

CRaSH guide

CRaSH



Julien Viet (eXo Platform)

Copyright © 2011

Preface	iii
1. Interacting with the shell	1
1.1. Running CRaSH	1
1.1.1. Web archive deployment	1
1.1.2. Standalone mode	1
1.2. Shell usage	1
1.2.1. Connection	1
1.2.2. Features	2
1.3. Command usage	2
1.3.1. Getting basic help	2
1.3.2. Command line usage	3
1.4. Base commands	5
1.4.1. <code>sleep</code> command	5
1.4.2. <code>man</code> command	5
1.4.3. <code>log</code> command	5
1.4.4. <code>thread</code> command	7
1.4.5. <code>system</code> command	8
1.4.6. <code>memory</code> command	9
1.4.7. <code>jdbc</code> command	9
1.5. JCR	10
1.5.1. JCR commands	10
1.5.2. SCP usage	19
2. Configuring the shell	20
2.1. Change the SSH server key	20
2.2. Change the ports of the telnet or SSH server	20
2.3. Remove the telnet or SSH access	20
2.4. Configure the shell default message	20
3. Extending the shell	22
3.1. Groovy command system	22
3.1.1. Groovy file	22
3.1.2. Groovy execution	22
3.1.3. Shell context	23
4. Developers	24
4.1. <code>settings.xml</code> (maven)	24
5. Hey, I want to contribute!	25

Preface

The Common Reusable SHell (CRaSH) deploys in a Java runtime and provides interactions with the JVM. Commands are written in Groovy and can be developed at runtime making the extension of the shell very easy with fast development cycle.

The `bye` command disconnect from the shell.

1.2.1.2. SSH access

SSH connection is done on port 2000 with the password **crash** :

```
juliens-macbook-pro:~ julien$ ssh -p 2000 -l root localhost
root@localhost's password:
CRaSH 1.0.0-beta22 (http://crsh.googlecode.com)
Welcome to juliens-macbook-pro.local!
It is Fri Jan 08 21:12:53 CET 2010 now.
%
```

The `bye` command disconnect from the shell.

1.2.1.3. Native access

A third mode is available for standalone CRaSH usage because it uses the JVM native input and output. When you are using it, CRaSh will be available just after the JVM is launched.

1.2.2. Features

- Line edition: the current line can be edited via left and right arrow keys
- History: the key up and key down enable history browsing
- Quoting: simple quotes or double quotes allow to insert blanks in command options and arguments, for instance *"old boy"* or *'old boy'*. One quote style can quote another, like *"ol' boy"*.
- Completion: an advanced completion system is available

1.3. Command usage

1.3.1. Getting basic help

The `help` command will display the list of known commands by the shell.

```
[ / ]% help
% help
Try one of these commands with the -h or --help switch:

  cd           changes the current node
  commit      saves changes
  consume     collects a set of nodes
  cp          copy a node to another
  env         display the term env
  exportworkspace Export a workspace on the file system (experimental)
  fail        Fails
  help        provides basic help
  importworkspace Import a workspace from the file system (experimental)
  invoke      Invoke a static method
  log         logging commands
  ls          list the content of a node
  man         format and display the on-line manual pages
  mixin       mixin commands
  mv          move a node
  node        node commands
  produce     produce a set of nodes
  pwd         print the current node path
```

rm	remove one or several node or a property
rollback	rollback changes
select	execute a JCR sql query
setperm	modify the security permissions of a JCR node
sleep	sleep for some time
thread	vm thread commands
version	versioning commands
wait	Invoke a static method
ws	workspace commands
xpath	execute a JCR xpath query

1.3.2. Command line usage

The basic CRaSH usage is like any shell, you just type a command with its options and arguments. However it is possible to compose commands and create powerful combinations.

1.3.2.1. Basic command usage

Typing the command followed by options and arguments will do the job

```
% ls /
...
```

1.3.2.2. Command help display

Any command help can be displayed by using the `-h` argument:

```
% ls -h
usage: ls [-h | --help] [-h | --help] [-d | --depth] path

[-h | --help] command usage
[-h | --help] command usage
[-d | --depth] Print depth
path           the path of the node content to list
```

In addition of that, commands can have a complete manual that can be displayed thanks to the `man` command:

```
% man ls
NAME
    ls - list the content of a node

SYNOPSIS
    ls [-h | --help] [-h | --help] [-d | --depth] [-d | --depth] path

DESCRIPTION
    The ls command displays the content of a node. By default it lists the content of the current node, however it
    accepts a path argument that can be absolute or relative.

[//]% ls
/
+-properties
| +-jcr:primaryType: nt:unstructured
| +-jcr:mixinTypes: [exo:owneable,exo:privilegeable]
| +-exo:owner: '__system'
| +-exo:permissions: [any read,*:/platform/administrators read,*:/platform/administrators add_node,*:/pl
+-children
| +-/workspace
| +-/contents
| +-/Users
| +-/gadgets
| +-/folder

PARAMETERS
    [-h | --help]
        Provides command usage
```

```

[-h | --help]
    Provides command usage

[-d | --depth]
    Print depth

path
    the path of the node content to list

```

1.3.2.3. Advanced command usage

A CRaSH command is able to consume and produce a stream of object, allowing complex interactions between commands where they can exchange stream of compatible objects. Most of the time, JCR nodes are the objects exchanged by the commands but any command is free to produce or consume any type.

By default a command that does not support this feature does not consumer or produce anything. Such commands usually inherits from the `org.crsh.command.ClassCommand` class that does not care about it. If you look at this class you will see it extends the `org.crsh.command.BaseCommand`.

More advanced commands inherits from `org.crsh.command.BaseCommand` class that specifies two generic types `<C>` and `<P>`:

- `<C>` is the type of the object that the command consumes
- `<P>` is the type of the object that the command produces

The command composition provides two operators:

- The pipe operator `|` allows to stream a command output stream to a command input stream
- The distribution operator `+` allows to distribute an input stream to several commands and to combine the output stream of several commands into a single stream.

1.3.2.4. Connecting a `<Void,Node>` command to a `<Node,Void>` command through a pipe

Example 1.1. Remove all `nt:unstructured` nodes

```
% select * from nt:unstructured | rm
```

1.3.2.5. Connecting a `<Void,Node>` command to two `<Node,Void>` commands through a pipe

Example 1.2. Update the security of all `nt:unstructured` nodes

```
% select * from nt:unstructured | setperm -i any -a read + setperm -i any -a write
```

1.3.2.6. Connecting two `<Void,Node>` command to a `<Node,Void>` commands through a pipe

Example 1.3. Add the mixin `mix:referenceable` to any node of type `nt:file` or `nt:folder`

```
% select * from nt:file + select * from nt:folder | admixin mix:referenceable
```

1.3.2.7. Mixed cases

When a command does not consume a stream but is involved in a distribution it will not receive any stream but will be nevertheless invoked.

Likewise when a command does not produce a stream but is involved in a distribution, it will not produce anything but will be nevertheless invoked.

1.4. Base commands**1.4.1. `sleep` command**

```
NAME
    sleep - sleep for some time

SYNOPSIS
    sleep [-h | --help] time

PARAMETERS
    [-h | --help]
        Provides command usage

    time
        Sleep time in seconds
```

1.4.2. `man` command

```
NAME
    man - format and display the on-line manual pages

SYNOPSIS
    man [-h | --help] command

PARAMETERS
    [-h | --help]
        Provides command usage

    command
        the command
```

1.4.3. `log` command

```
NAME
    log add - create one or several loggers

SYNOPSIS
    log [-h | --help] add ... name
```


PARAMETERS

```
[-h | --help]
    Provides command usage

... name
    The name of the logger
```

NAME

log set - configures the level of one of several loggers

SYNOPSIS

```
log [-h | --help] set [-l | --level] [-p | --plugin] ... name
```

DESCRIPTION

The set command sets the level of a logger. One or several logger names can be specified as arguments and the -l option specify the level among the trace, debug, info, warn and error levels. When no level is specified, the level is cleared and the level will be inherited from its ancestors.

```
% logset -l trace foo
% logset foo
```

The logger name can be omitted and instead stream of logger can be consumed as it is a <Logger,Void> command. The following set the level warn on all the available loggers:

```
% log ls | log set -l warn
```

PARAMETERS

```
[-h | --help]
    Provides command usage

[-l | --level]
    The logger level to assign among {trace, debug, info, warn, error}

[-p | --plugin]
    Force the plugin implementation to use

... name
    The name of the logger
```

NAME

log send - send a message to a logger

SYNOPSIS

```
log [-h | --help] send [-m | --message] [-l | --level] name
```

DESCRIPTION

The send command log one or several loggers with a specified message. For instance the following impersonate the javax.management.mbeanserver class and send a message on its own logger.

```
##% log send -m hello javax.management.mbeanserver
```

Send is a <Logger, Void> command, it can log messages to consumed log objects:

```
% log ls | log send -m hello -l warn
```

PARAMETERS

```
[-h | --help]
    Provides command usage

[-m | --message]
    The message to log

[-l | --level]
    The logger level to assign among {trace, debug, info, warn, error}

name
    The name of the logger
```

NAME

log info - display info about a logger

SYNOPSIS

```
log [-h | --help] info ... name
```

DESCRIPTION

The loginfo command displays information about one or several loggers.

```
% loginfo javax.management.modelmbean
javax.management.modelmbean<INFO>
```

The loginfo command is a <Logger,Void> command and it can consumed logger produced by the logls command:

```
% logls -f javax.* | loginfo
javax.management.mbeanserver<INFO>
javax.management.modelmbean<INFO>
```

PARAMETERS

```
[-h | --help]
    Provides command usage

... name
    The name of the logger
```

NAME

```
log ls - list the available loggers
```

SYNOPSIS

```
log [-h | --help] ls [-f | --filter]
```

DESCRIPTION

The logls command list all the available loggers., for instance:

```
% logls
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/].[default]
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/eXoGadgetServer].[concat]
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/dashboard].[jsp]
...
```

The -f switch provides filtering with a Java regular expression

```
% logls -f javax.*
javax.management.mbeanserver
javax.management.modelmbean
```

The logls command is a <Void,Logger> command, therefore any logger produced can be consumed.

PARAMETERS

```
[-h | --help]
    Provides command usage

[-f | --filter]
    A regular expressions used to filter the loggers
```

1.4.4. thread command

NAME

```
thread stop - stop vm threads
```

SYNOPSIS

```
thread [-h | --help] stop
```

DESCRIPTION

Stop a VM thread, this method cannot be called as is and should be used with a pipe to consume a list of

PARAMETERS

```
[-h | --help]
    Provides command usage
```

NAME

thread ls - list the vm threads

SYNOPSIS

thread [-h | --help] ls [-n | --name] [-f | --filter] [-s | --state]

PARAMETERS

[-h | --help]

Provides command usage

[-n | --name]

Retain the thread with the specified name

[-f | --filter]

Filter the threads with a regular expression on their name

[-s | --state]

Filter the threads by their status (new,runnable,blocked,waiting,timed_waiting,terminated)

1.4.5. system command

NAME

system get - get the vm system properties

SYNOPSIS

system [-h | --help] get name

PARAMETERS

[-h | --help]

Provides command usage

name

The name of the property

NAME

system remove - remove the vm system properties

SYNOPSIS

system [-h | --help] remove name

PARAMETERS

[-h | --help]

Provides command usage

name

The name of the property

NAME

system set - set the vm system properties

SYNOPSIS

system [-h | --help] set name value

PARAMETERS

[-h | --help]

Provides command usage

name

The name of the property

value

The value of the property

NAME

system ls - list the vm system properties

```
SYNOPSIS
    system [-h | --help] ls
```

```
PARAMETERS
    [-h | --help]
    Provides command usage
```

1.4.6. memory command

```
NAME
    memory gc - call garbage collector
```

```
SYNOPSIS
    memory [-h | --help] gc
```

```
PARAMETERS
    [-h | --help]
    Provides command usage
```

```
NAME
    memory total - show total memory
```

```
SYNOPSIS
    memory [-h | --help] total [-u | --unit] [-d | --decimal]
```

```
PARAMETERS
    [-h | --help]
    Provides command usage

    [-u | --unit]
    The unit of the memory space size {(B)yte, (O)ctet, (M)egaOctet, (G)igaOctet}

    [-d | --decimal]
    The number of decimal (default 0)
```

```
NAME
    memory free - show free memory
```

```
SYNOPSIS
    memory [-h | --help] free [-u | --unit] [-d | --decimal]
```

```
PARAMETERS
    [-h | --help]
    Provides command usage

    [-u | --unit]
    The unit of the memory space size {(B)yte, (O)ctet, (M)egaOctet, (G)igaOctet}

    [-d | --decimal]
    The number of decimal (default 0)
```

1.4.7. jdbc command

```
NAME
    jdbc close - close the current connection
```

```
SYNOPSIS
    jdbc [-h | --help] close
```

```
PARAMETERS
    [-h | --help]
```

Provides command usage

```

NAME
    jdbc query - execute SQL query

SYNOPSIS
    jdbc [-h | --help] query ... sqlQuery

PARAMETERS
    [-h | --help]
        Provides command usage

    ... sqlQuery
        The query
    
```

```

NAME
    jdbc open - open connection from datasource

SYNOPSIS
    jdbc [-h | --help] open datasource

PARAMETERS
    [-h | --help]
        Provides command usage

    datasource
        The datasource
    
```

```

NAME
    jdbc connect - connect to database with JDBC connection string

SYNOPSIS
    jdbc [-h | --help] connect connectionString

PARAMETERS
    [-h | --help]
        Provides command usage

    connectionString
        The connection string
    
```

1.5. JCR

1.5.1. JCR commands

1.5.1.1. ws command

```

NAME
    ws login - login to a workspace

SYNOPSIS
    ws [-h | --help] login [-u | --username] [-p | --password] [-c | --container] workspaceName

DESCRIPTION
    This command login to a JCR workspace and establish a session with the repository.
    When you are connected the shell maintain a JCR session and allows you to interact with the session in a
    oriented fashion. The repository name must be specified and optionally you can specify a user name and pa
    have more privileges.

    % ws login -c portal portal-system
    
```

```
Connected to workspace portal-system

% ws login -c portal -u root -p gtn portal-system
Connected to workspace portal-system
```

PARAMETERS

```
[-h | --help]
    Provides command usage

[-u | --username]
    The user name

[-p | --password]
    The user password

[-c | --container]
    The portal container name (eXo JCR specific)

workspaceName
    The name of the workspace to connect to
```

NAME

ws logout - logout from a workspace

SYNOPSIS

ws [-h | --help] logout

DESCRIPTION

This command logout from the currently connected JCR workspace

PARAMETERS

```
[-h | --help]
    Provides command usage
```

1.5.1.2. cd command

NAME

cd - changes the current node

SYNOPSIS

cd [-h | --help] path

DESCRIPTION

The cd command changes the current node path. The command used with no argument changes to the root node. A relative or absolute path argument can be provided to specify a new current node path.

```
[/]% cd /gadgets
[/gadgets]% cd /gadgets
[/gadgets]% cd
[/]%
```

PARAMETERS

```
[-h | --help]
    Provides command usage
```

path
The new path that will change the current node navigation

1.5.1.3. pwd command

NAME

pwd - print the current node path

SYNOPSIS

pwd [-h | --help]

DESCRIPTION

The `pwd` command prints the current node path, the current node is produced by this command.

```
[/gadgets]% pwd
/gadgets
```

PARAMETERS

```
[-h | --help]
    Provides command usage
```

1.5.1.4. `ls` command

NAME

`ls` - list the content of a node

SYNOPSIS

```
ls [-h | --help] [-d | --depth] path
```

DESCRIPTION

The `ls` command displays the content of a node. By default it lists the content of the current node, however it accepts a path argument that can be absolute or relative.

```
[/]% ls
/
+-properties
| +-jcr:primaryType: nt:unstructured
| +-jcr:mixinTypes: [exo:owneable,exo:privilegeable]
| +-exo:owner: '__system'
| +-exo:permissions: [any read,*:/platform/administrators read,*:/platform/administrators add_node,*:/platform/administrators delete_node,*:/platform/administrators delete_node_and_children,*:/platform/administrators move_node,*:/platform/administrators move_node_and_children,*:/platform/administrators rename_node,*:/platform/administrators rename_node_and_children,*:/platform/administrators set_properties,*:/platform/administrators set_properties_and_permissions,*:/platform/administrators unset_properties,*:/platform/administrators unset_properties_and_permissions]
+-children
| +-/workspace
| +-/contents
| +-/Users
| +-/gadgets
| +-/folder
```

PARAMETERS

```
[-h | --help]
    Provides command usage

[-d | --depth]
    The depth of the printed tree

path
    The path of the node content to list
```

1.5.1.5. `cp` command

NAME

`cp` - copy a node to another

SYNOPSIS

```
cp [-h | --help] source target
```

DESCRIPTION

The `cp` command copies a node to a target location in the JCR tree.

```
[/registry]% cp foo bar
```

PARAMETERS

```
[-h | --help]
    Provides command usage

source
    The path of the source node to copy

target
    The path of the target node to be copied
```

1.5.1.6. mv command

```

NAME
    mv - move a node

SYNOPSIS
    mv [-h | --help] source target

DESCRIPTION
    The mv command can move a node to a target location in the JCR tree. It can be used also to rename a node.
    The mv command is a <Node,Node> command consuming a stream of node to move them and producing nodes that were moved.

    [/registry]% mv Registry Registry2

PARAMETERS
    [-h | --help]
        Provides command usage

    source
        The path of the source node to move, absolute or relative

    target
        The destination path absolute or relative

```

1.5.1.7. rm command

```

NAME
    rm - remove one or several node or a property

SYNOPSIS
    rm [-h | --help] ... paths

DESCRIPTION
    The rm command removes a node or property specified by its path either absolute or relative. This operation
    is executed against the JCR session, meaning that it will not be effective until it is committed to the JCR.

    [/]% rm foo
    Node /foo removed

    It is possible to specify several nodes.

    [/]% rm foo bar
    Node /foo /bar removed

    rm is a <Node,Void> command removing all the consumed nodes.

PARAMETERS
    [-h | --help]
        Provides command usage

    ... paths
        The paths of the node to remove

```

1.5.1.8. node command

```

NAME
    node add - creates one or several nodes

SYNOPSIS
    node [-h | --help] add [-t | --type] ... paths

DESCRIPTION
    The addnode command creates one or several nodes. The command takes at least one node as argument, but it can
    take more. Each path can be either absolute or relative, relative path creates nodes relative to the current
    node. By default the node type is the default repository node type, but the option -t can be used to specify an
    explicit node type.

    [/registry]% addnode foo
    Node /foo created

```



```
[/registry]% addnode -t nt:file bar juu
Node /bar /juu created
```

The addnode command is a <Void,Node> command that produces all the nodes that were created.

PARAMETERS

```
[-h | --help]
  Provides command usage

[-t | --type]
  The name of the primary node type to create.

... paths
  The paths of the new node to be created, the paths can either be absolute or relative.
```

NAME

node set - set a property on the current node

SYNOPSIS

```
node [-h | --help] set [-t | --type] propertyName propertyValue
```

DESCRIPTION

The set command updates the property of a node.

Create or destroy property foo with the value bar on the root node:

```
[/]% set foo bar
Property created
```

Update the existing foo property:

```
[/]% set foo juu
```

When a property is created and does not have a property descriptor that constraint its type, you can specify with the -t option

```
[/]% set -t LONG long_property 3
```

Remove a property

```
[/]% set foo
```

set is a <Node,Void> command updating the property of the consumed node stream.

PARAMETERS

```
[-h | --help]
  Provides command usage

[-t | --type]
  The property type to use when it cannot be inferred

propertyName
  The name of the property to alter

propertyValue
  The new value of the property
```

NAME

node import - imports a node from an nt file

SYNOPSIS

```
node [-h | --help] import source target
```

DESCRIPTION

Imports a node from an nt:file node located in the workspace:

```
[/]% importnode /gadgets.xml /
Node imported
```

PARAMETERS

```
[-h | --help]
    Provides command usage

source
    The path of the imported nt:file node

target
    The path of the parent imported node
```

```
NAME
    node export - export a node to an nt file

SYNOPSIS
    node [-h | --help] export source target

DESCRIPTION
    Exports a node as an nt file in the same workspace:

    [/% node export gadgets /gadgets.xml
    The node has been exported

PARAMETERS
    [-h | --help]
        Provides command usage

    source
        The path of the exported node

    target
        The path of the exported nt:file node
```

1.5.1.9. mixin command

```
NAME
    mixin add - add a mixin to one or several nodes

SYNOPSIS
    mixin [-h | --help] add mixin ... paths

DESCRIPTION
    The add command adds a mixin to one or several nodes, this command is a <Node,Void> command, and can
    add a mixin from an incoming node stream, for instance:

    [/% select * from mynode | mixin add mix:versionable

PARAMETERS
    [-h | --help]
        Provides command usage

    mixin
        the mixin name to add

    ... paths
        the paths of the node receiving the mixin
```

```
NAME
    mixin remove - removes a mixin from one or several nodes

SYNOPSIS
    mixin [-h | --help] remove ... paths

DESCRIPTION
    The remove command removes a mixin from one or several nodes, this command is a <Node,Void> command, and
    remove a mixin from an incoming node stream, for instance:

    [/% select * from mynode | mixin remove mix:versionable
```

PARAMETERS

```

[-h | --help]
  Provides command usage

... paths
  the paths of the node receiving the mixin

```

1.5.1.10. select command

NAME

select - execute a JCR sql query

SYNOPSIS

```
select [-h | --help] [-o | --offset] [-l | --limit] [-a | --all] ... query
```

DESCRIPTION

Queries in SQL format are possible via the `##select##` command. You can write a query with the same syntax by the specification and add options to control the number of results returned. By default the number of results returned is 5:

```

[/]% select * from nt:base
The query matched 1114 nodes
+--/
| +-properties
| | +-jcr:primaryType: nt:unstructured
| | +-jcr:mixinTypes: [exo:owneable,exo:privilegeable]
| | +-exo:owner: '__system'
| | +-exo:permissions: [any read,*/platform/administrators read,*/platform/administrators add_node,*/platform/administrators delete_node]
+--/workspace
| +-properties
| | +-jcr:primaryType: mop:workspace
| | +-jcr:uuid: 'a69f226ec0a80002007ca83e5845cdac'
...

```

Display 20 nodes from the offset 10:

```

[/]% select * from nt:base -o 10 -l 20
The query matched 1114 nodes
...

```

It is possible also to remove the limit of displayed nodes with the `-a` option (you should use this option with care for large results):

```

[/]% select * from nt:base -a
The query matched 1114 nodes
...

```

`select` is a `<Void,Node>` command producing all the matched nodes.

PARAMETERS

```

[-h | --help]
  Provides command usage

[-o | --offset]
  The offset of the first node to display

[-l | --limit]
  The number of nodes displayed, by default this value is equals to 5

[-a | --all]
  Display all the results by ignoring the limit argument, this should be used with care for large results

... query
  The query, as is

```

1.5.1.11. xpath command

NAME

xpath - execute a JCR xpath query

SYNOPSIS

```
xpath [-h | --help] [-o | --offset] [-l | --limit] [-a | --all] query
```

DESCRIPTION

Executes a JCR query with the xpath dialect, by default results are limited to 5. All results matched by t

PARAMETERS

[-h | --help]

Provides command usage

[-o | --offset]

The offset of the first node to display

[-l | --limit]

The number of nodes displayed, by default this value is equals to 5

[-a | --all]

Display all the results by ignoring the limit argument, this should be used with care for large resul

query

The query

1.5.1.12. commit command

NAME

commit - saves changes

SYNOPSIS

```
commit [-h | --help] path
```

DESCRIPTION

Saves the changes done to the current session. A node can be provided to save the state of the this nodes and its descendants only.

PARAMETERS

[-h | --help]

Provides command usage

path

The path of the node to commit

1.5.1.13. rollback command

NAME

rollback - rollback changes

SYNOPSIS

```
rollback [-h | --help] path
```

DESCRIPTION

Rollbacks the changes of the current session. A node can be provided to rollback the state of the this nodes and its descendants only.

PARAMETERS

[-h | --help]

Provides command usage

path

the path to rollback

1.5.1.14. version command

NAME

version checkin - checkin a node

SYNOPSIS

```
version [-h | --help] checkin path
```

DESCRIPTION

Perform a node checkin

PARAMETERS

[-h | --help]

Provides command usage

path

The node path to checkin

NAME

version checkout - checkout a node

SYNOPSIS

version [-h | --help] checkout path

DESCRIPTION

Perform a node checkout

PARAMETERS

[-h | --help]

Provides command usage

path

The node path to checkout

1.5.1.15. setperm command

NAME

setperm - modify the security permissions of a JCR node

SYNOPSIS

setperm [-h | --help] [-i | --identity] [-a | --add] [-r | --remove] ... paths

DESCRIPTION

The setperm commands configures the security of a node based on (see eXo JCR access control at <http://wiki.exoplatform.com/xwiki/bin/view/JCR/Access%20Control>). When a node is protected by access control mixin named `exo:privilegeable` that contains a `exo:permissions` property, for instance:

```
[/production]% ls
/production
+-properties
| +-jcr:primaryType: nt:unstructured
| +-jcr:mixinTypes: [exo:privilegeable]
| +-exo:permissions: [*/platform/administrators read,*/platform/administrators add_node,*/platform/administrators add_node]
+-children
| +-/production/app:gadgets
| +-/production/app:applications
| +-/production/mop:workspace
```

You can alter the node permission list with the setperm command:

```
[/production]% setperm -i */platform/mygroup -a read -a add_node /
Node /production updated to [read,add_node]
```

You can also remove a permission by using the `-r` option.

```
[/production]% setperm -i */platform/mygroup -r add_node /
Node /production updated to [read]
```

The setperm command will add automatically the `exo:privilegeable` mixin on the node when it is missing. The `<Node,Void>` command altering the security of the consumed node stream.

PARAMETERS

[-h | --help]

Provides command usage

[-i | --identity]

the identity

```
[-a | --add]
    the permissions to use

[-r | --remove]
    the permissions to remove

... paths
    The node path list to secure
```

1.5.2. SCP usage

Secure copy can be used to import or export content. The username/password prompted by the SSH server will be used for authentication against the repository when the import or the export is performed.

1.5.2.1. Export a JCR node

The following command will export the node `/gadgets` in the repository `portal-system` of the portal container `portal`:

```
scp -P 2000 root@localhost:portal:portal-system:/production/app:gadgets gadgets.xml
```

The node will be exported as `app_gadgets.xml`.

Note that the portal container name is used for GateIn. If you do omit it, then the root container will be used.

1.5.2.2. Import a JCR node

The following command will reimport the node:

```
scp -P 2000 gadgets.xml root@localhost:portal:portal-system:/production/
```

The exported file format use the JCR system view. You can get more information about that in the JCR specification.



Caution

The SCP feature is experimental

Chapter 2. Configuring the shell

CRaSH can be configured by tweaking various files of the CRaSH web archive

- *WEB-INF/crash/crash.properties*
- *WEB-INF/crash/commands/base/login.groovy*
- *WEB-INF/crash/commands/jcr/login.groovy*



Note

Configuration happens via the CRaSH war file and can be overridden from the JVM system properties by using the same property name.

2.1. Change the SSH server key

The key can be changed by replacing the file *WEB-INF/sshd/hostkey.pem*. Alternatively you can configure the server to use an external file by using the *crash.ssh.keypath* parameter in the *crash.properties*. Uncomment the corresponding property and change the path to the key file.

```
#crash.ssh.keypath=/path/to/the/key/file
```

2.2. Change the ports of the telnet or SSH server

The ports of the server are parameterized by the *crash.ssh.port* and *crash.telnet.port* parameters in the *crash.properties* file

```
# SSH configuration
crash.ssh.port=2000
```

```
# Telnet configuration
crash.telnet.port=5000
```

2.3. Remove the telnet or SSH access

- to remove the telnet access, remove the jar file in the *WEB-INF/lib/crsh.shell.telnet-1.0.0-beta22.jar* .
- to remove the SSH access, remove the jar file in the *WEB-INF/lib/crsh.shell.ssh-1.0.0-beta22.jar* .

2.4. Configure the shell default message

The *login.groovy* file contains two closures that are evaluated each time a message is required

- The `prompt` closure returns the prompt message

- The `welcome` closure returns the welcome message

Those closure can be customized to return different messages.

Chapter 3. Extending the shell

3.1. Groovy command system

The shell command system is based on the [Groovy](#) language and can easily be extended.

3.1.1. Groovy file

Each command has a corresponding Groovy file that contains a command class that will be invoked by the shell. The files are located in the `/WEB-INF/crash/commands` directory and new files can be added here.

New commands can directly be placed in the commands directory however they can also be placed in a sub directory of the command directory, which is useful to group commands of the same kind.

In addition of that there are two special files called `login.groovy` and `logout.groovy` that are executed upon login and logout of a user. They are useful to setup and cleanup things related to the current user session.

3.1.2. Groovy execution

When the user types a command in the shell, the command line is parsed by the `cmdline` framework and injected in the command class. Previously the `args4j` framework was used but this framework does not support natively code completion and could not be extended to support it. The support of command line completion is the main motivation of the development of such a framework. To learn more, the best is to study the existing commands as the framework is quite easy to use, the following features are supported:

- Annotation based framework
- Provide accurate contextual code completion
- Support sub commands (? la git like "git add") for grouping commands of the same kind inside the same class as methods
- Advanced support for usage and manual

A simple example, the `sleep 1` command pauses the shell for one second, let's briefly study its code:

```
class sleep extends CRaSHCommand {
    @Usage("sleep for some time")
    @Command
    Object main(@Usage("Sleep time in seconds") @Argument int time) throws ScriptException {
        if (time < 0)
            throw new ScriptException("Cannot provide negative time value $time");
        Thread.sleep(time * 1000);
        return null;
    }
}
```

The `@Usage` annotation gives short information about the command itself, another annotation is available for documenting more formally the command: `@Man` but it is not used in this example.

The `@Command` tags the `main(...)` method as a command method. Any number of method can be tagged as such, providing a convenient way to pack commands of the same kind. By default the `main` is a special convention indicating that executing the command should not require to explicitly use the main.

The `@Argument` annotation describes the command unique argument that is the time to sleep. The same `@Usage` annotation is used again to describe briefly the argument. Again it could be possible to use the `@Man` annotation.

3.1.3. Shell context

A command is a Groovy object and it can access or use the contextual variables. A few variables are maintained by the shell and should be considered with caution. The shell also provides a few functions that can be used, those functions defined in *login.groovy*

Chapter 4. Developers

4.1. settings.xml (maven)

Crash system properties :

```
<properties>  
  <crsh.deploy.directory>/path/to/webapp/directory</crsh.deploy.directory>  
  <crsh.appengine.sdk>/path/to/appengine/sdk</crsh.appengine.sdk>  
</properties>
```

Chapter 5. Hey, I want to contribute!

Drop me an email (see my @ on www.julienviet.com), any kind of help is welcome.