

All sync commands require an async enabled license and availability of the async executable (and asyncadmin). The Aspera Transfer Daemon 1.3+ includes this.

19.1 Starting a sync session

To start a sync session, use one of the three sync directions followed by a folder path (remote path for pull, local path elsewise). The path on the other side is specified using option: to_folder.

Direction(parameter)	Path(parameter)	to_folder (option)
push bidi	Local	Remote
	Local	Remote
pull	Remote	Local

An optional positional Hash argument (sync_info) can be provided in either conf (preferred) or args (legacy) format.

A single session can be specified using either formats.

19.1.1 sync info: conf format

This is the preferred syntax. It is the same payload as specified on the async option --conf or in Node API /asyncs

Documentation on Async Node API can be found on IBM Developer Portal.

Parameters local.path and remote.path are not allowed since they are provided on command line.

19.1.2 sync_info: args format

This is the <u>legacy</u> syntax. ascli defines a JSON equivalent to regular async options. It is based on a JSON representation of async command line options. Technically, it allows definition of multiple sync sessions in a single command, but ascli only accepts a single session for consistency with the previous syntax.

This is the mode selection if there are either keys sessions or instance in option sync_info.

Parameters local_dir and remote_dir are not allowed since they are provided on command line.

19.2 Sync management and monitoring

The admin command provides several sub commands:

- status: Uses the utility asyncadmin, available only on server products.
- Other commands access directly the Async snap database (snap.db).

For those commands, the user must provide the path to the database folder, i.e. a folder containing a subfolder named .private-asp . By default it is the local synchronized folder. If this folder contains only one session information (i.e. a folder containing the snap.db file), it will be used by default. Else, the user must specify a session name in the optional Hash , in name .

Hot folder

20.1 Requirements

ascli maybe used as a simple hot folder engine. A hot folder being defined as a tool that:

- Locally (or remotely) detects new files in a top folder
- Send detected files to a remote (respectively, local) repository
- Only sends new files, do not re-send already sent files
- Optionally: sends only files that are not still growing
- Optionally: after transfer of files, deletes or moves to an archive

In addition: the detection should be made continuously or on specific time/date.

20.2 Setup procedure

The general idea is to rely on:

- Existing ascp features for detection and transfer
- Take advantage of ascli configuration capabilities and server side knowledge
- The OS scheduler for reliability and continuous operation

20.2.1 ascp features

Interesting ascp features are found in its arguments: (see ascp manual):

- · Sending only new files
 - option -k 1,2,3 (resume_policy)
- Remove or move files after transfer:
 - --remove-after-transfer (remove after transfer)
 - --move-after-transfer (move_after_transfer)
 - --remove-empty-directories (remove_empty_directories)
- Send only files not modified since the last X seconds:
 - -- exclude-newer-than (exclude newer than)
 - --exclude-older-than (exclude_older_than)
- Top level folder shall not be created on destination
 - --src-base (src_base)

Note

ascli takes transfer parameters exclusively as a transfer-spec, with ts option.

Note

Usual native ascp arguments are available as standard <u>transfer-spec</u> parameters, but not special or advanced options.



Only for the direct transfer agent (not others, like connect or node), native ascp arguments can be provided with parameter ascp_args of option transfer_info.

20.2.2 Server side and configuration

Virtually any transfer on a repository on a regular basis might emulate a hot folder.

Note

File detection is not based on events (inotify , etc...), but on a simple folder scan on source side.

9 Tip

Options may be saved in an Option Preset and used with -P.

20.2.3 Scheduling

Once ascli command line arguments are defined, run the command using the OS native scheduler, e.g. every minute, or 5 minutes, etc... Refer to section Scheduler. (on use of option lock_port)

20.3 Example: Upload hot folder

The local folder (here, relative path: <code>source_hot</code>) is sent (upload) to an Aspera server. Source files are deleted after transfer. Growing files will be sent only once they don't grow anymore (based on an 8-second cool-off period). If a transfer takes more than the execution period, then the subsequent execution is skipped (<code>lock_port</code>) preventing multiple concurrent runs.

20.4 Example: Unidirectional synchronization (upload) to server

```
ascli server upload source_sync --to-folder=/Upload/target_sync --lock-port=12345

--ts=@json:'{"resume_policy":"sparse_csum","exclude_newer_than":-8,"src_base":"source_sync"}'
```

This can also be used with other folder-based applications: Aspera on Cloud, Shares, Node.

20.5 Example: Unidirectional synchronization (download) from Aspera on Cloud Files

```
ascli aoc files download . --to-folder=. --lock-port=12345 --progress-bar=no --display=data
--ts=@json:'{"resume_policy":"sparse_csum","target_rate_kbps":50000,"exclude_newer_than":-8,
--display=data
--ts=@json:'{"resume_policy":"sparse_csum","target_rate_kbps":50000,"exclude_newer_than":-8,
--display=data
```

Note

Option delete before transfer will delete files locally, if they are not present on remote side.

Note

Options progress and display limit output for headless operation (e.g. cron job)

Health check and Nagios

Most plugin provide a health command that will check the health status of the application. Example:

ascli console health

status	component	message
ok	console api	accessible

Typically, the health check uses the REST API of the application with the following exception: the server plugin allows checking health by:

- Issuing a transfer to the server
- Checking web app status with asctl all:status
- Checking daemons process status

ascli can be called by Nagios to check the health status of an Aspera server. The output can be made compatible to Nagios with option --format=nagios :

ascli server health transfer --to-folder=/Upload --format=nagios --progress-bar=no

OK - [transfer:ok]

SMTP for email notifications

ascli can send email, for that setup SMTP configuration. This is done with option smtp.

The smtp option is a Hash (extended value) with the following fields:

Field	Default	Example	Description
server	-	smtp.gmail.com	SMTP server address
tls	true	true	Enable STARTTLS (port 587)
ssl	false	false	Enable TLS (port 465)
port	587	587	Port for service
	465 25		
domain	domain of server	gmail.com	Email domain of user
username	-	john@example.com	User to authenticate on SMTP server Leave empty for open auth.
password	-	my_password_here	Password for above username
from_email	username if defined	johnny@example.com	Address used if receiver replies
from_name	same as email	John Wayne	Display name of sender

22.1 Example of configuration

```
ascli config preset set smtp_google server smtp.google.com
ascli config preset set smtp_google username john@gmail.com
ascli config preset set smtp_google password my_password_here
```

or

```
ascli config preset init smtp_google

→ @json:'{"server":"smtp.google.com","username":"john@gmail.com","password":"my_password_here"}'

or
```

```
ascli config preset update smtp_google --server=smtp.google.com --username=john@gmail.com
-- --password=my_password_here
```

Set this configuration as global default, for instance:

```
ascli config preset set cli_default smtp @val:@preset:smtp_google
ascli config preset set default config cli_default
```

22.2 Email templates

Sent emails are built using a template that uses the ERB syntax.

The template is the full SMTP message, including headers.

The following variables are defined by default:

- from_name
- from_email
- to

Other variables are defined depending on context.

22.3 Test

Check settings with smtp_settings command. Send test email with email_test.

```
ascli config --smtp=@preset:smtp_google smtp
ascli config --smtp=@preset:smtp_google email --notify-to=sample.dest@example.com
```

22.4 Notifications for transfer status

An e-mail notification can be sent upon transfer success and failure (one email per transfer job, one job being possibly multi session, and possibly after retry).

To activate, use option notify_to.

A default e-mail template is used, but it can be overridden with option notify_template.

The environment provided contains the following additional variables:

- subject : a default subject including transfer status
- status : global status of transfer
- ts: the transfer-spec used for the transfer
- from_email : email of sender (from smtp configuration)
- from_name : name of sender (from smtp configuration)
- to : recipient of the email (from notify_to)

Example of template:

```
From: <%=from_name%> <<%=from_email%>>
To: <<%=to%>>
Subject: <%=subject%>
Transfer is: <%=status%>
```

Tool: asession

This gem comes with a second executable tool providing a simplified standardized interface to start a FASP session: assession .

It aims at simplifying the startup of a FASP session from a programmatic standpoint as formatting a transfer-spec is:

- Common to Aspera Node API (HTTP POST /ops/transfer)
- Common to Aspera Connect API (browser JavaScript startTransfer)
- Easy to generate by using any third party language specific JSON library

Hopefully, IBM integrates this directly in ascp, and this tool is made redundant.

This makes it easy to integrate with any language provided that one can spawn a sub process, write to its STDIN, read from STDOUT, generate and parse JSON.

ascli expect one single argument: a session specification that contains parameters and a transfer-spec.

If no argument is provided, it assumes a value of: @json:@stdin:, i.e. a JSON formatted on stdin.

• Note

If JSON is the format, specify @json: to tell ascli to decode the Hash using JSON syntax.

During execution, it generates all low level events, one per line, in JSON format on stdout.

Top level parameters supported by assssion:

Parameter	Description
spec	The transfer-spec
agent	Same parameters as transfer-info for agent direct
loglevel	Log level of asession
file_list_folder	The folder used to store (for garbage collection) generated file lists.
	Default: [system tmp folder]/[username]_asession_filelists

23.1 Comparison of interfaces

Feature/Tool	Transfer Daemon	FASPManager	ascp	asession
status	Supported	Deprecated	Supported	Deprecated
language integration	Many	C/C++ C#/.net Go Python java	Any	Any

Feature/Tool	Transfer Daemon	FASPManager	ascp	asession
required additional components to ascp	Daemon	Library (+headers)	-	Ruby Aspera gem
startup	Daemon	API	Command line arguments	JSON on stdin (standard APIs: JSON.generate Process.spawn)
events	Poll	Callback	Possibility to open management port and proprietary text syntax	JSON on stdout
platforms	Like ascp and transferd	Like ascp and lib (if compiled)	Any with ascp	Any with Ruby and ascp

23.2 Simple session

Create a file session.json with:

```
{"remote_host":"demo.asperasoft.com","remote_user":"asperaweb","ssh_port":33001,"remote_password":

"my_password_here","direction":"receive","destination_root":"./test.dir","paths":[{"source":]

"/aspera-test-dir-tiny/200KB.1"}],"resume_level":"none"}
```

Then start the session:

asession < session.json

23.3 Asynchronous commands and Persistent session

asession also supports asynchronous commands (on the management port). Instead of the traditional text protocol as described in ascp manual, the format for commands is: one single line per command, formatted in JSON, where parameters shall be snake style, for example: LongParameter → long_parameter

This is particularly useful for a persistent session (with the transfer-spec parameter: "keepalive":true)

(events from FASP are not shown in above example. They would appear after each command)

23.4 Example of language wrapper

NodeJS: https://www.npmjs.com/package/aspera

23.5 Help

```
asession -h
USAGE
    asession
    asession -h|--help
    asession [<session spec extended value>]

If no argument is provided, default will be used: @json:@stdin
    -h, --help display this message
    <session spec extended value> a dictionary (Hash)
    The value can be either:
```

```
the JSON description itself, e.g. @json:'{"xx":"yy",...}'
       @json:@stdin, if the JSON is provided from stdin
       @json:@file:<path>, if the JSON is provided from a file
    The following keys are recognized in session spec:
       spec : mandatory, contains the transfer spec
       loglevel : modify log level (to stderr)
       agent : modify transfer agent parameters, e.g. ascp_args
       file_list_folder : location of temporary files
       sdk : location of SDK (ascp)
    Asynchronous commands can be provided on STDIN, examples:
       {"type":"START", "source": "/aspera-test-dir-tiny/200KB.2"} {"type": "START", "source": "xx", "destination": "yy"}
       {"type":"DONE"}
EXAMPLES
    asession @json:'{"spec":{"remote_host":"demo.asperasoft.com","remote_user":"asperaweb",
    "ssh_port":33001, "remote_password": "demoaspera", "direction": "receive", "destination_root": ". |
   /test.dir","paths":[{"source":"/aspera-test-dir-tiny/200KB.1"}]}}'
    echo '{"spec":{"remote_host":...}}'|asession @json:@stdin
```

Ruby Module: Aspera

Main components:

- Aspera generic classes for REST and OAuth
- Aspera::Agent::Direct: Starting and monitoring transfers using ascp.
- Aspera::Cli: ascli.

Working examples can be found in repo: https://github.com/laurent-martin/aspera-api-examples in Ruby examples.

History

When I joined Aspera, there was only one CLI: ascp, which is the implementation of the FASP protocol, but there was no CLI to access the various existing products (Server, Faspex, Shares). Once, Serban (founder) provided a shell script able to create a Faspex Package using Faspex REST API. Since all products relate to file transfers using FASP (ascp), I thought it would be interesting to have a unified CLI for transfers using FASP. Also, because there was already the ascp tool, I thought of an extended tool: eascp.pl which was accepting all ascp options for transfer but was also able to transfer to Faspex and Shares (destination was a kind of URI for the applications).

There were a few pitfalls:

- ascli was written in the aging perl language while most Aspera web application products (but the Transfer Server) are written in ruby.
- ascli was only for transfers, but not able to call other products APIs

So, it evolved into ascli:

- Portable: works on platforms supporting ruby (and ascp)
- Easy to install with the gem utility
- Supports transfers with multiple Transfer Agents, that's why transfer parameters moved from ascp command line to transfer-spec (more reliable, more standard)
- ruby is consistent with other Aspera products

Over the time, a supported command line tool aspera was developed in C++, it was later on deprecated. It had the advantage of being relatively easy to installed, as a single executable (well, still using ascp), but it was too limited IMHO, and lacked a lot of the features of this CLI.

Enjoy a coffee on me:

```
ascli config coffee --ui=text
ascli config coffee --ui=text --image=@json:'{"text":true}'
ascli config coffee
```

Common problems

ascli detects common problems and provides hints to solve them.

26.1 Error: "Remote host is not who we expected"

Cause: ascp >= 4.x checks fingerprint of the highest server host key, including ECDSA. ascp < 4.0 (3.9.6 and earlier) support only to RSA level (and ignore ECDSA presented by server). aspera.conf supports a single fingerprint.

Workaround on client side: To ignore the certificate (SSH fingerprint) add option on client side (this option can also be added permanently to the configuration file):

```
--ts=@json:'{"sshfp":null}'
```

Workaround on server side: Either remove the fingerprint from aspera.conf , or keep only RSA host keys in sshd_config.

References: ES-1944 in release notes of 4.1 and to HSTS admin manual section "Configuring Transfer Server Authentication With a Host-Key Fingerprint".

26.2 Error: "can't find header files for ruby"

Some Ruby gems dependencies require compilation of native parts (C). This also requires Ruby header files. If Ruby was installed as a Linux Packages, then also install Ruby development package: ruby-dev or ruby-devel, depending on distribution.

26.3 ED25519 key not supported

ED25519 keys are deactivated since ascli version 0.9.24 as it requires additional gems that require native compilation and thus caused problems. This type of key will just be ignored.

To re-activate, set env var ASCLI ENABLE ED25519 to true.

Without this deactivation, if such key was present in user's ...sh folder then the following error was generated:

```
OpenSSH keys only supported if ED25519 is available
```

Which meant that you do not have Ruby support for ED25519 SSH keys. You may either install the suggested Gems, or remove your ed25519 key from your .ssh folder to solve the issue.

In addition, host keys of type: ecdsa-sha2 and ecdh-sha2 are also deactivated by default. To re-activate, set env var ASCLI_ENABLE_ECDSHA2 to true.

26.4 JRuby: net-ssh: Unsupported algorithm

JRuby may not implement all the algorithms supported by OpenSSH.

Add the following to option ssh_options:

```
{"host_key":["rsa-sha2-512","rsa-sha2-256"],"kex":["curve25519-sha256","diffie-hellman-group14-]

sha256"],"encryption": ["aes256-ctr", "aes192-ctr", "aes128-ctr"]}

e.g.

--ssh-options=@json:'{"host_key":["rsa-sha2-512","rsa-sha2-256"],"kex":["curve25519-sha256","diffie-]

hellman-group14-sha256"],"encryption": ["aes256-ctr", "aes192-ctr", "aes128-ctr"]}'
```

26.5 Error: "SSL_read: unexpected eof while reading"

Newer OpenSSL library expects a clean SSL close. To deactivate this error, enable option IGNORE_UNEXPECTED_EOF for ssl_options in option http_options .

```
--http-options=@json:'{"ssl_options":["IGNORE_UNEXPECTED_EOF"]}'
```

26.6 Error: ascp:/lib64/libc.so.6: version 'GLIBC_2.28' not found

This happens on Linux x86 if you try to install transferd on a Linux version too old to support a newer executable.

Workaround: Install an older version:

```
ascli config transferd install 1.1.2
```

Refer to: Binary

26.7 Error: Cannot rename partial file

This is an error coming from ascp when it is configured to use a partial file name, and at the end of the transfer, the partial file, now complete, does not exist anymore.

This often happens when two transfers start in parallel for the same file:

- session 1 starts for file1, it creates file: file1.partial and fills it
- session 2 starts for file1 (same), it creates file: file1.partial and fills it
- session 1 finishes, and renames file1.partial to file1
- session 2 finishes, and tries to rename file1.partial to file1, but it fails as it does not exist anymore...

By default, ascli creates a config file: ~/.aspera/sdk/aspera.conf like this:

In container, this is located in /ibm_aspera.

One possibility to avoid that error is to disable partial filename suffix... But that only hides the problem.

For example, when using the container, override that file with a volume and remove the line for extension. Another possibility is to add this option: |--transfer-info==@json:'{"ascp_args":["--partial-file-suffix="]}': this overrides the value in config file.

• Note

If one relies on --lock-port when using containers to avoir parallel transfers in a cron job, this can be the problem, as lock_port does not lock between containers. Use flock instead.

