ADMINISTRATION CONSOLE COMMAND-LINE INTERFACE OVERVIEW

Overview

APPLIES TO:
CrashPlan PROe

You can interact with your Code42 environment through a command-line interface (CLI) as well as the regular graphical interface of the administration console. This article describes the CLI commands available and their uses.

To access the command-line interface, double click the logo in the header of the console.

Before You Begin

If you have any questions about command usage or best practices, contact our Customer Champions.

Some of these commands have the potential to interfere with backups, disconnect devices, or potentially destroy data. Some commands cannot be undone.

Because of this destructive potential, take care when modifying your Code42 environment on the command line.

Reading This Overview

This overview uses several standard conventions to describe command parameters and options.

Arrow Brackets

Text in arrow brackets should be replaced with text specific to your environment. Do not include the arrow brackets in your command.

Example instruction: `command <exampleEmail>`

Correct commands:
`command john.doe@code42.com`

Square Brackets

Text in square brackets indicates an optional command. Do not include the square brackets in your command.

Example instruction: `command [option]`

Correct commands:
`command`
`command option`

Curly Braces

Text in curly braces separated by pipes indicates mutually exclusive arguments. Do not include the curly braces in your command.

Example instruction: `command {abc | def | ghi}`

Incorrect command:

```
command abc def
```

Correct commands:

```
command abc
command ghi
```

Accessing The Command-Line Interface

To interact with your Code42 environment through the command-line interface, open the CLI from the administration console.

1. Sign in to your administration console in a web browser.
2. Double-click the logo in the upper left corner of the administration console.
   The command-line interface appears in the administration console.
3. Enter commands at the top of the command-line interface. Output can be reviewed below.

Commands

admin.backup.alerter

Send each org manager an alert email listing computers that have been offline or have incomplete backups.

Usage:

```
admin.backup.alerter
```

admin.backup.reporter

Send each org manager one email report of the backup status for each of their organizations.

Usage:

```
admin.backup.reporter
```

alertConnected

Print an alert message at the top of the Backup page on all currently connected clients.

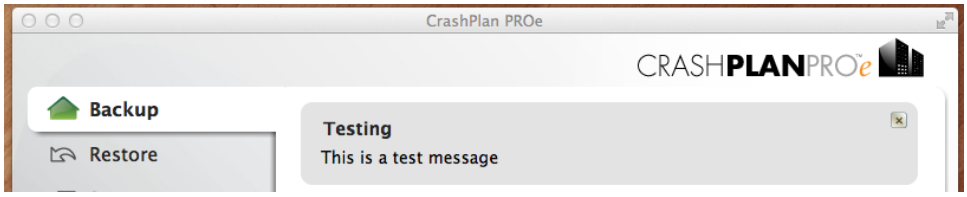
Usage:

```
alertConnected <title> <message>
```

Argument	Function
`<title>`	Title of the alert message
`<message>`	Body text of the alert message

Example:

`alertConnected "Testing" "This is a test message"` would display for all clients currently connected:



analyze.database

Reindex and optimize database tables. This process is run automatically as part of the daily services.

Usage:

```
analyze.database
```

applog

Print the current configuration information to the application log.

Usage:

```
applog
```

archive.records

Create a new statistics record for organizations, users, and store points, and move the previous entries into the history record. This service runs automatically at the time specified under **Settings** > **Server** (default: 2:01 am).

Usage:

```
archive.records
```

balance.calculate

Print the current data balancing information (sources, targets, and matchups) to the application log.

Usage:

```
balance.calculate
```

balance.data

Immediately perform data balancing operations.

Usage:

```
balance.data
```

clientCommand

Send a command, enclosed in quotation marks, to a client. If the client is not currently online, the command will fail and no action will be taken.

Usage:

```
clientCommand "<command>" <guid>
```

Alias:

```
cc "<command>" <guid>
```

Argument	Function
<command>	Command to pass to the client device
<guid>	GUID of the client device

Example:

```
clientcommand "java option Xmx 1024m" 123456789012345678
```

computer

Return user ID, name, address, and other information fields for a specific computer.

Usage:

```
computer <guid> [deauth]
```

Argument	Function
<guid>	GUID of the client device
[deauth]	(Optional) Return deauthorization information instead of standard output

crypto

Cryptographic utility.

Usage:

```
crypto {status | encrypt <string> | decrypt <string> | hash <string>
| hash512 <string> | keystore enable <password> <retype password> |
keystore unlock <password>}
```

Argument	Function
status	Returns the current state of the secure keystore mode.
encrypt <string>	Encrypt a given string using Blowfish. `Encrypt` requires secure keystore mode to be enabled.
decrypt <string>	Decrypt a given Blowfish-encrypted string. `Decrypt` requires secure keystore mode to be enabled.

`hash <string>`	Hash a given string in a secure, unique, irreversible way. The string is encoded with Base64. The salt is discarded since the hash is irreversible and non-testable. The argument `hash` uses the SHA-1 function, which is less secure than `hash512` but creates a smaller hash result.
`hash512 <string>`	Hash a given string in a secure, unique, irreversible way. The string is encoded with Base64. The salt is discarded since the hash is irreversible and non-testable. The argument `hash512` uses the SHA-512 function, which is more secure than `hash` but creates a larger hash result.
`keystore enable <password> <retype password>`	**Warning**: Secure keystore mode will require a password to be entered by the administrator on every server startup. **Warning**: Secure keystore mode **cannot** be disabled. Enable secure keystore mode: all archive, transport, and encryption keys will be encrypted. Secure keystore mode can only be enabled if there are no computers in the deployment.
`keystore unlock <password>`	Unlock the keystore to allow administrator access. This is a temporary option required on each server startup, used once `keystore enable` has been used to activate secure keystore mode. `Keystore unlock` does **not** reverse the effects of `keystore enable`.

daily.services

Manually initiate all services that run daily, and update usage statistics displayed by the administration console. Services include:

- BulkStatsBroadcast
- LicenseNotificationService
- UsageUpdaterService
- ConnectionAlerter
- DatabaseDump
- StorageNodeSyncJob
- ProviderSyncJob
- ArchiveRecordService
- BackupReporter
- AnalyzeDatabase
- RepositoryDiskVacuumer

Usage:

```
daily.services
```

directory.sync

Compare the local user directory with an external user directory, such as LDAP, and update the local directory if needed. Your Code42 environment can only read from the external directory, never write to it.

Usage:

```
directory.sync [simulate]
```

Argument	Function
`[simulate]`	(Optional) Return directory information without updating the local directory.

disconnect

Disconnect all devices, or a single device if given a GUID, from all cloud (non-provider) destinations. Does not deauthorize the device.

Usage:

```
disconnect [<guid>]
```


Argument	Function
[<guid>]	(Optional) GUID of the device you want to disconnect from all cloud destinations. If guid is not specified, disconnect disconnects all devices from all cloud destinations.

diskspace.alerter

Check the available space on each store point for this enterprise server, and send an email at the thresholds defined in **Settings -> Server -> Alerts**.

Usage:

```
diskspace.alerter
```

dump

Send (or "dump") system properties, backup properties, stack traces, disk, and network information to the log file.

Usage:

```
dump {all | app | env | backup | manifest | memory | netstat |
ifconfig | peer | threads | vss | disk | files}
```

Argument	Function
all	Dump all system properties
app	Dump application information
env	Dump environment information
backup	Dump backup information
manifest	Dump manifest information
memory	Dump memory information
netstat	Dump network statistics information
ifconfig	Dump IP configuration information
peer	Dump peering information
threads	Dump threading information
vss	Dump Volume Shadow Service information (Windows only)
disk	Dump disk information
files	Dump file information

email

Send an email.

Usage:

`email online` - normal online mode

`email offline <durationInSeconds>` - simulate an offline email server

`email send <sender> <recipient> <subject> [<body>] [count=<n>]` - send email

`email log {on|off}` - instead of sending emails to email server, simply log that the emails were sent.

Argument	Function
online	Set your master server's email service to normal online mode
offline	Simulate an offline email server. Requires an argument for durationInSeconds.
<durationInSeconds>	The duration, given in seconds, of the offline simulation. Used only for the offline argument.
send	Send an email using your master server's email service
<sender>	The sender of your email. Used only for the send argument.

`<recipient>`	The recipient of your email. Used only for the `send` argument.
`<subject>`	The subject line of your email. Used only for the `send` argument.
`<body>`	(Optional) The body text of your email. Used only for the `send` argument.
`[count=<n>]`	(Optional) The number of emails to send. Used only for the `send` argument.
`log`	Append log information to the email server **without** sending the emails. Requires either the `on` or `off` argument.

export.database

Manually run the export database process, which exports the database to the dumps folder in your installation directory and the root of each store point of this enterprise server.

Usage:

```
export.database
```

getLogs

Generate a copy of all server logs in a a zip file in the log directory.

Usage:

```
getLogs
```

hierarchy.sync

Synchronize hierarchy and business objects, including information on orgs, store points, and storage servers.

Usage:

```
hierarchy.sync
```

Example:

Use `hierarchy.sync` to update your Code42 environment if you have added an org and want to immediately update that information to all storage servers.

license.alerter

Compare the number of available licenses, and send an email at the thresholds defined in **Settings -> Server -> Alerts**.

Usage:

```
license.alerter
```

log

Control logging levels.

Usage:

```
log [<category>] [<level>]
```

Argument	Function
`[<category>]`	(Optional) Specify the category of logs to modify. Common options: com.code42.core.ldap com.code42.email.Emailer com.code42.ssoauth The complete list of options is available by contacting our Customer Champions.
`[<level>]`	(Optional) Specify the level of logging. Options, listed from most severe to least severe: • Error • Warn • Info • Fine • Trace

Examples:

`log com.code42.core.ldap FINE` - Enable debug-level logging for LDAP
`log com.code42.email.Emailer TRACE` - Enable trace-level logging for e-mail system logs
`log com.code42.ssoauth TRACE` - Enable trace-level logging for all SSO authorization modules

login.monitor

Information and operations pertaining to the login attempt monitor.

Usage:

```
login.monitor {info | disable | enable | reset <username> |
maxRetries <number> | lockoutTime <minutes> }
```

Argument	Function
`info`	Print the status of the login monitor, the maximum number of retries, and the lockout duration for failed login attempts
`disable`	Disable login attempt monitoring
`enable`	Enable login attempt monitoring
`reset <username>`	Reset a given user's login attempt status. Requires the `<username>` argument.
`maxRetries <number>`	Change the number of retries allowed before a user is locked out.
`lockoutTime {<minutes> | forever}`	Change the lockout period to `minutes`. Specifying `forever` instead of `minutes` will permanently lock out users who fail to log in correctly, until an administrator unlocks their accounts.

node.restart

Instruct a node to restart.

Usage:

```
node.restart [<guid>]
```

Argument	Function
`[<guid>]`	(Optional) GUID of the enterprise server to restart. If `guid` is not specified, `node.restart` will restart the active enterprise server.

node.sync

Synchronize a storage server with the master server

Usage:

```
node.sync [<guid>]
```

Argument	Function
`[<guid>]`	(Optional) GUID of the storage server to sync with the master server. If `guid` is not specified, `node.sync` will sync the active storage server with the master server.

node.upgrade

Run the storage node automatic upgrade process.

Usage:

```
node.upgrade <guid> [<remote version>]
```

Argument	Function
`<guid>`	GUID of the storage server to upgrade. If `guid` is not specified, `node.upgrade` will upgrade the active storage server.
`[<remote version>]`	(Optional) The remote version to upgrade

org.destination.script.validate

Validate an organization's destination assignment script.

Usage:

```
org.destination.script.validate <orgId>
```

Argument	Function
<orgId>	GUID of the organization to validate

prop.remove

Remove a system property. System properties are case-sensitive.

Usage:

```
prop.remove <name>
```

Argument	Function
<name>	Name of the system property to remove

prop.set

Set system property. System properties are case-sensitive.

Usage:

```
prop.set <name> <value> [save] [{none|all|destination <guid>}]
```

Argument	Function
<name>	Name of the system property to set
<value>	Value to assign to the system property
[save]	(Optional) Cause the property setting to persist after reboot. If `save` is not specified, the property setting will revert to its previous setting after reboot.
none	Apply the property setting to no enterprise server
all	Apply the property setting to all enterprise servers in your Code42 environment. Enterprise servers must be currently online.
destination <guid>	Apply the property setting to a specific destination, given by the destination's GUID. Destination must be currently online.
	If `none`, `all`, or `destination <guid>` are not specified, `prop.set` will set the system property on the local enterprise server.

prop.show

Show system properties.

Usage:

```
prop.show [all] [<filter_string>]
```

Argument	Function
[all]	(Optional) List all system properties
[<filter_string>]	Filter the list of system properties with a given string

provider.sync

Synchronizes all data in local server database with data in other master servers for which this server is the provider.

Usage:

```
provider.sync [<hostedOrgId>]
```


Argument	Function
[<hostedOrgId>]	(Optional) Specify to sync the database with a specific hosted org, given by the org's GUID. If `hostedOrgID` is not specified, `provider.sync` will sync all data in the database with other master servers for which this server is the provider.

reconnect

Reconnect all or the given computer to its assigned cloud destination.
Usage:
`reconnect [<guid>]`

Argument	Function
[<guid>]	(Optional) GUID of the computer to reconnect. If `guid` is not specified, `reconnect` will reconnect all computers to their assigned cloud destinations.

relation.sync

Synchronize store points between servers. Code42 prefers the use of `node.sync`.
Usage:
`relation.sync`

repository.vacuumer

Run the Repository Disk Vacuumer service to purge archives that have passed the cold storage retention limit.

Usage:
`repository.vacuumer`

Alias:
`rdv`

sendAuthorize

Automatically authorize deactivated devices. If a device is offline, then it will automatically authorize when it connects.
Usage:
`sendAuthorize <guids> [config] [show]` - colon separated.

Argument	Function
<guids>	GUIDs of the devices you want to authorize. Separate multiple GUIDs with colons.
[config]	(Optional) Authorize the given devices, and force the configuration stored on the enterprise server to overwrite the configuration on the target devices.
[show]	(Optional) Show the authorization queue.

Example:
`sendAuthorize 1111111111111111:22222222222222222:333333333333333333 config` - Authorize the three devices with the GUIDs 1111111111111111, 22222222222222222, and 333333333333333333 and overwrite their local configuration files with the configuration file from the enterprise server

shutdown

Shut down the enterprise server service.
Usage:
`shutdown`

Alias:
`stop`

```
exit
```

space.show

Show the map names, keys in a map, or the value of a key.

Usage:

```
space.show {maps|<mapName>} [<keyPrefix>] [<limit>]
```

Argument	Function
`maps`	Show the names of all maps in use
`<mapName>`	Show all keys in the `mapName` map
`[<keyPrefix>]`	(Optional) Filter the keys to only those that begin with `keyPrefix`
`[<limit>]`	(Optional) Show only the first `limit` number of keys

Examples:

`space.show maps` - shows the names of all maps in use

`space.show default` - shows all keys in the 'default' map (the default limit is 100 keys)

`space.show default /st/so/ 50` - shows the first 50 keys starting with '/st/so/' in the 'default' map

`space.show default /st/so/destguid=42` - shows the object stored in map 'default' at key '/st/so/destguid=42'

system.alert

List, test, or clear system alerts.

Usage:

```
system.alert {list|test|clear [<type>]|info}
```

Argument	Function
`list`	List all system alerts
`test`	Generate a test alert for each type of alert
`clear [<type>]`	Clear alerts. If the optional argument `type` is included, clear alerts of type `type`. If `type` is not included, clear all alerts.
`info`	Show information on the types of alerts available

test.email

Send one sample email or a set of sample emails.

Usage:

```
test.email <username> {customizable|code42|subscription|
<templatePath>}
```

Argument	Function
`<username>`	Recipient of the sample email, given by a Code42 environment username
`customizable`	Customize the content of the test email
`code42`	Send a standard Code42 email
`subscription`	Send a standard subscription notice email
`<templatePath>`	Send an email defined by a template stored in your Code42 environment installation path. For example: `/emails/templated/recover_password.eml`

usage.updater

Update the archiveBytes value with actual usage data from this enterprise server.

Usage:

```
usage.updater
```

version

Print the version date/time string given a long version number.
Usage:

```
version [<number>]
```

web

Control configuration of web server redirects on the enterprise server.
Usage:

```
web redirect {enable|disable|show|reload}
```

Argument	Function
`enable`	Enable redirection of web server subfolders.
`disable`	Disable redirection of web server subfolders. All currently active administration console sessions are logged out and the CrashPlan PROe web service restarts.
`show`	Show the full list of web server redirections.
`reload`	Reload the list of redirections from the configuration file stored on the enterprise server.

Related Topics

- Adding Users From The Administration Console
- Setting Up The PROe Cloud
- Managed Appliance Installation Guide
- SharePlan Quick Start Guide
- Admin Restore Overview & Settings