Sun N1 Service Provisioning System 5.2 Command-Line Interface Reference Manual

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–4444–10
April 2006

060331@14558

Contents

Preface

This book, *Sun N1 Service Provisioning System 5.2 Command-Line Interface Reference Manual*, provides information about the commands that you can run to manage hosts, users, components, and plans in the Sun N1 Service Provisioning System software.

Who Should Use This Book

This book is for people who want to use the Sun N1™ Service Provisioning System to manage hosts, users, and to install and manage applications in data centers.

How This Book Is Organized

This book contains an overview of the command-line interface, chapters that describe the commands associated with managing the provisioning software, and an appendix that describes the format of input types.

Related Books

The Sun N1 Service Provisioning System documentation includes these other books:

- *Sun N1 Service Provisioning System 5.2 Release Notes*
- *Sun N1 Service Provisioning System 5.2 Installation Guide*
- *Sun N1 Service Provisioning System 5.2 System Administration Guide*
- *Sun N1 Service Provisioning System 5.2 Operations and Provisioning Guide*
- *Sun N1 Service Provisioning System 5.2 Plan and Component Developer's Guide*
- *Sun N1 Service Provisioning System 5.2 XML Schema Reference Guide*
- *Sun N1 Service Provisioning System 5.2 Plug-In Developer's Guide*

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX® system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

◆ ◆ ◆ CHAPTER 1

Using the Command-Line Interface

This chapter describes the command-line interface (CLI) for the Sun N1 Service Provisioning System software, which enables you to submit commands to the master server from another server. You can run the CLI from a shell prompt, a DOS prompt, or a script. You can also use the CLI as an alternative to the browser interface to access the master server. Before you can use the CLI, you must first install the Command-Line Interface Client on the local server.

The CLI can be run in these two modes:

- **Single-line –** This mode requires that you type complete commands and does not prompt you for missing information.
- **Interactive –** This mode initiates a login session in which you can run one or more commands. When you initiate a login session in this way, you only need to authenticate once. This mode depends on the Jython programming language already being installed on the server. See the *Sun N1 Service Provisioning System 5.2 Installation Guide*.

  The Jython programming language is a Java™ implementation of the object-oriented Python language. You must install Jython on any system on which you plan to install the Command-Line Interface Client, see `http://www.jython.org`.

This chapter covers the following topics:

CLI Command Structure

The `cr_cli` command has the following syntax:

`cr_cli -cmd` *command common-arguments authentication-arguments* `[` *other-arguments* `]`

Note that the *common-arguments* must be used in the position shown above , before the *authentication arguments* and *other-arguments*. The following table lists the *common-arguments* supported by all commands.

TABLE 1–1 Using the *common arguments*

Argument		Syntax	Description
o	Optional	String	The name of the formatter
of	Optional	String	The name of the output file
h	Optional	String	Command help

Note that most commands must use the authentication arguments: either `-u` and `-p`, or `-s`. To determine whether a command requires the authentication arguments, see the description of the command. Also, the authentication arguments do not have a fixed position on the command line, but they must appear after the `-cmd`, `-h`, `-o`, and `-of` arguments.

`-cmd` *command*

Name of the command to run.

All CLI commands use the following format:

subsystem`.`*object*`.`*command*

For example, the command for adding a host to the database is `hdb.h.add`.

subsystem is the name of the subsystem, such as `hdb` for the host database commands.

object is the object that the command affects, such as `h` for a host.

command is the action that the command performs, such as `add` that adds a host to the database.

authentication-arguments

Most of the CLI commands require authentication, however, you do not need to authenticate to get help or to list the available commands.

`-u` *user-name*

User name to use for authentication. To authenticate, you must also specify a password for the specified user.

As an alternative to supplying the user name and password, you can specify a session ID.

`-p` *password* — Password that is used to authenticate the user specified by `-u`.

A password that you specify on the command line is not secure, as it can appear in process lists and in a shell's command-line history. So, to keep your password secure, store your user name and password in a file on the local file system and refer to that file as input to your command. See the first example in Example 1–1. Make sure that the user owns the file and that the file is only readable by that user.

`-s` *session-ID* — Session ID that is used to authenticate a session.

For information about how to get the session ID and use it for authentication, see "Reading Input From a File" on page 19.

As an alternative to supplying the session ID, you can specify a user name and password by using the `-u` and `-p` arguments.

other-arguments — Arguments and values that are associated with *command*.

The `cr_cli` command returns 0 on success and 1 on failure.

EXAMPLE 1–1 Using the CLI

The following are some examples of the provisioning software's CLI:

- This example shows how to read the user name and password from a file to authenticate the command. The file, `.terrypw`, contains the user name and password for Terry in the following format:

  ```
  -u
  terry
  -p
  securepasswd
  ```

  To authenticate the `hdb.h.lo` command, Terry runs the following command:

  ```
  cr_cli -cmd hdb.h.lo exp:.terrypw
  ```

  Store the file on a local file system and use the file permissions to restrict access to the file. For more information, see "Reading CLI Arguments From a File" on page 20.

EXAMPLE 1–1 Using the CLI *(Continued)*

- This example shows how user, `terry`, retrieves information about the host, `barolo7`. This command supplies the password directly on the command line.

  ```
  cr_cli -cmd hdb.h.lo -ID NM:barolo7 -u terry -p password
  ```

  Note that specifying the password in this way is insecure.
- This example shows how to pass arguments that include spaces by enclosing the string in quotes. In this example, user `terry` modifies the description of the component named `myWebServer`:

  ```
  cr_cli -cmd cdb.c.mod -comp myWebServer
    -desc "Version 3.7 of My Web Server" -u terry -p 123xyz
  ```

  On UNIX® systems, you can escape each space with a backslash character (\).

  ```
  cr_cli -cmd cdb.c.mod -comp myWebServer -u terry -p 123xyz
    -desc Version\ 3.7\ of\ My\ Web\ Server
  ```

Using `NM:` to Perform ID Substitution

Most of the objects that you create are associated with ID numbers. An ID number is a unique identifier for an object in the repository, such as a user account or a component.

While ID numbers are useful, they can be cumbersome to use. To use names rather than ID numbers, introduce the name of an object by using the `NM:` notation.

For a complete list of the supported `NM:` mappings, see Appendix A.

For example, the following syntax is used to represent object IDs, such as hosts, user names, user group names, and host type names:

`NM:`*host*
`NM:`*user-name*
`NM:`*user-group-name*
`NM:`*host-type-name*

You can also use the `NM:` syntax to identify components and plans by name and optional version number:

`NM:`*plan-name*`[:`*version*`]`

The following are some sample uses of this notation:

```
NM:simplePlan
NM:simplePlan:1.0
NM:/foo/bar/simplePlan:1.1
```

If a version number is not specified, the provisioning software uses the latest version.

CLI Input and Output

The following sections describe how to use files as input to the CLI and how to use files to store output from the CLI commands.

Redirecting Output to a File

To redirect command output to a file, you can either use the `-of` argument or use a shell redirect. The argument to the `-of` argument is a full path to a file.

For example, this command writes the output to a file called `hostFile`. Note that the `-of` argument must immediately follow the command specified by `-cmd`.

```
cr_cli -cmd hdb.h.add -of hostFile -u user-name
-p password -name myhost -tID NM:system#crhost
```

After the command is run, `hostFile` contains output in `detail` format, which is the default output format for the `hdb.h.add` command.

Another example of redirecting output to a file involves using a stored session ID to authenticate commands. First, save the RPC-serialized representation of the session ID to a file called `session`. You can redirect this output to a file using either method.

- Create a file using a shell redirect

  ```
  cr_cli -cmd udb.login -o serialized -u user-name -p password > session
  ```

- Create a file using the `-of` argument

  ```
  cr_cli -cmd udb.login -o serialized -of session -u user-name -p password
  ```

Once you have stored your session ID in a file, you can use the file as input to another command. See "Reading Input From a File" on page 19.

Reading Input From a File

To pass data from a file to a command, identify the file by adding the prefix `file:` to the file name.

For example, if you have stored a session ID in a file and want to use the file to authenticate the `hdb.h.la` command, use the file as input to the command.

```
cr_cli -cmd hdb.h.la -s file:session
```

If you did not store the session id to a file , you can pass a serialized version of the session object directly to another command by using the `ser:` prefix. In the example below, the serialized session object obtained by using udb.login command with the with the `-o serialized` argument, is available in `sessionStr` .

```
cr_cli -cmd hdb.h.la -s ser:sessionStr
```

Reading CLI Arguments From a File

To read a CLI argument from an input file, identify the input file by prefixing `exp:` to the file name. First, create a file that contains the information you want to pass to the CLI. Note that each argument must be listed on a separate line and must appear in the order required by the command.

Then, have the command get the arguments from the file.

For example, these files, `file1.txt` and `file2.txt`, can be used in conjunction with `exp:` to specify command arguments.

`file1.txt` contains the following:

```
hdb.h.la
-u
user-name
exp:file2.txt
```

`file2.txt` contains the following:

```
-p
password
```

To execute the command that is described by these files, type the following:

```
cr_cli -cmd exp:file1.txt
```

Output Formats of CLI Commands

You can adjust the output format a CLI command. To specify the output format for a CLI command, the `-o` argument must immediately follow the command you specified by `-cmd`.

For example, to specify the `string` output format for the `hdb.h.la` command, type the following:

```
cr_cli -cmd hdb.h.la -o string -u user-name -p password
```

These are the standard output formats that are available to all CLI commands:

`raw`	Produces an internal string representation of the result that depends on the command.
`string`	Produces brief output.
`serialized`	Produces XML serialized text output.

	Use this argument when you want to pass information to another script. Do not depend on the structure of this format, as it could change.
`sink`	Discards output. On UNIX systems, output is redirected to `/dev/null`.
`detail`	Shows detailed information. Note that this format is not available for all commands.

If you do not specify an output format, the command uses its default output format. To see the default output format for a command, run the following:

```
cr_cli -cmd command -h
```

Converting the Output Format

Use the `reformat.util` command to convert the output format of a command that has been written to a file. The `reformat.util` command reads data from the specified output file and reformats it by using the specified output format.

For example, you created a host and stored the output from the `hdb.h.add` command in `serialized` format in a file named `hostFile`:

```
cr_cli -cmd hdb.h.add -o serialized -u user-name -p password -name myhost
-tID NM:roxhost > hostFile
```

Use the `util.reformat` command to reformat the output to be the `hdb.detail` format.

```
cr_cli -cmd util.reformat -o hdb.detail -u user-name -p password
-self file:hostFile
```

The previous command results in the following output:

```
ID: 010010000204-1027365659275-00170-1199101891
Name: myhost
Description:
Virtual: false
Hidden: false
Type ID: 010010001024-0000000000000-00001-0000000004
Attributes:
<Table is empty>
Applications:
<Table is empty>
```

Running the Command-Line Interface

To execute one CLI command at a time, use the `cr_cli` tool. This tool invokes the CLI in single-line command mode.

Single-line command mode accepts one command at a time as input. Each command submitted must be complete, you are not interactively prompted for the next input parameter. When operating in this mode, the Command Line Execution Client does not maintain a command history.

`cr_cli` commands can be stored in a file and called from a shell script. This feature is useful for repetitive tasks such as running execution plans, comparisons, or populating hosts.

The interactive command-line mode uses the Jython interpreter as its shell. When operating in this mode, the CLI offers you these advantages:

- You can take advantage of the command history that is stored by the shell.
- You can call the provisioning software commands from a Jython script.
- You can create more powerful scripts for complex, repetitive operations.

How to Run the Single-Command CLI

From a server where the CLI Client is installed, type the command.

`./cr_cli -cmd` *subsystem.object.command* **`-u`** *user* **`-p`** *password*

How to Run the CLI Jython Interpreter

1. **From a server where the CLI Client and Jython are installed, start the CLI Jython Interpreter.**

   ```
   ./cr_clij
   ```

2. **Include the following code at the beginning of your script.**

   ```
   from clui import *
   app=PyCLUI()
   app.execStr(CLI command)
   app.close()
   ```

   The assignment `app=PyCLUI()` calls the CLI. The `App.close()` call deletes the instance of this Jython class.

Note – Only the execStr and execRaw calls are supported for SPS scripts with Jython. The returned objects from execRaw can only be used as parameters for other execRaw calls. If you attempt to use an object returned from an execRaw call as a parameter to an execStr call, the call fails. You can use the toString() method of the object to construct the string to feed to the execStr call.

Getting Command-Line Help

To get general help about the CLI, type the following:

```
cr_cli -help
```

To get help for a specific command, type the following:

```
cr_cli -cmd command -h
```

The command descriptions use the following notation to indicate whether an argument is required or optional:

`Optional`	The argument is optional.
`Required`	The argument is required.
`[O/R]`	The argument is usually optional, but might be required in some cases. In such situations, the help indicates when the argument is required.

The following is the help information for the `hdb.h.add` command:

```
-u [O/R]: The user username for authentication: String
-p [O/R]: The user password for authentication: String
-s [O/R]: The session ID for authentication: SessionID
-name: The host name: String
-desc Optional: The host description: String
-tID: The ID of the host type: HostTypeID
-attr [O/R]: The host attributes; required if the host type requires
  them: Hashtable
```

In this example, the only required arguments are `-name`, `-tID`, and the authentication arguments: either `-u` and `-p`, or `-s`. If the host type of the new host requires that host attributes be specified, the `-attr` argument is also required.

Note that help you get by using the `-h` does not use the `Required` notation to identify required arguments. Required arguments are shown without any notation.

Listing the CLI Commands

To list all of the available commands, type the following:

```
cr_cli -cmd -l
```

To list all of the commands that have the same command prefix, use the wildcard character * after the prefix. You might need to escape the * with a backslash (\) or by enclosing it in double quotes. For example, the following command lists all of the `hdb` commands:

```
cr_cli -cmd -l "hdb.*"
```

The CLI commands are grouped in the following categories:

- `cat` – Categories
- `cdb` – Component database
- `cfg` – Configuration generator
- `cmp` – Comparison engine
- `fdb` – Folder repository
- `hdb` – Host repository
- `net` – Network operations
- `pdb` – Plan repository
- `pe` – Plan execution
- `plg` – Plug-in repository
- `rule` – Rules for notifications
- `udb` – User repository
- `util` – Miscellaneous utilities

◆ ◆ ◆ CHAPTER 2

cat: Commands for Managing Categories

This chapter describes the commands that you need to use to manage categories.

You can use categories to classify:

- Plans
- Components
- Comparisons

Overview of the `cat` Commands

The following table summarizes the CLI commands available for managing categories.

TABLE 2–1 Summary of cat Commands

Command	Description
cat.add	Adds a new category.
cat.del	Deletes the specified category
cat.mod	Modifies a category.
cat.la	Lists all the categories that have been defined.

`cat.add`

This command adds a new category to the repository. The category is given a name and a description.

TABLE 2–2 Arguments and Result for the cat.add Command

Argument/Result		Syntax	Description
name	Required	String	The name of the Category
desc	Required	String	The description of the Category
result		Category	The new Category object

cat.del

This command deletes the specified category. Deleting a category does not delete the objects, such as components and plans, in the category.

TABLE 2–3 Argument for the cat.del Command

Argument		Syntax	Description
ID	Required	CategoryID	The Category instance ID

cat.mod

This command modifies the specified category. Omitted arguments are overwritten.

TABLE 2–4 Arguments and Result for the cat.mod Command

Argument/Result		Syntax	Description
ID	Required	CategoryID	The Category instance ID
name	Optional	String	The new name of the Category
desc	Optional	String	The new description of the Category
result		Category	The modified Category

cat.la

This command lists all categories defined in the Sun N1 Service Provisioning System software.

TABLE 2–5 Result for the cat.la Command

Result	Syntax	Description
result	CategoryArray	The category instances

◆ ◆ ◆ CHAPTER 3

cdb: CLI Commands for Managing Components

This chapter describes commands that you need to use to manage components and check-in jobs.

Overview of the cdb Commands

The CLI includes the following sets of commands for managing components.

TABLE 3–1 Sets of Commands for Managing Components

CLI Prefix	Description of Command Set
cdb.c	Commands for managing components
cdb.ic	Commands for retrieving information about installed components.
cdb.vs	Commands for managing variable settings objects.
cdb.ssr	Commands for managing system service ref objects.
cdb.ctr	Commands for managing component type ref objects.
cdb.rsrc	Commands for managing browsable components.

TABLE 3–1 Sets of Commands for Managing Components *(Continued)*

CLI Prefix	Description of Command Set
cdb.cj	Commands for controlling and monitoring component check-in jobs.

This chapter describes all the commands in each of these sets.

cdb.c: Managing Components

The `cdb.c` commands provide general-purpose controls for managing components.

TABLE 3–2 CLI Commands for Managing Components

Command	Description
cdb.c.ci	Checks in non-browsable components and component models.
cdb.c.co	Checks out a component.
cdb.c.la	Lists all versions of all components.
cdb.c.lo	Lists detailed information about a component.
cdb.c.lv	Lists all versions of a component.
cdb.c.mod	Modifies a component.
cdb.c.mv	Moves or renames a component.
cdb.c.sc	Applies one or more categories to a component.
cdb.c.sh	Shows or hides a component.
cdb.c.del	Deletes a component

cdb.c.ci

Use the `cdb.c.ci` command to check in certain components. You will need to use this command in the following scenarios.

- You need to check in a non-browsable component (for example: an `untyped` or `container` component).

  If you need to check in a browsable component, use the `cdb.rsrc.ci` command. For a description of the command and its arguments, see "cdb.rsrc.ci" on page 45.
- You need to check in a component model (the XML representation of a component), but not the referenced components or source objects. You might use this functionality when updating control blocks and variable values.

If you want to create new versions of the component and all of its referenced components, use the `cdb.rsrc.ci` command. For a description of the command and its arguments, see "cdb.rsrc.ci" on page 45.

TABLE 3–3 Arguments and Result for the cdb.c.ci Command

Argument/Result		Syntax	Description
path	Required	InputStreamWrapper	The location of the XML component definition
major	Optional	Boolean	Whether to checkin as a new major version; default false.
import	Optional	Boolean	Whether to import variable settings, default true
hidePrev	Optional	Boolean	Whether to hide the previous component, default true.
parents	Optional	Boolean	Whether nonexistent parent folders are created during component check-in. Default is false.
result		Component	The new component

cdb.c.co

This command checks out a component. It outputs the specified component in XML format.

TABLE 3–4 Argument and Result for the cdb.c.co Command

Arguments/Result		Syntax	Description
comp	Required	ComponentID	The ID of the component XML to view.
result		Component	The component in XML format

cdb.c.la

The command lists all versions of all components.

TABLE 3–5 Arguments and Result for the cdb.c.la Command

Arguments/Result		Syntax	Description
sh	Optional	Boolean	Whether hidden components are shown, default false
cat	Optional	CategoryID	Category filter to apply, default "all"
folderID	Optional	FolderID	Parent folder ID; default is the root folder (NM:/)
flatView	Optional	Boolean	Whether results should be displayed in flat view; default is true
result		SummaryComponent-Array	The components

cdb.c.lo

This command lists the details of a specified component.

TABLE 3–6 Arguments and Result for the cdb.c.lo Command

Argument/Result		Syntax	Description
ID	Required	ComponentID	The ID of the component to view
result		Component	The component

cdb.c.lv

This command lists all the versions of the specified component.

TABLE 3–7 Argument and Result for the cdb.c.lv Command

Argument/Result		Syntax	Description
comp	Required	ComponentID	The component
result		SummaryComponent-Array	All the versions of the component

cdb.c.mod

This command modifies a component, which results in a new version of the component.

TABLE 3–8 Arguments and Result for the cdb.c.mod Command

Argument/Result		Syntax	Description
comp	Required	ComponentID	The component
label	Optional	String	The component label
desc	Optional	String	The component description
rva	Optional	StringArray	The component versions; use version number, "#" for recommended, "+" for default, and "-" for latest; or omit this argument to use latest for all components. This argument is only applicable for composite components.
hidePrev	Optional	Boolean	Whether to hide the previous version. Default is true.
result		Component	The component

cdb.c.mv

This command moves or renames a component.

TABLE 3–9 Arguments for the cdb.c.mv Command

Argument		Syntax	Description
ID	Required	ComponentID	The ID of the component to move or rename
fullname	Required	String	The new full name (path + name) of the component

cdb.c.sc

This command associates a component with a set of categories.

TABLE 3–10 Arguments and Result for the cdb.c.sc Command

Argument/Result		Syntax	Description
ID	Required	ComponentID	The ID of the component to affect
catIDs	Required	CategoryIDSet	The IDs of the Categories to associate with this component
all	Optional	Boolean	Whether to change all versions of the component, default false

cdb.c.sh

This command shows or hides a component.

TABLE 3–11 Arguments for cdb.c.sh

Argument		Syntax	Description
ID	Required	ComponentID	The ID of the component to hide or show
hide	Required	Boolean	Whether the component is set to hidden
all	Optional	Boolean	Whether to change all versions of the component, default false

cdb.c.del

This command deletes a component.

TABLE 3–12 Arguments for the cdb.c.del Command

Argument		Syntax	Description
ID	Required	ComponentID	The ID of the component to delete

TABLE 3–12 Arguments for the cdb.c.del Command *(Continued)*

Argument		Syntax	Description
all	Optional	Boolean	Whether to delete all versions of the component, default false

cdb.ic: Managing Installed Components

The `cdb.ic` commands retrieve information about components that are already installed on hosts.

TABLE 3–13 CLI Commands for Managing Installed Components

Command	Description
cdb.ic.lbc	Lists all the hosts on which a component is installed.
cdb.ic.lbh	Lists all the components installed on a specific host.
cdb.ic.ldo	Lists dependencies on other components
cdb.ic.lod	Lists the components that are dependent upon this component
cdb.ic.vs.lo	Lists details of the specified generated variable settings object.

cdb.ic.lbc

This command lists all the hosts on which a particular component is installed.

TABLE 3–14 Argument and result for the cdb.ic Command

Argument/Result		Syntax	Description
comp	Required	ComponentID	The component ID
result		InstalledComponent-BeanArray	The installed components

cdb.ic.lbh

This command lists all the components installed on a particular host.

TABLE 3–15 Argument and Result for the cdb.ic.lbh Command

Argument/Result		Syntax	Description
host	Required	HostID	The host ID
cat	Optional	CategoryID	Category filter to apply; default all
result		InstalledComponent-BeanArray	The installed components

cdb.ic.ldo

This command lists a component's dependencies.

TABLE 3–16 Argument and Result for the cdb.ic.ldo Command

Argument/Result		Syntax	Description
ID	Required	InstalledComponentID	The installed component ID
result		DependencyArray	The dependencies on other components

cdb.ic.lod

This command lists the components that are dependent on this component.

TABLE 3–17 Argument and Result for the cdb.ic.lod Command

Argument/Result		Syntax	Description
ID	Required	InstalledComponentID	The installed component ID
result		DependencyArray	The components dependent on this component

cdb.ic.vs.lo

This command lists details of a particular generated variable settings object..

TABLE 3–18 Argument and Result for the cdb.ic.vs.lo Command

Argument/Result		Syntax	Description
ID	Required	InstalledComponentID	The ID of the installed component whose generated variable settings will be viewed.
result		GeneratedVariable-Settings	The generated variable settings

cdb.vs: Managing Variable Settings

The `cdb.vs` commands manage variable settings for components.

TABLE 3–19 CLI Commands for Managing Variable Settings

Command	Description
cdb.vs.add	Adds a new variable settings object.
cdb.vs.del	Deletes a variable settings object.
cdb.vs.imp	Imports a variable settings object from one component into another.
cdb.vs.la	Lists all the variable settings objects associated with a specific component.
cdb.vs.lo	Lists the details of a specific variable settings object.
cdb.vs.mod	Modifies a variable settings object.

cdb.vs.add

This command adds a new variable settings object

TABLE 3–20 Arguments and Result for the cdb.vs.add Command

Argument/Result		Syntax	Result
comp	Required	ComponentID	The component
name	Required	String	The new name
vars	Required	Hashtable	The new override values

TABLE 3–20 Arguments and Result for the cdb.vs.add Command *(Continued)*

Argument/Result	Syntax	Result
result	ComponentVariable-Settings	The new component variable settings

cdb.vs.del

This command deletes an existing variable settings object.

TABLE 3–21 Argument for the cdb.vs.del Command

Argument		Syntax	Description
vs	Required	ComponentVariable-SettingsID	The ID of the component variable settings to delete

cdb.vs.imp

This command imports variable settings from one component into another.

TABLE 3–22 Arguments for the cdb.vs.imp Command

Argument		Syntax	Description
src	Required	ComponentID	The component to import variable settings from.
dst	Required	ComponentID	The component to import variable settings to

cdb.vs.la

This command lists all variable settings objects associated with a particular component.

TABLE 3–23 Argument and Result for the cdb.vs.la Command

Argument/Result		Syntax	Description
comp	Required	ComponentID	The component.
result		ComponentVariable-SettingsArray	The component variable settings.

cdb.vs.lo

This command lists details of a particular variable settings object.

TABLE 3–24 Argument and Result for the cdb.vs.lo Command

Argument/Result		Syntax	Description
vs	Required	ComponentVariable-Settings	The component variable settings to view
result		ComponentVariable-Settings	The component variable settings

cdb.vs.mod

This command modifies an existing variable settings object

TABLE 3–25 Arguments and Result for the cdb.vs.mod Command

Argument/Result		Syntax	Description
vs	Required	ComponentVariable-Settings	The component variable settings
name	Optional	String	The new name
vars	Optional	Hashtable	The new override values
result		ComponentVariable-Settings	The modified component variable settings

cdb.ssr: System Service Ref Commands

TABLE 3–26 CLI Commands for System Service Ref

Command	Description
cdb.ssr.add	Adds a system service ref
cdb.ssr.mod	Modifies an existing system service ref; omitted arguments are overwritten
cdb.ssr.del	Deletes a system service ref
cdb.ssr.lo	Retrieves a system service ref
cdb.ssr.la	Lists all system service refs

cdb.ssr.add

This command adds a new system service ref.

TABLE 3–27 Argument and Result for the cdb.ssr.add Command

Argument/Result		Syntax	Description
name	Required	String	The system service ref name
desc	Optional	String	The system service ref description
icn	Required	String	The name of the referenced installed component
icv	Required	String	The version of the referenced installed component
icp	Optional	String	The install path of the referenced installed component
result		SystemServiceRef	The new system service ref

cdb.ssr.mod

This command modifies an existing system service ref; omitted arguments are overwritten.

TABLE 3–28 Argument and Result for the cdb.ssr.mod Command

Argument/Result		Syntax	Description
ssr	Required	SystemServiceRef	The target system service ref
name	Optional	String	The system service ref name
desc	Optional	String	The system service ref description
icn	Optional	String	The name of the referenced installed component

TABLE 3–28 Argument and Result for the cdb.ssr.mod Command *(Continued)*

Argument/Result		Syntax	Description
icv	Optional	String	The version of the referenced installed component
icp	Optional	String	The install path of the referenced installed component
result		SystemServiceRef	The modified system service ref

cdb.ssr.del

This command deletes a system service ref..

TABLE 3–29 Argument and Result for the cdb.ssr.del Command

Argument/Result		Syntax	Description
ID	Required	SystemServiceRefID	The system service ref ID

cdb.ssr.lo

This command retrieves a system service ref.

TABLE 3–30 Argument and Result for the cdb.ssr.lo Command

Argument/Result		Syntax	Description
ID	Required	SystemServiceRef	The target system service ref
result		SystemServiceRef	The system service ref

cdb.ssr.la

This command lists all system service refs.

TABLE 3–31 Argument and Result for the cdb.ssr.lo Command

Argument/Result	Syntax	Description
result	SystemServiceRefArray	The system service refs

cdb.ctr: Component Type Commands

TABLE 3–32 CLI Commands for Component Type Ref

Command	Description
cdb.ctr.add	Adds a new component type ref
cdb.ctr.mod	Modifies an existing component type ref; omitted arguments are overwritten
cdb.ctr.del	Deletes a component type ref
cdb.ctr.lo	Retrieves a component type ref
cdb.ctr.la	Lists all component type refs

cdb.ctr.add

This command adds a new component type ref..

TABLE 3–33 Argument and Result for the cdb.ctr.add Command

Argument/Result		Syntax	Description
name	Required	String	The component type ref name
desc	Optional	String	The component type ref description
order	Required	String	The component type ref order
group	Required	String	The component type ref group
indentLevel	Required	String	The component type indent level
compref	Required	String	The name of the component ref within the component type ref

TABLE 3–33 Argument and Result for the cdb.ctr.add Command *(Continued)*

Argument/Result		Syntax	Description
compver	Required	String	The version of the component ref within the component type ref
result		ComponentTypeRef	The new component type ref

cdb.ctr.mod

Modifies an existing component type ref; omitted arguments are overwritten.

TABLE 3–34 Argument and Result for the cdb.ctr.mod Command

Argument/Result		Syntax	Description
ctr	Required	ComponentTypeRef	The target component type ref
name	Optional	String	The component type ref name
desc	Optional	String	The component type ref description
order	Optional	String	The component type ref order
group	Optional	String	The component type ref group
indentLevel	Optional	String	The component type indent level
compver	Optional	String	The version of the component ref within the component type ref
result		ComponentTypeRef	The modified component type ref

cdb.ctr.del

This command deletes a component type ref.

TABLE 3–35 Argument and Result for the cdb.ctr.del Command

Argument/Result		Syntax	Description
ID	Required	ComponentTypeRefID	The component type ref ID

cdb.ctr.lo

This command retrieves a component type ref.

TABLE 3–36 Argument and Result for the cdb.ctr.lo Command

Argument/Result		Syntax	Description
ID	Required	ComponentTypeRef	The target component type ref
result		ComponentTypeRef	The component type ref

cdb.ctr.la

This command lists all component type refs.

TABLE 3–37 Argument and Result for the cdb.ctr.la Command

Argument/Result	Syntax	Description
result	ComponentTypeRef-Array	The component type refs

cdb.rsrc: Managing Components

The `cdb.rsrc` commands provide general-purpose controls for managing components.

TABLE 3–38 CLI Commands for Managing Components

Command	Description
cdb.rsrc.ci	Checks in certain components and their resources to the repository.
cdb.rsrc.cib	Checks in all the components listed in a batch file.
cdb.rsrc.co	Checks out the specified component.

TABLE 3–38 CLI Commands for Managing Components *(Continued)*

Command	Description
cdb.rsrc.gd	Generates a resource descriptor.
cdb.rsrc.rci	Rechecks in a component.
cdb.rsrc.showopts	Shows the check-in options that are supported by a particular type.

cdb.rsrc.ci

Use the `cdb.rsrc.ci` command to check in certain components and their source objects. You will need to use this command in the following scenarios.

- You need to check in a browsable component (for example: `file` or `Weblogic EJB`)

  If you want to check in a non-browsable component, use the `cdb.c.ci` command. For a description of the command and its arguments, see "cdb.c.ci" on page 30.
- You need to check in source objects for a simple component.
- You need to check in the referenced components of a browsable, composite component.

Each invocation of the `cdb.rsrc.ci` command is considered a "check-in job," and can be managed with the CLI commands for managing check-in jobs. For example, to determine which `cdb.rsrc.ci` commands are running, you can run the `cdb.cj.la` command, which lists all the current check-in jobs. You can also pass `compCheckInID` value returned by `cdb.rsrc.ci` as an argument to `cdb.cj.lo` to get status information about a specific check-in job.

TABLE 3–39 Arguments and Result for the cdb.rsrc.ci Command

Argument/Result		Syntax	Description
src	Required	String	The local file/directory being checked in
dst	Required	String	Which component name to check in as
type	Required	String	The type of the component
platform	Optional	HostSetID	The platform of the component
desc	Optional	String	A description of the component

TABLE 3–39 Arguments and Result for the cdb.rsrc.ci Command *(Continued)*

Argument/Result		Syntax	Description
major	Optional	Boolean	Whether the version increment should be major or minor, default false
config	Optional	Boolean	Whether the component is a config file; the default is false
hidePrev	Optional	Boolean	Whether to hide the latest component; the default is true
includeOwners	Optional	Boolean	Whether to include owner information; the default is true
includeGroups	Optional	Boolean	Whether to include group information; the default is true
addTo	Optional	Boolean	Whether the files being checked in should be added to the existing files to create a new version of the component, instead of completely replacing the existing files to create a new component
hostID	Optional	HostID	The ID of the local host
redun	Optional	Boolean	Whether redundancy checking should apply; the default is true
pickerName	Optional	String	The name of the component picker to use (defaults to null for the default picker).

TABLE 3–39 Arguments and Result for the cdb.rsrc.ci Command *(Continued)*

Argument/Result		Syntax	Description
extraOpts	Optional	Hashtable	Names and values for any additional options for the type. `config`, `includeOwners`, `includeGroups`, `addTo`, and `redun` cannot be specified using the `extraOpts` argument. Instead, use the command-line equivalent options described in this table to specify these values. One example of an `extraOpts` parameter is `descriptorPath`. This parameter enables you to specify the path to the resource descriptor file that you plan to reference when checking in files. For more information, see Chapter 5, "Resource Descriptor Schema," in *Sun N1 Service Provisioning System 5.2 XML Schema Reference Guide*.
result		CompCheckInID	The ID for this component check in job.

cdb.rsrc.cib

The command is the "check-in batch" command. It checks in all the components listed in a batch file.

TABLE 3–40 Arguments and Result of the cdb.rsrc.cib

Argument/Result		Syntax	Description
batchfile	Required	String	The name of the batch file listing the components to be checked in

TABLE 3–40 Arguments and Result of the cdb.rsrc.cib *(Continued)*

Argument/Result		Syntax	Description
haltonerror	Optional	Boolean	When true, first error will halt batch execution, default true
pwdrelative	Optional	Boolean	When true, relative paths are relative to the user directory; otherwise they are relative to the batchfile location, defaults to false
result		String	Message indicating the operation is complete

Overview of Batch Files

The `rsrc.cib` operates on a batch file that includes a line for each component that will be checked in. Batch files enable you to check-in large numbers of component with a single command.

Each line in the batch file corresponds to a single component on the local machine that will be checked in as a single component. Each line consists of a series of fields that are separated by the pipe character (|). Some fields are optional and may be omitted. If an optional field is omitted but is followed by other fields, the omitted field should be followed by a | character, so that `rsrc.cib` can accurately identify each field.

You can include comments in a batch file. Any line that begins with the pound character (#) is interpreted as a comment.

The following table describes the syntax of a line of a batch file.

TABLE 3–41 Syntax of a Line in a Batch File

Content	Optional/Required
The location of the component on the local machine	Required
The name to be assigned to the component when checked in	Required
The component type	Required
The platform the component is intended for expressed as a HostSetID in the form NM:<platform_name>, where <platform_name> is one of the platform names listed in Table 3–42.	Optional
A description of the component	Optional

TABLE 3–41 Syntax of a Line in a Batch File *(Continued)*

Content	Optional/Required
A boolean designation of whether the file is a configuration file	Optional (Default is false)
A boolean designation of whether check-in should be assigned a major version number (e.g., 2.0)	Optional (Default is false)
A boolean designation of whether to hide the previous most recent version of the component	Optional (Default is true)
A boolean designation of whether to include owner information when storing permissions information	Optional (Default is true)
A boolean designation of whether to include group information when storing permissions information	Optional (Default is true)
A boolean designation of whether the files being checked in should be added to the existing files to create a new version, instead of creating a new version by completely replacing the existing files	Optional (Default is true)
If this component is being checked in a from a host, the host ID of the host from which the component is being checked in	Optional
A boolean designation of whether redundancy checking should apply	Optional (Default is true)
The name of the picker to use (optional, defaults to null for the default picker)	
A Hashtable in string form containing extra options supported by the type's exporter. Note that the boolean values for the following cannot be specified using the `extraOpts` argument: ■ Whether the file is a configuration template ■ Whether to include owner information ■ Whether to include group information ■ Whether the files being checked in should be added to existing files ■ Whether to perform redundancy checking Instead, use the batch file format equivalent options to specify these values.	

The following table lists the names that you can use in the fourth field of a batch file line to specify a platform for the component.

TABLE 3–42 Names for Platforms

Platform Name	Description
`any`	Any platform supported by the Sun N1 Service Provisioning System software
`AIX - any version`	Either IBM AIX 5.2 or IBM AIX 5.2
`AIX 5.2`	IBM AIX 5.2
`AIX 5.2`	IBM AIX 5.2
`Solaris - any version`	Solaris™ 6, Solaris 7, or Solaris 8 releases
`Solaris 7`	Solaris 7 release
`Solaris 8`	Solaris 8 release
`Solaris 9`	Solaris 9 release
`Solaris 10`	Solaris 10 release
`Windows 2000 Server`	Microsoft Windows 2000 Server
`Red Hat Linux`	Red Hat Advanced Server 2.1

Example of a Line in a Batch File

To check in a local file named `home/etc/myfile` as the component `mypath/mycomponentname` as the component type file for the platform Solaris 7 with the description "this is my file" and no designation as a configuration file, you would enter the following line in a batch file:

```
/home/myfile|mypath/mycomponentname|file|NM:Solaris 7|this is my file
```

If the file being checked in was a configuration file, you would add a boolean field to the end of the line and the field to `true`. For example:

```
/home/myfile|mypath/mycomponentname|file|NM:Solaris 7|this is my file|true
```

If you wanted to omit a description for the mycomponent, you do change this line to the following (note the adjacent pipe separators):

/home/myfile|mypath/mycomponentname|file|NM:Solaris 7||true

To check in the component as a major version (e.g., 2.0 as opposed to 1.7), you would add `true` in the boolean field for major version check-ins:

/home/myfile|mypath/mycomponentname|file|NM:Solaris 7`||true |true`

If the check in was desired to not hide the previous component, the line above would become (note the additional `false` in the final field):

/home/myfile|mypath/mycomponentname|file|NM:Solaris 7`||true |true|false`

Similar format considerations apply to the optional boolean specifying whether to include owner and group information when storing permissions information.

Batch File Syntax

The Sun N1 Service Provisioning System software applies these rules when parsing batch files.

- In fields that are known to be path names, slashes (whether forward or backward) are always translated to accommodate the convention used on the native file system.
- Blank lines are allowed as visual separators of clusters of files.
- Leading or trailing whitespace is not stripped from fields.
- Both absolute and relative paths are allowed in a batchfile. By default, relative paths are interpreted as being relative to the batchfile location; this can be overridden with the `-pwdrelative` flag, in which case relative paths will be interpreted as being relative to the current working directory.

Invocation

Batch check in via text file is invoked via a `cdb.src.cib` command ("cib" = check in batch) of the form

```
cdb.rsrc.cib -batchfile [batchfile location] [-haltonerror true|false]
[-pwdrelative true|false]
```

Before checking in any components, the `cdb.rsrc.cib` command performs a syntax check of the file. Next it verifies the existence of all the local files that are to be checked in. If `cdb.rsrc.cib` detects errors in either of these processes, it reports the errors and halts execution (regardless of the setting of the `--haltonerror` boolean argument).

The haltonerror Argument

The command line includes an optional `-haltonerror` argument (false by default) that designates whether or not an error from the check-in of a single file should halt the check-in of subsequent files. This boolean argument applies only to errors encountered after `cdb.rsrc.cib` has performed its preliminary error-checking (described in the section above).

The pwdrelative Argument

The command line includes an optional `-pwdrelative` provision (false by default) that designates whether relative paths in the batch file should be interpreted as being relative to the current working directory (`pwdrelative` = true) or relative to the location of the batchfile (`pwdrelative` = false).

Batch File Processing

Batchfile processing is non-transactional. This means that if batch file processing fails and/or halts before completion, any components that have been successfully checked in remain checked in, and are not "un"-checked in.

Concurrent batch check ins are not arbitrated. If two different batch check-ins targeting the same set of components begin to run at the same time, there is no mechanism throttling the processing of one batch file while another completes. Both batch files will be processed in the interleaved manner that results from their proximate timing.

cdb.rsrc.co

This command checks out the specified component. It transfers a copy from the repository to the local machine.

TABLE 3–43 Arguments and Result for the cdb.rsrc.co

Argument/Result		Syntax	Description
src	Required	String	The name of the component to transfer
v	Required	String	The version of the component
dst	Required	String	The location where the component is to be placed
result		String	Message indicating the operation is complete

cdb.rsrc.gd

This command generates a resource descriptor for the specified component. For more information about resource descriptors, see "Resource Descriptor File Concepts" in *Sun N1 Service Provisioning System 5.2 Plan and Component Developer's Guide*.

TABLE 3–44 Argument and Result for the cdb.rsrc.gd Command

Argument/Result		Syntax	Description
`ID`	Required	ComponentID	The ID of the component for which to generate the descriptor.
example: `cr_cli -cmd cdb.rsrc.gd -ID NM:`*component-name*`[:`*version*`] -u <username> -p <password>`			

cdb.rsrc.rci

This command re-checks in a component. If a check-in job has been interrupted, you can use this command to repeat the check-in without artificially incrementing the version number of the checked-in component.

TABLE 3–45 Argument and Result for the cdb.rsrc.rci Command

Argument/Result		Syntax	Description
`ID`	Required	ComponentID	The ID of the component to re-check in.
result		CompCheckInID	The ID of the resulting check in job.

cdb.rsrc.showopts

The command shows the checkin options supported by a particular component type.

TABLE 3–46 Arguments and Result of the cdb.rsrc.showopts

Argument/Result		Syntax	Description
type	Required	String	The type of the component
result		BrowserInfoArray	The component picker names and options supported by the exporter

cdb.cj: Managing Check-in Jobs

Checking in a component creates a check-in job. A check-in job is a process that lasts until the component has been fully entered in the repository and assigned a version number. The `cdb.cj` commands enable you to control and monitor check-in jobs.

TABLE 3–47 CLI Commands for Controlling and Monitoring Check-in Jobs

Command	Description
cdb.cj.la	Lists all check-in jobs.
cdb.cj.lo	Lists the status and details of a check-in job.
cdb.cj.stop	Stops the check-in job.

cdb.cj.la

This command lists check-in jobs (components being checked in through the HTML user interface or through a CLI check-in command such as `rsrc.ci`). It lists all the jobs that are currently active, as well as the last 20 jobs that completed.

TABLE 3–48 Result of the cdb.cj.la Command

Result	Syntax	Description
result	CompCheckInIdArray	The list of check in job IDs

cdb.cj.lo

This command displays the status and details of the specified check-in job. You specify a job by its CompCheckInID. This value is returned by `cdb.rsrc.ci` when you check in a component.

TABLE 3–49 Argument and Result of the cdb.cj.lo Command

Argument/Result		Syntax	Description
ID	Required	CompCheckInID	The ID of the check in job
result		CompStatus	The check in job that was specified

Note – Because this command requires the ID of the check-in job, it does not support ID NM: notation for its argument. See Appendix A for a detailed description of compCheckInId syntax.

cdb.cj.stop

This command stops the specified check-in job.

TABLE 3–50 Argument and Result of the rsrc.cj.stop

Argument/Result		Syntax	Description
ID	Required	CompCheckInID	The ID of the check in job
result		CompStatus	The check in job that was specified

Note – Because this command requires the ID of the check-in job, it does not support ID NM: notation for its argument. See Appendix A for a detailed description of CheckInJobID syntax.

◆ ◆ ◆ CHAPTER 4

cfg: CLI Commands for Performing Config-Generation

This chapter describes the command needed to perform config-generation.

Overview of the `cfg` Command

The `cfg` commands perform config-generation.

TABLE 4–1 Summary of the cfg Commands

Command	Description
cfg.gen	Generates configuration for an input.

cfg.gen

This command generates configuration for an input.

TABLE 4–2 Arguments for the cfg.gen Command

Argument		Syntax	Description
path	Required	ReaderWrapper	The input source
host	Required	Host	The target host
comp	Required	Component	The target component
vs	Optional	ComponentVariable-Settings	The target override variable settings

◆ ◆ ◆ CHAPTER 5

cmp: CLI Commands for Running Comparisons

This chapter describes commands that you need to use to run comparisons.

Overview of the `cmp` Commands

The `cmp` commands control comparisons.

TABLE 5–1 Summary of cmp Commands

Command Name	Description
cmp.dj.add	Adds (starts) a new comparison.
cmp.dj.del	Deletes (stops) a new comparison.
cmp.dj.la	Lists running and completed comparisons.
cmp.dj.lo	Retrieves a running comparison to display its status.
cmp.ds.add	Adds comparison settings to the specified comparison.
cmp.ds.la	Lists all comparison settings.
cmp.ds.lo	Retrieves the specified comparison settings.
cmp.ds.del	Deletes comparison settings from the specified comparison.
cmp.ds.sc	Associates a comparison settings with a set of categories.
cmp.ds.mod	Modifies the specified comparison settings.

cmp.dj.add

This command adds (starts) a new comparison.

TABLE 5–2 Argument and Result for the cmp.dj.add Command

Argument/Result		Syntax	Description
ID	Required	DifferenceSettings	The comparison settings
result		DifferenceJobID	The ID of the new comparison

cmp.dj.del

This command deletes (stops) a running comparison.

TABLE 5–3 Argument and Result for the cmp.dj.del Command

Argument/Result		Syntax	Description
ID	Required	DifferenceJobID	The ID of the running comparison
result		Boolean	True if the comparison was successfully stopped

cmp.dj.la

This command lists running and completed comparisons.

TABLE 5–4 Argument and Result for the cmp.dj.la Command

Argument/Result		Syntax	Description
max	Optional	Integer	The maximum number of jobs to list; ignored if there are more running comparisons than the given number
result		RunningDiffBeanArray	All the running comparisons and some completed ones

cmp.dj.lo

This command retrieves a running comparison to display its progress.

TABLE 5–5 Arguments for the cmp.dj.lo

Argument/Result		Syntax	Description
ID	Required	DifferenceJobID	The ID of the comparison
old	Optional	Boolean	Whether to show old messages; default is false
new	Optional	Boolean	Whether to poll for new messages; default is true

cmp.ds.add

This command adds new comparison settings; some optional arguments may be required for some style/level/scope combinations

TABLE 5–6 Arguments and Result for cmp.ds.add

Argument/Result		Syntax	Description
name	Required	String	The name of the comparison settings
desc	Optional	String	The description of the settings
style	Required	Style	The style of the settings
level	[R/O]	Level	The level of the settings
scope	[R/O]	Scope	The scope of the settings
srchID	[R/O]	HostID	The source host ID of the settings
dsthID	[R/O]	HostID	The destination host ID of the settings
dsthsID	[R/O]	HostSetID	The destination host set ID of the settings
srcdir	[R/O]	String	The source directory of the settings
dstdir	[R/O]	String	The destination directory of the settings
ignp	[R/O]	StringArray	The ignore paths of the settings

TABLE 5–6 Arguments and Result for cmp.ds.add *(Continued)*

Argument/Result		Syntax	Description
cRef	[R/O]	InstalledComponent- Ref	The component reference of the settings
tout	Required	TimeInterval	The timeout of the settings
fsl	Optional	Boolean	Whether to follow symbolic links or not; default is true.
incd	Optional	Boolean	The include subdirectories flag of the settings; default is false
srcP	[R/O]	ReaderWrapper	The source prepare mini-plan of the settings
dstP	[R/O]	ReaderWrapper	The destination prepare mini-plan of the settings
srcC	[R/O]	ReaderWrapper	The source cleanup mini-plan of the settings
dstC	[R/O]	ReaderWrapper	The destination cleanup mini-plan of the settings
usePlans	[R/O]	Boolean	The flag of the settings that indicates whether mini-plans should be executed; default is true
dstUseSrcP	[R/O]	Boolean	The flag of the settings that indicates whether source prepare mini-plan is used on destination; default is true
dstUseSrcC	[R/O]	Boolean	The flag of the settings that indicates whether source cleanup mini-plan is used on destination; default is true
result		DifferenceSettings	The new comparison settings

cmp.ds.la

This command lists all comparison settings.

TABLE 5–7 Argument and Result for the cmp.ds.la Command

Argument/Result		Syntax	Description
cat	Optional	CategoryID	Category filter to apply, default “all”
result		DifferenceSettings- Array	All the comparison settings

cmp.ds.lo

This command retrieves the specified comparison settings.

TABLE 5–8 Argument and Result for the cmp.ds.lo

Argument/Result		Syntax	Description
ID	Required	DifferenceSettingsID	The ID of the comparison settings
result		DifferenceSettings	The comparison settings

cmp.ds.del

This command deleted a comparison settings object.

TABLE 5–9 Argument and Result for the cmp.ds.del

Argument/Result		Syntax	Description
ID	Required	DifferenceSettingsID	The ID of the comparison settings

cmp.ds.mod

This command adds new comparison settings by using existing settings as a template. Omitted arguments are overwritten.

TABLE 5–10 Argument and Result for the cmp.ds.mod

Argument/Result		Syntax	Description
ID	Required	DifferenceSettingsID	The ID of the template comparison settings

TABLE 5–10 Argument and Result for the cmp.ds.mod *(Continued)*

Argument/Result		Syntax	Description
desc	Optional	String	The description of the settings
style	Optional	Style	The style of the settings
level	[R/O]	Level	The level of the settings
scope	[R/O]	Scope	The scope of the settings
srchID	[R/O]	HostID	The source host ID of the settings
dsthID	[R/O]	HostID	The destination host ID of the settings
dsthsID	[R/O]	HostSetID	The destination host set ID of the settings
srcdir	[R/O]	String	The source directory of the settings
dstdir	[R/O]	String	The destination directory of the settings
ignp	[R/O]	StringArray	The ignore path of the settings
cRef	[R/O]	InstalledComponent- Ref	The component reference of the settings
tout	Optional	TimeInterval	The timeout of the settings
fsl	Optional	Boolean	Whether to follow symbolic links or not; default is true. If this argument is omitted, the value is overwritten.
incd	Optional	Boolean	The include subdirectories flag of the settings
srcP	[R/O]	ReaderWrapper	The source prepare mini-plan of the settings
dstP	[R/O]	ReaderWrapper	The destination prepare mini-plan of the settings
srcC	[R/O]	ReaderWrapper	The source cleanup mini-plan of the settings

TABLE 5–10 Argument and Result for the cmp.ds.mod *(Continued)*

Argument/Result		Syntax	Description
dstC	[R/O]	ReaderWrapper	The destination cleanup mini-plan of the settings
usePlans	[R/O]	Boolean	The flag of the settings that indicates whether mini-plans should be executed; default is true. If this argument is omitted,the value is overwritten .
dstUseSrcP	[R/O]	Boolean	The flag of the settings that indicates whether source prepare mini-plan is used on destination; default is true. If this argument is omitted, the value is overwritten.
dstUseSrcC	[R/O]	Boolean	The flag of the settings that indicates whether source cleanup mini-plan is used on destination; default is true. If this argument is omitted, the value is overwritten.
result		DifferenceSettings	The new comparison settings

cmp.ds.sc

This command associates a comparison setting with a set of categories.

TABLE 5–11 Argument and Result for the cmp.ds.sc Command

Argument/Result		Syntax	Description
ID	Required	DifferenceSettingsID	The ID of the comparison setting to affect
catIDs	Required	CategoryIDSet	The IDs of the categories to associate with this comparison setting

◆ ◆ ◆ CHAPTER 6

fdb: CLI Commands for Folder Maintenance

This chapter describes the commands that manage folders.

Overview of the fdb Commands

The following table summarizes the CLI commands for managing folders.

TABLE 6–1 Summary of the fdb Commands

Command	Description
fdb.f.add	Create a folder.
fdb.f.mod	Modify a folder.
fdb.f.mv	Move or rename a folder.
fdb.f.del	Delete a folder.
fdb.f.la	List all folders.
fdb.f.lo	List the details about one folder.
fdb.f.co	Change the owner group of a folder.
fdb.f.mp	Change the permissions for a folder.

fdb.f.add

This command created a folder.

TABLE 6–2 Arguments and Result for the `fdb.f.add` Command

Argument/Result		Syntax	Description
fullname	Required	String	The full name of the folder, which is the path to the folder and the folder name
desc	Optional	String	The description of the folder
parents	[0]	Boolean	Whether nonexistent parent folders are created. The default is false.
result		Folder	The new folder

fdb.f.mod

This command modifies the specified folder.

TABLE 6–3 Arguments and Result for the `fdb.f.mod` Command

Argument/Result		Syntax	Description
ID	Required	FolderID	The folder instance ID
desc	Optional	String	The description of the folder
result		Folder	The modified folder

fdb.f.mv

This command moves or renames a folder. The contents of the folder are moved, as well.

TABLE 6–4 Arguments for the `fdb.f.mv` Command

Argument		Syntax	Description
ID	Required	FolderID	The ID of the folder to move or rename
fullname	Required	String	The new full name of the folder, which is the path to the folder and the folder name

fdb.f.del

This command deletes a folder and its contents.

TABLE 6–5 Argument and Result for the `fdb.f.del` Command

Argument/Result		Syntax	Description
ID	Required	FolderID	The ID of the folder to delete
result		none	

fdb.f.la

This command lists all folders.

TABLE 6–6 Arguments for the `fdb.f.la` Command

Argument		Syntax	Description
folderID	Optional	FolderID	The parent folder ID; default is the root folder, (NM:/)
flatView	Optional	Boolean	Should the results be listed in flat view (recursively from parent); defaults to true
result		FolderArray	The list of folders

fdb.f.lo

This command lists the details of one folder.

TABLE 6–7 Argument and Result for the `fdb.f.lo` Command

Argument/Result		Syntax	Description
ID	Required	FolderID	The ID of the folder
result		Folder	The folder

fdb.f.co

This command changes the owner group of a folder.

TABLE 6–8 Arguments and Result for the `fdb.f.co` Command

Argument/Result		Syntax	Description
ID	Required	FolderID	The folder ID
groupID	Required	GroupID	The group ID
result		Folder	The modified folder

fdb.f.mp

This command changes the permissions of a folder.

TABLE 6–9 Arguments and Result for the `fdb.f.mp` Command

Argument/Result		Syntax	Description
ID	Required	FolderID	The folder ID
groupID	Required	GroupID	The group ID
write	Optional	Boolean	Whether the group has `write` permissions on the folder. Default is false. If set to true, `checkin-current` and `autorun` permissions are also set to true.
checkin-current	Optional	Boolean	Whether the group has `checkin-current` permission on the folder. Default value equals `write` permission value. If unspecified while updating a group's permissions, value reverts to default.
autorun	Optional	Boolean	Whether the group has `autorun` permissions on the folder. Default value equals the `write` permissions value. If unspecified while updating a group's permissions, value reverts to default.

TABLE 6–9 Arguments and Result for the `fdb.f.mp` Command *(Continued)*

Argument/Result		Syntax	Description
execute	Optional	HostSetID	The hostSetID for which the `execute` permission is to be applied. An empty value will remove the `execute` permission on any hostsets. To set this permission for "all" hostsets, clients use the "`allhosts`" sentinel value. (Note that this sentinel value is rendered as "universal set" in the browser interface.)
delete-history	Optional	HostSetID	The host set ID for which the delete-history permission is to be applied. An empty value will remove the delete-history permission on any host set. To set this permission for "all" host sets, clients use the "allhosts" sentinel value.
result		Folder	The modified folder

◆ ◆ ◆ CHAPTER 7

hdb: CLI Commands for Managing Hosts

This chapter describes the commands that you need to manage hosts.

Introduction

The CLI includes the following sets of commands for managing hosts.

TABLE 7–1 Sets of Commands for Managing Hosts

CLI Prefix	Description of Command Set
hdb.a	Commands for managing application instances (Applications are the individual Remote Agents, and Local Distributors).
hdb.h	Commands for managing target hosts.
hdb.hr	Commands for managing host searches.
hdb.hs	Commands for managing host sets.
hdb.ht	Commands for managing host types.

This chapter describes all the commands in each of these sets.

hdb.a: Managing Application Instances

The `hdb.a` family of commands controls instances of the provisioning software's applications, such as the Remote Agents and Local Distributors.

TABLE 7–2 Summary of hdb.a Commands

Command Name	Description
hdb.a.add	Adds a new application instance
hdb.a.del	Deletes an application instance
hdb.a.la	Lists all application instances
hdb.a.lo	Retrieves information about a specific application instance
hdb.a.mod	Modifies an existing application instance
hdb.a.clear	Clears the resource cache of an application instance

hdb.a.add

This command adds a new application instance; it registers a specific configuration of a specific Sun N1 Service Provisioning System software application with the Master Server.

TABLE 7–3 Arguments and Result for the hdb.a.add Command

Argument/Result		Syntax	Description
hID	Required	HostID	The ID of the host to contain the new instance
type	Required	AppType	The application type (RA|LD)
pID	Required	AppInstanceID	The parent application ID
ip	Required	String	The application IP address (x.y.z.w)
port	Optional	Integer	The application port (not required for ssh)
conn	Required	ConnectionType	The connection type (raw|ssh|ssl)
param	Optional	String	The parameters
result		UIAppInstanceUpdate	The new application instance and associated warnings

Note – Omitting the `-port` argument or setting the port to 0 on an application instance with connection type `ssh` will cause the default `ssh` port to be used.

hdb.a.del

This command deletes an application instance from the Master Server's internal files. Once deleted, the application instance is no longer recognized by the Master Server.

Note – Deleting an application instance does not delete Sun N1 Service Provisioning System software files on the machine where the application instance was installed.

TABLE 7–4 Argument for the hdb.a.del Command

Argument		Syntax	Description
ID	Required	AppInstanceID	The application instance ID

EXAMPLE 7–1 Example of Deleting an Application Instance

This example demonstrates deleting a Remote Agent and Local Distributor from a host named `NewYork-v240`.

Host name `NewYork-v240`

```
[Deleting the Remote Agent]
% ./cr_cli -cmd hdb.a.del -ID NM:NewYork-v240:RA -u admin -p admin

[Deleting the Local Distributor]
% ./cr_cli -cmd hdb.a.del -ID NM:NewYork-v240:LD -u admin -p admin
```

hdb.a.la

This command lists all the application instances known to the Master Server.

This command does not accept any arguments.

TABLE 7–5 Result Returned by the hdb.a.la command

Result	Result Syntax	Description
result	AppInstanceArray	The application instances

hdb.a.lo

This command retrieves information about a specified application instance.

TABLE 7–6 Argument and Results of the hdb.a.lo Command

Argument/Result		Syntax	Description
ID	Required	AppInstanceID	The application instance ID
result		AppInstance	The application instance

hdb.a.mod

This command modifies the specified attributes of an existing application instance. Omitted arguments are overwritten.

TABLE 7–7 Arguments and Result for the hdb.a.mod Command

Argument/Result		Syntax	Description
ID	Required	AppInstanceID	The application instance ID
pID	Optional	AppInstanceID	The new parent application ID
ip	Optional	String	The new application IP address (x.y.z.w)
port	Optional	Integer	The new application port
conn	Optional	ConnectionType	The new connection type (raw|ssh|ssl)
param	Optional	String	The new parameters
result		UIAppInstanceUpdate	The modified application instance and associated warnings

Note – Omitting the `-port` argument or setting the port to 0 on an application instance with connection type `ssh` will cause the default `ssh` port to be used.

hdb.a.clear

This command clears the resource cache of an application instance.

TABLE 7–8 Arguments and Result for the hdb.a.clear Command

Argument/Result		Syntax	Description
ID	Required	AppInstanceID	The application instance ID

hdb.h: Managing Hosts

The `hdb.h` commands manage target hosts: physical and virtual hosts whose applications you are managing with the Sun N1 Service Provisioning System software.

TABLE 7–9 Summary of hdb.h Commands

Command	Description
hdb.h.add	Adds a new host.
hdb.h.del	Deletes a host
hdb.h.la	Lists all hosts.
hdb.h.lo	Retrieves information about a host.
hdb.h.lq	Queries for matching hosts.
hdb.h.mod	Modifies an existing host.

hdb.h.add

This command adds a new host to the host repository.

TABLE 7–10 Arguments and Results for the hdb.h.add Command

Argument/Result		Syntax	Description
name	Required	String	The host name
desc	Optional	String	The host description
tID	Required	HostTypeID	The ID of the host type
attr	Optional	Hashtable	The host attributes
hide	Optional	Boolean	Whether the host is hidden, default false
pID	Optional	HostID	The ID of the parent host if virtual, or empty if physical
result		Host	The new host

Note – The `-attr` argument only sets the overrides. Attributes not explicitly included retain their default value.

EXAMPLE 7–2 Example of Adding a Host

This example demonstrates adding a basic host to the provisioning software.

Host name `NewYork-v240`

Host type `system#crhost`

```
% ./cr_cli -cmd hdb.h.add -name NewYork-v240  \
-tID NM:system#crhost -u admin -p admin
```

EXAMPLE 7–3 Example of Adding a Virtual Host

This example demonstrates adding a virtual host that is located on a server named `NewYork-v240`.

Virtual host name `MyAppServer-vhost`

Host type `system#crhost`

Parent host name `NewYork-v240`

```
% ./cr_cli -cmd hdb.h.add -name MyAppServer-vhost  \
-tID NM:system#crhost -pID NM:NewYork-v240 -u admin -p admin
```

hdb.h.del

This command deletes a host and all application instances and components that were installed on the host. For more information, see "Deleting a Host" in *Sun N1 Service Provisioning System 5.2 System Administration Guide.*

TABLE 7–11 Arguments and Results for the hdb.h.del Command

Argument/Result		Syntax	Description
ID	Required	HostID	The host ID

EXAMPLE 7–4 Example of Deleting a Host Record

This example demonstrates deleting a host.

Host name `NewYork-v240`

```
% ./cr_cli -cmd hdb.h.del -ID NM:NewYork-v240 -u admin -p admin
```

hdb.h.la

This command lists all hosts in the repository.

TABLE 7–12 Argument and Result for the hdb.h.la Command

Argument/Result		Syntax	Description
sh	Optional	Boolean	Whether hidden hosts are shown, default false
result		HostArray	The hosts

hdb.h.lo

This command retrieves information about a host from the repository.

TABLE 7–13 Argument and Result for the hdb.h.lo Command

Argument/Result		Syntax	Description
ID	Required	HostID	The host ID
result		Host	The host

hdb.h.lq

This command queries for hosts that match either specified criteria or a specified filter.

Note – You must call this command with either a query (with `-query`) or a filter (with `-filt` or `-phys`).

TABLE 7–14 Arguments and Result for the hdb.h.lq Command

Argument/Result		Syntax	Description
query	Optional	AttributeCriteriaList	The query criteria
filt	Optional	AppTypeCriteria	The application type filter
phys	Optional	PhysicalCriteria	A filter to restrict to physical or virtual hosts
sh	Optional	Boolean	Whether hidden hosts are shown, default false
result		HostArray	The hosts

hdb.h.mod

This command modifies the specified attributes of an existing host. Omitted arguments are overwritten.

TABLE 7–15 Arguments and Result of the hdb.h.mod Command

Argument/Result		Syntax	Description
ID	Required	HostID	The host ID
name	Optional	String	The host name
desc	Optional	String	The host description
tID	Optional	HostTypeID	The ID of the host type
attr	Optional	Hashtable	The host attributes
hide	Optional	Boolean	Whether the host is hidden
pID	Optional	HostID	The ID of the parent host if virtual, or "<null>" if physical

TABLE 7–15 Arguments and Result of the hdb.h.mod Command *(Continued)*

Argument/Result	Syntax	Description
result	Host	The modified host

Note – Use the `-attr` argument to specify the attributes whose values you want to modify. Attributes not explicitly specified retain (or are reset to) their default values.

hdb.hr: Managing Host Searches

Overview

The `hdb.hr` commands manage host searches.

TABLE 7–16 Summary of hdb.hr Commands

Command	Description
hdb.hr.add	Adds a new host search.
hdb.hr.del	Deletes a host search.
hdb.hr.la	Lists all host searches.
hdb.hr.le	Lists all the hosts returned by a host search.
hdb.hr.lo	Retrieves information about a host search.
hdb.hr.mod	Modifies an existing host search.

hdb.hr.add

This command adds a new host search. It enters the host search's name and search criteria in the repository.

TABLE 7–17 Arguments and Result for the hdb.hr.add Command

Argument/Result		Syntax	Description
name	Required	String	The host search name

TABLE 7–17 Arguments and Result for the hdb.hr.add Command *(Continued)*

Argument/Result		Syntax	Description
desc	Optional	String	The host search description
q	[R/O]	AttributeCriteriaList	The dynamic query; must be specified if filt and phys are not
filt	[R/O]	AppTypeCriteria	A filter to restrict the search to certain types of hosts; must be specified if q and phys are not
phys	[R/O]	PhysicalCriteria	A filter to restrict to physical or virtual hosts; must be specified if q and filt are not
hide	Optional	Boolean	Whether the search is hidden, default false
result		HostSearch	The new host search

hdb.hr.del

This command deletes a host search.

TABLE 7–18 Arguments and Result for the hdb.hr.del Command

Argument		Syntax	Description
ID	Required	HostSearchID	The host search ID

hdb.hr.la

This command lists all host searches defined in the repository.

TABLE 7–19 Argument and Result for the hdb.hr.la Command

Argument/Result		Syntax	Description
sh	Optional	Boolean	Whether hidden searches are shown, default false
result		HostSearchArray	The host searches

hdb.hr.le

This command lists all hosts that match the criteria of the specified host search. The lists represents hosts that match these criteria when hdb.hr.le is run.

TABLE 7–20 Arguments and Result for the hdb.hr.le Command

Argument/Result		Syntax	Description
ID	Required	HostSearchID	The host search ID
sh	Optional	Boolean	Whether hidden searches are shown, default false
result		HostArray	The hosts

hdb.hr.lo

This command retrieves information about a specified host search.

TABLE 7–21 Argument and Result for the hr.hr.lo

Argument/Result		Syntax	Description
ID	Required	HostSearchID	The host search ID
result		HostSearch	The host search

hdb.hr.mod

This command modifies an existing host search. Omitted arguments are overwritten.

TABLE 7–22 Arguments and Results for the hdb.hr.mod Command

Argument/Result		Syntax	Description
ID	Required	HostSearchID	The host search ID
name	Optional	String	The host search name
desc	Optional	String	The host search description
q	Optional	AttributeCriteriaList	The dynamic query
filt	Optional	VarValueCriteria	An optional filter to restrict the search to certain types of hosts

TABLE 7–22 Arguments and Results for the hdb.hr.mod Command *(Continued)*

Argument/Result		Syntax	Description
phys	Optional	PhysicalCriteria	A filter to restrict to physical or virtual hosts
hide	Optional	Boolean	Whether the search is hidden
result		HostSearch	The modified host search

hdb.hs: Managing Host Sets

The hdb.hs commands manage host sets.

TABLE 7–23 Summary of hdb.hs Commands

Command	Description
hdb.hs.add	Adds a new host set.
hdb.hs.del	Deletes a host set.
hdb.hs.la	Lists all host sets.
hdb.hs.le	Lists all the hosts contained in a host set.
hdb.hs.lo	Retrieves information about a host set.
hdb.hs.mod	Modifies an existing host set.

hdb.hs.add

This command adds a new host set to the repository.

TABLE 7–24 Arguments and Result for the hdb.hs.add Command

Argument/Result		Syntax	Description
name	Required	String	The host set name
desc	Optional	String	The host set description
hIDs	Optional	HostIDSet	The IDs of the static member hosts
sIDs	Optional	HostSetIDSet	The IDs of the host subsets

TABLE 7–24 Arguments and Result for the hdb.hs.add Command *(Continued)*

Argument/Result		Syntax	Description
rIDs	Optional	HostSearchIDSet	The IDs of the host searches
hide	Optional	Boolean	Whether the host set is hidden, default false
result		HostSet	The new host set

hdb.hs.del

This command deletes a host set.

TABLE 7–25 Arguments and Result for the hdb.hs.del Command

Argument		Syntax	Description
ID	Required	HostSetID	The host ID

hdb.hs.la

This command lists all host sets defined in the Sun N1 Service Provisioning System software.

TABLE 7–26 Argument and Result for the hdb.hs.la Command

Argument/Result		Syntax	Description
hide	Optional	Boolean	Whether hidden host sets are shown, default false
result		HostSetArray	The host sets

hdb.hs.le

This command lists all the hosts contained in the specified host set.

TABLE 7–27 Arguments and Results for the hdb.hs.le Command

Argument		Syntax	Description
ID	Required	HostSetID	The host set ID
hide	Optional	Boolean	Whether hidden host sets are shown, default false

TABLE 7–27 Arguments and Results for the hdb.hs.le Command *(Continued)*

Argument	Syntax	Description
result	HostArray	The hosts

hdb.hs.lo

This command retrieves the specified host set.

TABLE 7–28 Argument and Result for the hdb.hs.lo Command

Argument/Result		Syntax	Description
ID	Required	HostSetID	The host set ID
result		HostSet	The host set

hdb.hs.mod

This command modifies an existing host set; omitted arguments are overwritten.

TABLE 7–29 Arguments and Result for the hdb.hs.mod Command

Argument/Result		Syntax	Description
ID	Required	HostSetID	The host set ID
name	Optional	String	The host set name
desc	Optional	String	The host set description
hIDs	Optional	HostIDSet	The IDs of the static member hosts
sIDs	Optional	HostSetIDSet	The IDs of the host subsets
rIDs	Optional	HostSearchIDSet	The IDs of the host searches
hide	Optional	Boolean	Whether the host set is hidden
result		HostSet	The modified host set

hdb.ht: Managing Host Types

Overview

The `hdb.ht` commands manage host types.

TABLE 7–30 Summary of hdb.ht Commands

Command	Description
hdb.ht.add	Adds a new host type.
hdb.ht.del	Deletes a host type.
hdb.ht.la	Lists all host types.
hdb.ht.lo	Retrieves information about a host type.
hdb.ht.mod	Modifies an existing host type.

hdb.ht.add

This command adds a new host type. It assigns the host type a name and defines the host type's attributes.

TABLE 7–31 Arguments and Result for the hdb.ht.add Command

Argument/Result		Syntax	Description
name	Required	String	The host type name
desc	Optional	String	The host type description
attr	Required	HostTypeVarList	The host type attributes
hide	Optional	Boolean	Whether the host type is hidden, default false
result		HostType	The new host type

hdb.ht.del

This command deletes a host type.

TABLE 7–32 Arguments and Result for the hdb.ht.del Command

Argument		Syntax	Description
ID	Required	HostTypeID	The host type ID

hdb.ht.la

This commands lists all the host types defined, including the default host type, `crhost`.

TABLE 7–33 Argument and Result for the hdb.ht.la Command

Argument/Result		Syntax	Description
sh	Optional	Boolean	Whether hidden host types are shown, default false
result		HostTypeArray	The host types

hdb.ht.lo

This command retrieves a host type.

TABLE 7–34 Argument and Result for the hdb.ht.lo Command

Argument/Result		Syntax	Description
ID	Required	HostTypeID	The host type ID
result		HostType	The host type

hdb.ht.mod

This command modifies an existing host type. Omitted arguments are overwritten.

TABLE 7–35 Arguments and Result for the hdb.ht.mod Command

Argument/Result		Syntax	Description
ID	Required	HostTypeID	The host type ID
name	Optional	String	The host type name
desc	Optional	String	The host type description

TABLE 7–35 Arguments and Result for the hdb.ht.mod Command *(Continued)*

Argument/Result		Syntax	Description
attr	Optional	HostTypeVarList	The host type attributes
hide	Optional	Boolean	Whether the host type is hidden
result		HostType	The modified host type

◆ ◆ ◆ CHAPTER 8

net: CLI Commands for Performing Network Operations

This chapter describes the commands that you need to use to perform network operations.

Overview of the net Commands

The net commands perform networking tasks related to Sun N1 Service Provisioning System software applications.

TABLE 8–1 Summary of the net Commands

Command	Description
net.gencfg	Generates the Transport Config file for a Sun N1 Service Provisioning System software application.
net.ping	Checks connectivity to a Remote Agent or Local Distributor by executing the TCP/IP ping command.
net.traceroute	Uses the IP traceroute utility to find the route to a Remote Agent or Local Distributor

Note – A user session is not needed for the net commands.

net.gencfg

This command generates the Transport Config file for an application.

TABLE 8–2 Argument for the net.gencfg Command

Argument		Syntax	Description
appID	Required	AppInstanceID	The AppInstanceID of the AppInstance for which the Transport config needs to be generated

net.ping

This command checks the connectivity to a Remote Agent or Local Distributor by executing the TCP/IP ping command.

TABLE 8–3 Argument and Result for the net.ping Command

Argument/Result		Syntax	Description
d	Required	RoxAddress	The address of the Remote Agent or Local Distributor to ping in the format [*DNS-host-name*|*ipAddress*:*port*]
result		PingResult	Message indicating that the ping command succeeded, or error details if the command failed

net.traceroute

This command uses the IP traceroute command to find the route to a Remote Agent or Local Distributor

TABLE 8–4 Argument and Result for the net.traceroute Command

Argument/Result		Syntax	Description
d	Required	RoxAddress	The address of the Remote Agent or Local Distributor to find the route to in the format [*DNS-host-name*|*ipAddress*:*port*]

TABLE 8–4 Argument and Result for the `net.traceroute` Command *(Continued)*

Argument/Result	Syntax	Description
result	UITraceResult	Message indicating that the `traceroute` command succeeded, or error details if the command failed

◆ ◆ ◆ CHAPTER 9

node: CLI Commands for Performing Auto-upgrade Tasks

This chapter describes the commands that you need to use to perform auto-upgrade tasks.

Overview of the node Commands

The node commands perform auto-upgrade tasks on hosts, host sets, and application instances.

TABLE 9–1 Summary of the node Commands

Command	Description
node.au	Performs auto-upgrade tasks

node.au.la

This command provides a summary of all the completed and currently running auto-upgrade tasks.

TABLE 9–2 Argument and Result for the node.au.la Command

Argument/Result	Syntax	Description
result	UpgradeTaskSummaryArray	The upgrade tasks

node.au.lo

This command provides the status of a specified auto-upgrade task.

TABLE 9–3 Argument and Result for the `node.au.lo` Command

Argument/Result		Syntax	Description
ID	Required	UpgradeTaskID	Lists the detailed or summary status of an auto-upgrade operation given an UpgradeTaskID. Default is detailed status.
result		UpgradeTaskStatus	The upgrade tasks

node.au.run

To run an auto-upgrade task on a host, host set, application instance, a combination of these three, or all nodes. Combinations of HostIDArray, HostSetIDArray and AppInstanceIDArray can be passed together to this command. However, when using the `-all` argument, IDs should not be specified.

TABLE 9–4 Argument for the `node.au.run` Command

Argument/Result		Syntax	Description
hID	Required or Optional	HostIDArray	The HostID array. The application instances on these hosts are upgraded; required if the argument all OR hsID/ID are not supplied
hsID	Required or Optional	HostSetIDArray	The HostSetID array. The application instances on the hosts in these host sets are upgraded; required if the argument all OR hID/ID are not supplied
ID	Required or Optional	AppInstanceIDArray	The AppInstanceID array. The application instances are upgraded; required if the argument all OR hID/hsID are not supplied
all	Required of Optional	Boolean	True if entire network needs to be upgraded; required if none of the arguments hID/hsID/ID are supplied

TABLE 9–4 Argument for the node.au.run Command *(Continued)*

Argument/Result	Syntax	Description
result	UpgradeTaskID	The ID of the upgrade task run

node.au.stop

This command stops an auto-upgrade task.

TABLE 9–5 Argument and Result for the node.au.stop Command

Argument/Result		Syntax	Description
ID	Required	UpgradeTaskID	The ID of the upgrade task run to stop

◆ ◆ ◆ CHAPTER 10

pdb: CLI Commands for Managing Plans

This chapter describes the commands that you need to use to manage plans.

Overview of the pdb Commands

You can use the pdb commands to manage plans. A plan is an XML document that specifies the operations to be performed on components and hosts.

TABLE 10–1 Summary of the pdb Commands

Command	Description
pdb.p.ci	Checks in a new version of a plan.
pdb.p.co	Checks out a plan (outputs a plan in XML).
pdb.p.genplan	Generates and outputs a plan in XML.
pdb.p.la	Lists the latest versions of all plans.
pdb.p.lo	Views a plan.
pdb.p.del	Deletes an execution plan.
pdb.p.lv	Lists all the versions of the specified plan.
pdb.p.mv	Moves and/or renames a plan.
pdb.p.sh	Shows or hides a plan.
pdb.p.sc	Associates a plan with a set of categories.

pdb.p.ci

Checks in a new version of a plan from XML

TABLE 10–2 Arguments and Result for the pdb.p.ci

Argument/Result		Syntax	Description
path	Required	InputStreamWrapper	The plan in XML format
major	Optional	Boolean	Whether to checkin as a new major version; default false
hidePrev	Optional	Boolean	Whether to hide the previous plan, default true
parents	Optional	Boolean	Whether nonexistent parents are created during plan check-in. The default value is false.
result		ExecutionPlan	The new execution plan

pdb.p.co

This command outputs a plan in XML format to stdout.

TABLE 10–3 Argument for the pdp.p.co Command

Argument		Syntax	Description
plan	Required	ExecutionPlan	The execution plan

pdb.p.genplan

This command generates and outputs a plan as XML.

TABLE 10–4 Arguments for the pdb.p.genplan Command

Argument		Syntax	Description
componentID	Required	ComponentID	The ID of the component to generate the plan for.
pn	Required	StringArray	List of procedure names to use in this plan.
pt	Required	NamedBlockType[]	List of procedure types named install, uninstall, and call

pdb.p.la

This command lists the latest versions of all plans.

TABLE 10–5 Arguments and Result for the pdb.p.la Command

Argument/Result		Syntax	Description
sh	Optional	Boolean	Whether hidden plans are shown. Default is false
cat	Optional	CategoryID	Category filter to apply. Default is “all”.
folderID	Optional	FolderID	Parent folder ID; default is the root folder (NM:/)
flatView	Optional	Boolean	Whether to display results in flat view; defaults to true
result		ExecutionPlanArray	The execution plans

pdb.p.lo

This command views an execution plan.

TABLE 10–6 Argument and Result for the pdb.p.lo Command

Argument/Result		Syntax	Description
ID	Required	ExecutionPlanID	The ID of the plan to view
result		ExecutionPlan	The execution plan

pdb.p.del

This command deletes an execution plan.

TABLE 10–7 Argument and Result for the pdb.p.del Command

Argument/Result		Syntax	Description
ID	Required	ExecutionPlanID	The ID of the plan to delete

TABLE 10–7 Argument and Result for the pdb.p.del Command *(Continued)*

Argument/Result		Syntax	Description
all	Optional	Boolean	Whether to delete all versions of the plan, default false

pdb.p.lv

This command lists all the versions of the specified plan.

TABLE 10–8 Argument and Result of the pdb.p.lv Command

Argument/Result		Syntax	Description
ID	Required	ExecutionPlanID	The ID of the execution plan
result		ExecutionPlanArray	The execution plans

pdb.p.mv

This command moves or renames a plan.

TABLE 10–9 Arguments for the pdb.p.mv Command

Argument		Syntax	Description
ID	Required	ExecutionPlanID	The ID of the plan to move or rename
fullname	Required	String	The new full name (path + name) of the plan

pdb.p.sh

This command hides or shows (un-hides) a plan.

TABLE 10–10 Arguments for the pdb.p.sh Command

Argument		Syntax	Description
ID	Required	ExecutionPlanID	The ID of the plan to hide or un-hide

TABLE 10–10 Arguments for the pdb.p.sh Command *(Continued)*

Argument		Syntax	Description
hide	Required	Boolean	Whether the plan is set to hidden
all	Optional	Boolean	Whether to change all versions of the plan, default false

pdb.p.sc

This command associates a plan with a set of categories.

TABLE 10–11 Arguments for the pdb.p.sc Command

Argument		Syntax	Description
ID	Required	ExecutionPlanID	The ID of the plan to which the categories should be applied.
catIDs	Required	CategoryIDSet	The IDs of the categories to associate with this plan
all	Optional	Boolean	Whether to update the categories for all versions of the plan, default false

◆ ◆ ◆ CHAPTER 11

pe: CLI Commands for Running Plans

This chapter describes the commands that you need to use to run plans.

Overview of the pe Commands

The pe commands provide tools for running, stopping, and monitoring plans.

TABLE 11–1 Summary of the pe Commands

Command	Description
`pe.h.prep`	Prepares a set of hosts.
`pe.p.en`	Displays the output of an `<execNative>` or an `<execJava>` step.
`pe.p.la`	Lists running and completed plans.
`pe.p.lo`	Lists information about a running or completed plan.
`pe.p.del`	Deletes the history of a completed plan run.
`pe.p.lp`	Lists the subplans and targets associated with a plan.
`pe.p.run`	Runs a plan.
`pe.p.stop`	Stops a plan that is currently running.
`pe.pi.lo`	Lists the parameters used to run a plan.

pe.h.prep

This command prepares a set of hosts.

TABLE 11–2 Argument and Result of the `pe.h.prep` Command

Argument/Result		Syntax	Description
tar	Required	HostIDSet	A comma-separated list of host IDs to prepare
result		TaskID	The ID of the preparation plan run

pe.p.en

This command displays the output of an `<execNative>` or an `<execJava>` step.

TABLE 11–3 Arguments and Result for the `pe.p.en` Command

Argument/Result		Syntax	Description
ID	Required or Optional	StepDescriptionContainer	Which step in the plan to display. This can be gathered from the output of pe.p.lo command or specified by label.
IDs	Required or Optional	StepDescriptionContainerArray	Which steps in the plan to display. These can be gathered from the output of pe.p.lo command or specified by label. IDs must be specified if ID is not. If both ID and IDs is specified, IDs is ignored.
output	Optional	Either “so” or “se”	Here for backward compatibility. The output will contain both the standard error and standard out. When available the exit status is also included.
result		ExecNativeOutputArray	Output of the step or steps indicated.

Examples Using the Command

A `StepDescriptionContainer` can be either a stepID or a Label. You should use a single `StepDescriptionContainer` for the `-ID` argument. The `-IDs` argument should be a comma-separated list of `StepDescriptionContainers.` The examples use the following information to create valid commands.

taskID `1001`

stepID `2011` with a label value of `label1`

`2012` with a label value of `label2`

hostID `3001` with a hostname of `host1`

EXAMPLE 11–1 Using a Step ID

```
./cr_cli -cmd pe.p.en -ID ID:2011
./cr_cli -cmd pe.p.en -ID 2011
```

EXAMPLE 11–2 Using a Label

```
./cr_cli -cmd pe.p.en -ID L:label1:1001:NM:host1
./cr_cli -cmd pe.p.en -ID L:label1:ID:1001:ID:3001
```

EXAMPLE 11–3 Using a `StepDescriptionContainerArray`

In this example one element is a step id and the other is a label.

```
./cr_cli -cmd pe.p.en -IDs L:label1:ID:1001:ID:3001,2012
```

pe.p.la

This command lists running and completed plans.

TABLE 11–4 Arguments and Result for the `pe.p.la` Command

Argument/Result		Syntax	Description
max	Optional	Integer	The maximum number of plans to list The default is 5. This argument is ignored if there are more running plans than the given number (in this case , all running plans are listed). If the number of running plans is less than the maximum specified, the sum of all running plans and completed plans up to the maximum specified is returned.
plan	Optional	String	Restrict results to plans with this name

TABLE 11–4 Arguments and Result for the `pe.p.la` Command *(Continued)*

Argument/Result	Syntax	Description
result	RunningPlanBeanArray	All the running plans and some completed ones

pe.p.lo

This command reports on the status of a plan that is running or that has finished running. It displays the StepID of each step.

TABLE 11–5 Arguments for the `pe.p.lo` Command

Argument		Syntax	Description
ID	Required	TaskID	The ID of the plan run to view
old	Optional	Boolean	Whether to show old messages; default is false
new	Optional	Boolean	Whether to poll for new messages; default is true

pe.p.del

This command deletes the history of a completed plan run.

TABLE 11–6 Arguments for the `pe.p.del` Command

Argument		Syntax	Description
ID	Required	TaskID	The ID of the plan run to delete

pe.p.lp

This command lists the subplans and targets associated with a plan. The named plan is listed first.

TABLE 11–7 Argument and Result for the `pe.p.lp` Command

Argument/Result		Syntax	Description
`ID`	Required	ExecutionPlanID	The ID of the execution plan

TABLE 11–7 Argument and Result for the `pe.p.lp` Command *(Continued)*

Argument/Result	Syntax	Description
result	SubplanPrompt-Array	Array of subplans and targets

pe.p.run

This command runs a plan. If there are plan prompt variables, it interactively queries the user for responses.

When you run a composite plan through the CLI, you must specify a set of targets, component versions, and variable sets for each subplan. See Appendix A for further description of the syntax.

TABLE 11–8 Arguments and Result for the `pe.p.run` Command

Argument/Result		Syntax	Description
PID	Required	ExecutionPlanID	The ID of the execution plan to run
tar	Required	HostIDArrayArray	Individual hosts or host sets on which to run the plan and each of its subplans, ordered according to subplan prompts
comp	[O/R]	StringArrayArray	Component versions to install as a part of the plan and subplans. Versions are ordered according to component selectors within subplan prompts. Each referenced component must be represented in this list. Use the component version number, or "+" for default, "#" for recommended, and "-" for latest. Do not specify when there are no component versions.

TABLE 11–8 Arguments and Result for the `pe.p.run` Command *(Continued)*

Argument/Result		Syntax	Description
vs	[O/R]	StringArrayArray	Variable settings to use with each selected component version. Use variable settings name, or "+" for default. Do not specify when there are no variable settings.
po	Optional	Boolean	True if only preflight should be run
pdp	Optional	Boolean	True if detailed preflight should be run; default false.
hrl	Optional	Integer	Limits the number of hosts running at the same time; default set by server config file.
pto	Required	TimeInterval	The maximum plan execution time
nto	Required	TimeInterval	The maximum native call execution time
f	Optional	ReaderWrapper	Plan variables can be passed with a file or standard input; default is standard input
result		TaskID	The ID of the plan run

EXAMPLE 11–4 Running a Plan With a Single Subplan, Component, and Variable Set

This example demonstrates a simple plan being run on one host.

Plan name	`/test/test1`
Host targeted by plan	`masterserver`
Variable set name	`nors`
Maximum time for plan execution	30 minutes
Maximum time for native call execution	10 minutes
Component versions to be used	default

EXAMPLE 11–4 Running a Plan With a Single Subplan, Component, and Variable Set *(Continued)*

For a single component and variable set, you do not need to surround the value with quotation marks.

```
./cr_cli -cmd pe.p.run -PID NM:/test/test1 -tar H:NM:masterserver  \
-vs nors -pto 30 -nto 10 -comp + -u admin -p admin
```

EXAMPLE 11–5 Running a Plan With More Than One SubPlan, Component, and Variable Set

This example demonstrates a composite plan with two subplans.

subplan1 install compA install compB subplan2 install compC install compD

Each subplan is run on a different host and uses two different variable sets for each referenced component.

Plan name	`/test/test2`
Hosts targeted by plan	`masterserver` and `host2`
Variable set names for subplan 1	`nors` and `tmc`
Variable set names for subplan 2	`varset1` and `varset2`
Maximum time for execution	30 minutes
Maximum time for native call execution	10 minutes
Component versions to be used	default

```
./cr_cli -cmd pe.p.run -PID NM:/test/test2  \
-tar "H:NM:masterserver;H:NM:host2" -vs "nors,tmc;varset1,varset2" \
-pto 30 -nto 10 -comp "+,+;+,+" -u admin -p admin
```

pe.p.stop

This command stops a plan that is running.

TABLE 11–9 Argument and Result for the `pe.p.stop` Command

Argument/Result		Syntax	Description
ID	Required	TaskID	The ID of the plan to stop running.
result		Boolean	True if the plan was stopped successfully.

pe.pi.lo

This command lists the parameters used to run a plan.

TABLE 11–10 Argument and Result of the `pe.pi.lo` Command

Argument/Result		Syntax	Description
ID	Required	TaskID	The ID of the plan run.
result		TaskInfo	The plan run parameters

◆ ◆ ◆ CHAPTER 12

plg: CLI Commands for Plug-ins

This chapter describes the commands that manage plug-ins.

Overview of the plg Commands

The following table summarizes the CLI commands for managing plug-ins.

TABLE 12–1 Summary of the plg Commands

Command	Description
plg.p.add	Imports a new or existing plug-in from the file system.
plg.p.del	Deletes a plug-in.
plg.p.la	Lists all plug-ins.
plg.p.lo	Lists the details of a plug-in.
plg.p.mod	Modifies a plug-in.

plg.p.add

This command imports a new or existing plug-in from the file system. Note that Ctrl-C terminates the command, but does not affect the progress of the import.

TABLE 12–2 Arguments and Result for the plg.p.add Command

Argument/Result		Syntax	Description
path	Required	InputStreamWrapper	The plug-in JAR file
result		PluginID	The plug-in ID

plg.p.del

This command deletes a plug-in.

TABLE 12–3 Argument for the plg.p.del Command

Argument		Syntax	Description
ID	Required	PluginID	The plug-in ID

plg.p.la

This command lists all of the plug-ins.

TABLE 12–4 Result for the plg.p.la Command

Result	Syntax	Description
result	PluginArray	An array of plug-ins

plg.p.lo

This command lists the details about one plug-in.

TABLE 12–5 Argument and Result for the plg.p.lo Command

Argument/Result		Syntax	Description
ID	Required	PluginID	The plug-in ID
result		Plugin	The plug-in

plg.p.mod

This command modifies a plug-in. If any arguments are omitted, the values are overwritten.

TABLE 12–6 Arguments and Result for the plg.p.mod Command

Argument/Result		Syntax	Description
ID	Required	PluginID	The plug-in ID
order	Optional	String	The order of the plug-in menu

TABLE 12–6 Arguments and Result for the `plg.p.mod` Command *(Continued)*

Argument/Result	Syntax	Description
result	Plugin	The plug-in

◆ ◆ ◆ CHAPTER 13

rule: CLI Commands for Notifications

This chapter describes the commands that you need to use to manage event notifications.

Overview of the `rule` Commands

The `rule` commands let you configure email notifications to be sent when specific types of events occur in the Sun N1 Service Provisioning System software.

The following table summarizes the CLI commands for managing rules for notifications.

TABLE 13–1 Summary of the rule Commands

Command	Description
rule.add	Adds a notification rule.
rule.del	Deletes a notification rule.
rule.la	Lists all notification rules.
rule.lo	Retrieves a specific rule.
rule.mod	Modifies an existing rule.

rule.add

This command adds a new rule for sending email notifications.

TABLE 13–2 Arguments and Result for the rule.add Command

Argument/Result		Syntax	Description
n	Required	String	The rule name
d	Optional	String	The rule description
tf	Optional	CriteriaShorthand	The rule event type filter
mf	Optional	String	The rule message filter
haf	Optional	HostIDArray	The rule hosts filter
hsf	Optional	HostSetID	The rule host set filter
sf	Optional	String	The rule severity filter. Possible values are `INFO`, `WARNING`, and `ERROR`. In addition, the severity may end with +, which means this severity or greater. For example, `WARNING+` corresponds to events with severity `WARNING` or `ERROR`.
ea	Required	String	The email address to notify
result		RuleMetaData	The new rule

The following table lists the keywords you can pass with the -tf argument to specify the type of event a rule applies to.

TABLE 13–3 CriteriaShorthand for the -tf Argument

Criteria Shorthand	Event
any	Any event
planStart	A plan starts
planEndAbnormal	A plan ends abnormally
planEndNormal	A plan completes normally
cmpStart	A comparison starts
cmpEndAbnormal	A comparison ends abnormally
cmpEndNormal	A comparison completes normally

TABLE 13–3 CriteriaShorthand for the -tf Argument *(Continued)*

Criteria Shorthand	Event
system	Any system event
admin	Any administrative event
custom	Any custom events

rule.del

This command deletes the specified rule.

TABLE 13–4 Argument for the rule.del Command

Argument		Syntax	Description
ID	Required	RuleID	The rule ID

rule.la

This command lists all the rules defined in the Sun N1 Service Provisioning System software.

TABLE 13–5 Result of the rule.la Command

Result	Syntax	Description
result	RuleMetaDataArray	The rules

rule.lo

This command retrieves a rule, which is specified by its ID.

TABLE 13–6 Argument and Result for the rule.lo Command

Argument/Result		Syntax	Description
ID	Required	RuleID	The rule ID
result		RuleMetaData	The rule

rule.mod

This command modifies the specified attributes of an an existing rule. Omitted arguments are overwritten.

TABLE 13–7 Arguments and Result for the rule.mod Command

Argument/Result		Syntax	Description
ID	Required	RuleID	The rule ID
n	Optional	String	The rule name
d	Optional	String	The rule description
tf	Optional	CriteriaShorthand	The rule event type filter
mf	Optional	String	The rule message filter
haf	Optional	HostIDArray	The rule hosts filter
hsf	Optional	HostSetID	The rule host set filter
sf	Optional	String	The rule severity filter
ea	Optional	String	The email address to notify
result		RuleMetaData	The modified rule

◆ ◆ ◆ CHAPTER 14

udb: CLI Commands for Managing Users and Groups

This chapter describes the commands that you need to use to manage users and groups.

Overview of the udb Commands

The CLI includes the following sets of commands for managing users and groups.

TABLE 14–1 Sets of Commands for User Accounts, Groups, and Logins

CLI Prefix	Description of Command Set
udb.g	Commands for managing user groups.
udb.login udb.logout udb.whoami	Commands for managing login sessions
udb.p	Commands for managing permissions
udb.u	Commands for managing user accounts
udb.sv	Commands for managing session variables.
udb.l	Command for listing all login configurations.

This chapter describes all the commands in each of these sets.

udb.g: Managing User Groups

You can use the `udb.g` commands to define, modify, delete, and list user groups.

TABLE 14–2 Summary of udb.g Commands

Command Name	Description
udb.g.add	Adds a new user group
udb.g.del	Deletes a user group
udb.g.la	Lists all the user groups
udb.g.lo	Retrieves information about the specified user group.
udb.g.lp	Lists the permissions granted to the specified group
udb.g.lu	Lists the users who are members of the specified group
udb.g.mod	Modifies an existing user group

udb.g.add

This command adds a new group.

TABLE 14–3 Arguments and Result for the udb.g.add Command

Argument/Result		Syntax	Description
n	Required	String	The new group name
d	Optional	String	The new group description
hostWrite	Optional	Boolean	Whether the new group has write permission on hosts; default is false
notRuleWrite	Optional	Boolean	Whether the new group has write permission on notification rules; default is false
adminWrite	Optional	Boolean	Whether the new group has write permission on “admin: users and groups;” default is false

TABLE 14–3 Arguments and Result for the udb.g.add Command *(Continued)*

Argument/Result		Syntax	Description
diffWrite	Optional	Boolean	Whether the new group has write permission on comparisons; default is false
diffRun	Optional	String	The hostSet ID for which the new group has execute permission for comparisons. An empty value removes the execute permission on any hostsets. To set this permission for "all" hostsets, clients use the "allhosts" sentinel value.
ua	Optional	UserArray	The new group users
pga	Optional	GroupArray	The new group parent groups
cga	Optional	GroupArray	The new group child groups
result		Group	The new group

udb.g.del

This command deletes the specified group.

Note – Deleting a group does not delete the user accounts in the group. It simply deletes the group as a classification for the user accounts.

TABLE 14–4 Argument for the udb.g.del Command

Argument		Syntax	Description
ID	Required	GroupID	The group ID

udb.g.la

This command lists all the groups defined in the Sun N1 Service Provisioning System software.

TABLE 14–5 Result for the udb.g.la Command

Result	Syntax	Description
result	GroupArray	The groups

udb.g.lo

This command retrieves the specified group.

TABLE 14–6 Argument and Result for the udb.g.lo Command

Argument/Result		Syntax	Description
ID	Required	GroupID	The group ID
result		Group	The group

udb.g.lp

This command lists the permissions granted to a group

TABLE 14–7 Argument and Result for the udb.g.lp

Argument/Result		Syntax	Description
ID	Required	GroupID	The group ID
result		PermissionArray	The permissions

udb.g.lu

This command lists the members of the specified group

TABLE 14–8 Argument and Result for the udb.g.lu Command

Argument/Result		Syntax	Description
ID	Required	GroupID	The group ID
result		UserArray	The users

udb.g.mod

This command modifies an existing group. Omitted arguments are overwritten.

TABLE 14–9 Arguments and Result for the udb.g.mod Command

Argument/Result		Syntax	Description
ID	Required	GroupID	The group ID
n	Optional	String	The new group name
d	Optional	String	The new group description
hostWrite	Optional	Boolean	Whether the new group has write permission on hosts
notRuleWrite	Optional	Boolean	Whether the new group has write permission on notification rules
adminWrite	Optional	Boolean	Whether the new group has write permission on "admin: users and groups"
diffWrite	Optional	Boolean	Whether the new group has write permission on comparisons
diffRun	Optional	String	The hostSet ID for which the new group has execute permission for comparisons. An empty value removes the execute permission on any hostsets. To set this permission for "all" hostsets, clients use the "allhosts" sentinel value.
ua	Optional	UserArray	The new group users
pga	Optional	GroupArray	The new group parent groups
cga	Optional	GroupArray	The new group child groups
result		Group	The modified group

udb.u: Managing User Accounts

You can use the `udb.u` commands to manage individual user accounts.

TABLE 14–10 Summary of udb.u Commands

Command Name	Description
udb.u.add	Adds a new user account
udb.u.cp	Changes the password of the specified user
udb.u.la	Lists all user accounts
udb.u.lo	Retrieves information about the specified user.
udb.u.lp	Lists the permissions granted to the specified user
udb.u.mod	Modifies the specified user account

udb.u.add

This command adds a new user.

TABLE 14–11 Arguments and Result for the udb.u.add Command

Argument		Syntax	Description
nu	Required	String	The user name of the new user
np	[O/R]	String	The plaintext password for the new user; required if an encoded password is not available or supplied.
nep	[O/R]	String	The encoded password for the new user; required if a plaintext password is not available or supplied.
ng	Optional	GroupArray	The user groups for the new user
hide	Optional	Boolean	Whether the user is set to hidden, default false

TABLE 14–11 Arguments and Result for the udb.u.add Command *(Continued)*

Argument		Syntax	Description
loginConfig	[O/R]	String	Login configuration to use for this user; default is "internal," if available, otherwise required
result		User	The new user

udb.u.cp

This command changes the password of the specified user.

TABLE 14–12 Arguments for the udb.u.cp Command

Argument		Syntax	Description
un	Required	String	The user name of the user whose password should be changed.
op	[O/R]	String	The old plaintext password.
oep	[O/R]	String	The old encoded password.
np	[O/R]	String	The new plaintext password.
nep	[O/R]	String	The new encoded password.

udb.u.la

This command lists all user accounts.

TABLE 14–13 Argument and Result for the udb.u.la Command

Argument/Result		Syntax	Description
sh	Optional	Boolean	Whether hidden users are shown, default false
result		UserArray	The users

udb.u.lo

The `udb.u.lo` command retrieves the specified user.

TABLE 14–14 Argument and Result for the udb.u.lo Command

Argument/Result		Syntax	Description
ID	Required	UserID	The user ID
result		User	The user

udb.u.lp

This command lists the permissions granted to a user.

TABLE 14–15 Argument/Result for the udb.u.lp Command

Argument/Result		Syntax	Description
ID	Required	UserID	The user ID
result		PermissionArray	The permissions

udb.u.mod

This command modifies an existing user; omitted arguments are overwritten.

TABLE 14–16 Argument/Result for the udb.u.mod Command

Argument/Result		Syntax	Description
ID	Required	UserID	The user ID
np	Optional	String	The new plaintext password for the user, cannot be used in conjunction with the an encoded password
nep	Optional	String	The new encoded password for the user, cannot be used in conjunction with the a plaintext password

TABLE 14–16 Argument/Result for the udb.u.mod Command *(Continued)*

Argument/Result		Syntax	Description
ng	Optional	GroupArray	The new user groups for the user
hide	Optional	Boolean	Whether the user is set to hidden
active	Optional	Boolean	Whether the user is set to active
forceFlush	Optional	Boolean	True means flush the user's session variables, if needed, false means abort the modification. Defaults to false.
loginConfig	Optional	String	The new login configuration for the user
result		User	The modified user

udb.sv: Managing Session Variables

You can use the `udb.sv` commands to manage session variables.

TABLE 14–17 Summary of udb.sv Commands

Command Name	Description
udb.sv.add	Adds a new session variable.
udb.sv.del	Deletes a session variable.
udb.sv.fl	Flushes all of a user's session variables.
udb.sv.la	Lists all session variables.
udb.sv.lo	Retrieves information about the session variable.
udb.sv.mod	Modifies the specified session value.
udb.sv.re	Reencrypts all of a user's session variables.

udb.sv.add

This command adds a new session variable (a password must be set using the `-p` parameter if variables are to be persisted).

Note – If you are logged in to the HTML user interface and you add a session variable through the CLI, the session variable name will display without the value when you refresh the list of variables. To display the new session variable's value, log out of the HTML user interface and log back in.

TABLE 14–18 Arguments and Result for the udb.sv.add Command

Argument		Syntax	Description
name	Required	String	The new session variable name
secure	Optional	Boolean	Whether or not the value should be displayed; true means no; default false
desc	Optional	String	The new session variable value description
value	Required	String	The new session variable value for this user. If the value for the variable is an empty string, enter: `- value ""`
result		SessionVariable	The new session variable

udb.sv.del

This command deletes a session variable.

TABLE 14–19 Arguments for the udb.sv.del Command

Argument		Syntax	Description
name	Required	String	The name of the session variable to delete

udb.sv.fl

This command flushes all of a user's session variables.

TABLE 14–20 Arguments for the udb.sv.fl Command

Argument		Syntax	Description
u	Required	String	The name of the user
p	[O/R]	String	The plaintext password for the user
ep	[O/R]	String	The encoded password for this user

udb.sv.la

This command lists all session variables.

TABLE 14–21 Argument and Result for the udb.sv.la Command

Argument/Result	Syntax	Description
result	SessionVariableSet	The variables available to this user

udb.sv.lo

This command retrieves the specified session variable

TABLE 14–22 Argument and Result for the udb.sv.lo Command

Argument/Result		Syntax	Description
name	Required	String	The name of the session variable to show
result		SessionVariable	The session variable

udb.sv.mod

This command modifies a session variable; a password must be set using the -p parameter if variables are to be persisted.

TABLE 14–23 Argument/Result for the udb.sv.mod Command

Argument/Result		Syntax	Description
name	Required	String	The name of the session variable to modify
secure	Optional	String	Whether or not the value should be displayed; true means no; default false
desc	Optional	String	The new session variable description
value	Optional	String	The new session variable value for this user
result		SessionVariable	The new session variable

udb.sv.re

This command reencrypts all of a user's session variables.

TABLE 14–24 Arguments for the udb.sv.re Command

Argument		Syntax	Description
u	Required	String	The name of the user
p	[O/R]	String	The plaintext password for the user
ep	[O/R]	String	The encoded password for the user
op	[O/R]	String	The old plaintext password used to encrypt these variables
oep	[O/R]	String	The old encoded password used to encrypt these variables

Authentication Commands

udb.login

Logs in a user and returns a SessionID that can be used for authentication. To send the session ID to a file, the arguments `-o` and `-of` must be specified before the username and password. The usage of `-o` and `-of` options is shown inTable 1–1

TABLE 14–25 Result of the udb.login Command

Argument		Syntax	Description
u	Required	String	The username
p	[O/R]	String	The plaintext user password; required if the encoded password is not available or supplied.
ep	[O/R]	String	The encoded user password; required if the plaintext password is not available or supplied.
result		SessionID	The session ID

EXAMPLE 14–1 Example of Sending the Session ID to a File

This example demonstrates how to save a session ID for reuse.

Name of formatter `serialized`

Name of output file `sessionid`

```
# cr_cli -cmd udb.login -o serialized -of sessionid -u admin -p admin
```

udb.logout

This command logs out the user who runs it. To logout from a session using the CLI command cr_cli, the —s (session id) argument needs to be specified. The session id is returned after a successful execution of udb.login command. Please refer to the udb.login command on how to save a session id to a file.

```
# cr_cli -cmd udb.logout -s session_id
```

Note – The —s parameter is not required if the command is run inside a Jython shell (clij).

udb.whoami

This command returns the owner of the current session.

TABLE 14–26 Result of the udb.whoami Command

Result	Syntax	Description
result	UserID	The current user ID

udb.p: Commands for Managing Permissions

The `udb.p` commands enable you to display information about the permissions established in the Sun N1 Service Provisioning System software.

TABLE 14–27 Summary of the udb.p Commands

Command	Description
udb.p.la	Lists all permissions.
udb.p.lo	Retrieves the specified permission.

udb.p.la

This command lists all permissions.

TABLE 14–28 Result for the udb.p.la Command

Result	Syntax	Description
result	PermissionArray	The permissions

udb.p.lo

This command retrieves the specified permission..

TABLE 14–29 Argument and Result for the udb.p.lo Command

Argument/Result		Syntax	Description
ID	Required	PermissionID	The permission ID
result		Permission	The permission

udb.l: Managing Login Configurations

udb.l.la

This command lists all of the login configurations.

TABLE 14–30 Result for the udb.l.la Command

Argument	Syntax	Description
result	LoginConfiguration-Array	The list of login configurations

◆ ◆ ◆ CHAPTER 15

util: Miscellaneous CLI Commands

This chapter describes some miscellaneous commands that can be used to show the version of the Master Server application, formats an object, and perform Perl-type and XSLT transformations on a file.

Overview of the util Commands

The following table summarizes the CLI commands described in this chapter.

TABLE 15–1 Summary of the util Commands

Command	Description
util.msv	Show the version of the Master Server application.
util.reformat	Format an object.
util.xfm	Transform a serialized object representation to a human-readable representation.

util.msv

This command shows the version of the Master Server application.

TABLE 15–2 Arguments and Result for the util.msv Command

Result	Syntax	Description
result	String	The version of the Master Server application

util.reformat

This command formats an object.

The following list the command-specific output formats that are available to specific commands:

cdb.detail	Detailed text output of components
cmp.detail	Detailed text output of comparisons
fdb.detail	Detailed text output of folders
hdb.detail	Detailed text output of host data
net.detail	Detailed text output of network data
net.summary	Summarized output of network data
pdb.detail	Detailed text output of plans
pe.detail	Detailed text output from the running of plans
plg.detail	Detailed text output of plug-ins
rule.detail	Detailed text output of notification rules
udb.deep	Very detailed text output describing users and user groups
udb.detail	Detailed text output describing users and user groups
udb.summary	Summarized text output describing users and user groups

TABLE 15–3 Argument and Result for the util.reformat Command

Argument/Result		Syntax	Description
self	Required	Object	The argument and result
result		Object	The argument unaltered

util.xfm

This command is used to transform a serialized object representation to a human-readable representation.

TABLE 15–4 Arguments for the util.xfm Command

Argument		Syntax	Description
in	Required	ReaderWrapper	The input stream to be transformed

TABLE 15–4 Arguments for the util.xfm Command *(Continued)*

Argument		Syntax	Description
type	Required	TransformType	The type of transformation to apply to the input stream
xfm	Required	ReaderWrapper	The transformed stream

EXAMPLE 15–1 Using the util.xfm Command

This example performs a PERL transformation, which is described in the perl.xml file, on the text specified in the in.txt file. The result is written to standard output. You can use the -of argument to redirect the standard output to a file.

```
cr_cli -cmd util.xfm -in in.txt -type PERL -xfm perl.xml -u terry -p 123xyz
```

Since the class of the input stream is ReaderWrapper, you can specify any syntax supported by the String2ReaderWrapper converter to specify each stream's source. For example, to read input from standard input instead of from the in.txt file, use -in - instead of -in in.txt.

◆ ◆ ◆ APPENDIX A

Input Types

The following table describes the syntax of the input argument types that can be passed to a CLI command.

In addition to the typographic conventions described in the Preface, text that is surrounded by square brackets ([]) is optional.

TABLE A–1 Syntax of CLI Input Types

Input Type	Syntax
`AppInstance`, `AppInstanceID`	One of the following: ■ `NM:`*host-name*`:RA` ■ `NM:`*host-name*`:LD` ■ `NM:`*host-name*`:MS` ■ `[ID:]`*ID*
`AppTypeCriteria`	Comma-separated or pipe-separated list of the following values: ■ `RA` ■ `LD` ■ `MS` Or: `empty`

TABLE A–1 Syntax of CLI Input Types *(Continued)*

Input Type	Syntax
`AttributeCriteriaList`	*attribute-criteria*`[;`*attribute-criteria*`]*` Where *attribute-criteria* is one of the following: ■ *attribute-name*`-contains-`*glob-pattern-value* ■ *attribute-name*`-equals-`*glob-pattern-value* Where *attribute-name* is one of the following values: `sys.hostName`, `sys.description`, `sys.hostType`, `sys.ipAddress`, `sys.parent`, `sys.OS`, `sys.OSVersion`, `sys.OSArch`, or *user-defined-name* Where *glob-pattern-value* is a string that contain the wildcard characters * and .
`Boolean`	The following case-insensitive values for true and false: ■ `true`, `yes`, `t`, or `+` ■ `false`, `no`, `f` or `-`
`Category`, `CategoryID`, `CategoryIDSet`	One of the following: ■ `NM:`*category-name* ■ `[ID:]`*ID*
`CategoryArray`	*CategoryArray* Where *CategoryArray* is one of the following: ■ *Category*`[,`*Category*`]*` ■ *CategoryID*`[,`*CategoryID*`]*` ■ *CategoryIDSet*`[,`*CategoryIDSet*`]*` Examples: ■ `H:NM:cat1` ■ `HS:NM:cat1,H:NM:cat2;H:NM:cat3`
`CompCheckInID`	`[ID:]`*ID*
`ComponentTypeRefID`	One of the following: ■ `NM:`*component-type-ref-name* ■ `[ID:]`*ID*
`Component`, `ComponentID`	One of the following: ■ `NM:`*component-name*`[:`*version*`]` If *version* is not specified, the latest version is used ■ `[ID:]`*ID*

TABLE A–1 Syntax of CLI Input Types *(Continued)*

Input Type	Syntax
`ComponentVariableSettings`, `ComponentVariableSettingsID`	One of the following: ■ `NM:`*component-name*`[:`*version*`]:`*var-settings-name* If *version* is not specified, the latest version is used ■ `[ID:]`*ID*
`ConnectionType`	One of the following: ■ `raw` ■ `ssh` ■ `ssl`
`CriteriaShorthand`	One of the following: ■ `any` ■ `planStart` ■ `planEndNormal` ■ `planEndAbnormal` ■ `diffStart` ■ `diffEndNormal` ■ `diffEndAbnormal` ■ `system` ■ `admin` ■ `custom`
`DifferenceSettings`, `DifferenceSettingsID`	One of the following: ■ `NM:`*diff-name* ■ `[ID:]`*ID*
`ExecutionPlan`, `ExecutionPlanID`	One of the following: ■ `NM:`*plan-name*`[:`*version*`]` If *version* is not specified, the latest version is used ■ `[ID:]`*ID*
`Folder`, `FolderID`	One of the following: ■ `NM:`*folder-name* ■ `[ID:]`*ID*
`Group`, `GroupID`	One of the following: ■ `NM:`*group-name* ■ `[ID:]`*ID*

TABLE A–1 Syntax of CLI Input Types *(Continued)*

Input Type	Syntax
`Hashtable`	*key*`[=`*value*`][;`*key*`[=`*value*`]]*` Where *key* is a keyword Where *value* is one of the following: ■ *single-value* ■ `[`*single-value*`,`*single-value*`[,`*single-value*`]*]` The outermost square brackets are required Examples: ■ `color=red;emptykey` ■ `arraykey=[value1,value2]`
`Host`, `HostID`	One of the following: ■ `NM:`*host-name* ■ `[ID:]`*ID*
`HostIDArrayArray`	*host-ID-array*`[;`*host-ID-array*`]*` Where *host-ID-array* is one of the following: ■ *host-ID*`[,`*host-ID*`]*` ■ *host-set-ID*`[,`*host-set-ID*`]*` Where *host-ID* is `H:`*host-ID* and *host-set-ID* is `HS:`*host-set-ID* Examples: ■ `H:NM:host1` ■ `HS:NM:hostSetFoo,H:NM:h1;H:NM:h2`
`HostIDSet`	*host-ID*`[,`*host-ID*`]*`
`HostSearch`, `HostSearchID`	One of the following: ■ `NM:`*host-search-name* ■ `[ID:]`*ID*
`HostSearchIDSet`	*host-search-ID*`[,`*host-search-ID*`]*`
`HostSet`, `HostSetID`	One of the following: ■ `NM:`*host-set-name* ■ `[ID:]`*ID*
`HostSetIDSet`	*host-set-ID*`[,`*host-set-ID*`]*`
`HostType`. `HostTypeID`	One of the following: ■ `NM:`*host-type-name* ■ `[ID:]`*ID*
`HostTypeVarList`	*key*`[=`*single-value*`][;`*key*`[=`*single-value*`]]*`

TABLE A–1 Syntax of CLI Input Types *(Continued)*

Input Type	Syntax
`InputStreamWrapper`	One of the following: ■ `-` for standard input ■ *input-file-name*
`InstalledComponentID`	`[ID:]`*ID* The `cdb.ic.lbh` or `cdb.ic.lbc` commands provide the value for *ID*.
`InstalledComponentRef`	*component-name*:*component-version*:*install-path*
`InstalledResourceRef`	*resource-ID*:*install-path*
`Level`	One of the following: ■ `host` ■ `dir` ■ `file`
`NamedBlockType`	One of the following: ■ `install` ■ `uninstall` ■ `call`
`OutputStreamWrapper`	One of the following: ■ `[+]`*filename*, where + means to append ■ `-` for standard output ■ `-:` for standard error

TABLE A–1 Syntax of CLI Input Types *(Continued)*

Input Type	Syntax
`Permission`, `PermissionID`	One of the following: ■ *permID* ■ `NM:`*permID* ■ `NM:userdb.read` ■ `NM:userdb.write` ■ `NM:host.read` ■ `NM:host.write` ■ `NM:hostType.read` ■ `NM:hostType.write` ■ `NM:rule.read` ■ `NM:rule.write` ■ `NM:diff.read` ■ `NM:diff.write` ■ `NM:diffrun:allhosts` ■ `NM:diffrun:`*host-set-name* ■ `NM:folder:`*folder-name*`:write` ■ `NM:folder:`*folder-name*`:checkin-current` ■ `NM:folder:`*folder-name*`:autorun` ■ `NM:folder:`*folder-name*`:execute:allhosts` ■ `NM:folder:`*folder-name*`:execute:`*host-set-name* ■ `NM:folder:`*folder-name*`:owner`
`PhysicalCriteria`	*phys-attribute*`[|`*phys-attribute*`]*` Where *phys-attribute* is `VIRT` or `PHYS` Or: `empty`
`Plugin`, `PluginID`	One of the following: ■ `NM:`*plugin-name* ■ `[ID:]`*ID*
`ReaderWrapper`	One of the following: ■ - for standard input ■ *input-file-name*
`RoxAddress`	One of the following: ■ *IP-name*:*port-number* ■ *IP-address*:*port-number*
`RuleID`, `RuleMetaData`	One of the following: ■ `NM:`*rule-name* ■ `[ID:]`*ID*

TABLE A–1 Syntax of CLI Input Types *(Continued)*

Input Type	Syntax
`Scope`	One of the following: ■ `component` ■ `host` ■ `hostset`
`SeverityArray`	One of the following: ■ `empty` ■ `INFO[+]`, where + means "or greater" ■ `WARNING[+]` ■ `ERROR[+]`
`StepDescriptionContainer`	One of the following: ■ `[ID:]`*stepID* ■ `L:`*label*`:`*taskId*`:`*host* *taskID* can be specified as `[ID:]taskID`. *host* is specified using the `host` and `hostID` input types. Examples: ■ `ID:2011` ■ `L:label1:ID:1001:ID:3001`
`StepDescriptionContainerArray`	*StepDescriptionContainer* `[,`*StepDescriptionContainer*`]*`
`StringArray`	*string*`[,`*string*`]*` Each value is separated by a comma (,). If more than one value exists, all values are contained in quotation marks ("). Examples: ■ `val1` ■ `"appSet1, appset2"`
`StringArrayArray`	*string-array*`[;`*string-array*`]*` Each subplan is separated by a semicolon (;), and each value is separated by a comma (,). If more than one value exists, the values must be contained in quotation marks ("). Examples: ■ `"+,+,-"` ■ `"+;+"` ■ `"val1,val2;+,val4"`
`Style`	One of the following: ■ `mm` for model to model ■ `mi` for model to install ■ `ii` for install to install
`SystemServiceRefID`	One of the following: ■ `NM:`*system-service-ref-name* ■ `[ID:]`*ID*

TABLE A–1 Syntax of CLI Input Types *(Continued)*

Input Type	Syntax
`TimeInterval`	One of the following: ■ - for an indefinite interval ■ @*Date* from now to *Date* Where *Date* is of the form *mm*/*dd*/*yyyy HH*:*MM* ■ [*#weeks*`sw`][*#days*`sd`][*#hours*`sh`][*#minutes*`sm`][*#seconds*`ss`][*#millisecs*`sms`] ■ [*#weeks*`sw`][*#days*`sd`][*#hours*`:`]*#minutes*[`.`*#seconds*[`.`*#milliseconds*]]
`User`, `UserID`	One of the following: ■ `NM:`*user-name* ■ `[ID:]`*ID*
UpgradeTaskID	`[ID:]`*ID*
`WriterWrapper`	One of the following: ■ `[+]`*filename*, where + means to append ■ - for standard output ■ - : for standard error