Router CLI Overview

CradlePoint, Inc.

Preface

CradlePoint reserves the right to revise this publication and to make changes in the content thereof without obligation to notify any person or organization of any revisions or changes.

Manual Revisions

Revision	Date	Description	Author
1.0	Jan. 11, 2013	Initial release for Firmware version 4.1.1	Justin Mayfield

Trademarks

Table of Contents

1 INTRODUCTION

The CradlePoint Router CLI (Command Line Interface) is a captive text-based interface for configuring, managing, and debugging CradlePoint routers. The CLI is accessed through the SSH-2 (Secure Shell v2) protocol, which is made available to users through the router administration pages in the **System Settings → Administration** page under the *Local Management* tab. You can also enable remote access in the *Remote Management* tab of that same page if you wish to use the CLI through the WAN interface of the router.

2 TERMINOLOGY

- **Prompt**: The input field at the bottom of the CLI interface. This is where *commands* and *arguments* are entered into the system.
- **Command**: A single word instruction for the router to process. It is sent to the router as the first word entered at the *command prompt*.
- **Argument:** An argument refers to the text that follows the command. Most commands have optional arguments that control in influence the way the command will run. Arguments for each command can typically be discovered by running the command with the **--help** argument or by running **help COMMAND**.

3 ENABLING ACCESS

The default configuration for most CradlePoint routers has the SSH server in the router disabled. Follow these steps to enable the SSH server so a CLI session can be made.

3.1 Through the Administration Pages

1. Enter the LAN IP address or hostname of the router (assuming you are connecting from the LAN side) into a web browser's location field. By default, these are: http://192.168.0.1 and http://cp/.
2. Click on the *System Settings* tab on the top toolbar and select *Administration*.
3. Click on the *Local Management* tab in the center pane.
4. Check the *Enable SSH Server* entry box.

   Optionally select the *Remote Management* tab and check *Allow Remote SSH Access* if you would like to access the CLI from the Internet.
5. Click *Apply* on the bottom of the page.

3.2 With Curl (Advanced)

1. Run this command from in a terminal or command prompt of your PC and enter the admin password when prompted.

   $ curl -X PUT --digest -uadmin http://192.168.0.1/api/config/firewall/ssh_admin/enabled -d data=true
2. Optionally enable remote access.

   $ curl -X PUT --digest -uadmin http://192.168.0.1/api/config/firewall/ssh_admin/remote_access -d data=true

4 CLIENT SOFTWARE

CradlePoint Series 3 routers will support any standards-based SSH-2 client. If you are using a GNU/Linux, Apple OSX, or other modern operating system you are likely to already have the necessary tools to connect. For these platforms simply locate and execute the terminal emulator of your preference and run the **ssh** command with the username and password of the router as such:

```
$ ssh admin@192.168.0.1
```

For Microsoft Windows computers you can download the free and well-supported PuTTY SSH client. See the PuTTY web page and application help for details on making SSH connections. The only details you should need to provide are the username, password, and IP address of the router. The default username is always **admin**, as shown above.

5 COMMAND PROMPT

Once you successfully log in you will see a command prompt such as: **[admin@MBR1400-301: /]$**. This shows you your username (admin), system ID (MBR1400-301), and current working directory (/). The working directory is important for many commands, as it represents the ConfigStore context for many commands.

6 NAVIGATION

An important part of a command line shell is the notion of a hierarchical file system, which is usually backed by a tree structure of directories and files on a hard disk or SSD storage device. In the case of CradlePoint routers, this file system hierarchy is backed by the routers' configuration, status, and control system known as the *ConfigStore*.

The ConfigStore is like the central nervous system for the router and acts as a live interactive portal to the router's current state and configuration settings. We will expand on this thought as we move forward, but try to remember that the files

and directories that we talk about are live elements of the router. Any changes to those files and directories will result in immediate application and proliferation through the router's internal subsystems. Likewise, when viewing the contents of files, you will always be presented with the most current data.

When you first log in your current working directory will be the *root* directory (/). To change directories, use the **cd** command, which expects a single argument containing the new directory. Combine this with the list command **ls** and the **get** command and you can start exploring the system:

```
[admin@MBR1400-301: /]$ ls
control/        status/        config/

[admin@MBR1400-301: /]$ cd config

[admin@MBR1400-301: /config]$ ls
ethernet/    wan/        lan/        shell/      qos/        dns/
firewall/    failover/   alerts/     system/     hotspot/    wwan/
gre/         routing/    dhcpd/      webfilter/  wlan/       vpn/
stats/

[admin@MBR1400-301: /config]$ cd system

[admin@MBR1400-301: /config/system]$ ls
mac_monitor/              local_domain        timezone
at_passthrough_proxy/     serial/             upnp/
ui_activated              snmp/               first_boot
```

```
dyndns/                 system_id               email/
gps/                    users/                  power/
ntp/                    pci_dss                 wipc/
qxdmproxy/              logging/                admin/
watchdog/               disable_leds            reboot_schedule/
dst_enabled

[admin@MBR1400-301: /config/system]$ cd logging

[admin@MBR1400-301: /config/system/logging]$ ls
console         level           firewall        enabled
max_logs        modem_debug     remoteLogging/

[admin@MBR1400-301: /config/system/logging]$ get level
"info"
```

To change to the parent directory use the special '..' path. Each directory path must be separated with the '/' character. To go up two levels, for example, you would run 'cd ../..'.

The command line interface supports tab-completion, which can reduce the number of keystrokes required to enter a command name or path name. To use this feature simply type the first part of a command or path name and hit the tab key to complete the word. If the word is not unique you can hit the tab key twice to see a list of conflicts.

7 COMMANDS

To see a complete list of commands type **help**. To get detailed help about a specific command simply run **help COMMAND** such as **help ls**. Here is an incomplete list of some basic and important commands:

- **wan**: Show and configure wan devices. Detailed information about a wan can be obtained by providing the UID of the wan device as the first argument.
- **lan**: Show the current LAN configuration and status.
- **wireless**: Show status of all Access Points on the router and connected wireless clients.
- **get**: Get value for config item(s)
- **set**: Set a value to a config item
- **delete**: Delete an item from the config
- **inspect**: Show the data type definitions for a config path. This is useful for learning the requirements of a particular config resource.
- **log**: Show and search through the system log.
- **passwd**: Changes your password.
- **factory_reset**: Reset the config to factory defaults. Use with caution.

8 READING AND WRITING VALUES

The "get" and "set" commands are used to read and write values to the router's ConfigStore. The format for all these values is JSON. Because the file system in the CLI is also the ConfigStore, you can run *get* on the entire ConfigStore if desired. For example if you run *get* from the root directory you will see the complete contents of the router's ConfigStore. The *get*, *set*, *append*, and *delete* commands all take the current working directory into consideration, so the argument that

follows the command should be the path within or with reference to the current directory. Use the *ls* command to view the contents of the current directory when in doubt.

Here are a few examples that illustrate the multiple ways in which you can view the contents of the system.

We can run the get and ls commands from any directory we want if we use an absolute path as the argument. An absolute path is any path that begins with a "/" character; this indicates that the path begins at the root directory.

```
[admin@MBR1400-301: /]$ get /config/system/logging/level
"info"

[admin@MBR1400-301: /]$ ls /config/system/logging
console         level           firewall        enabled
max_logs        modem_debug     remoteLogging/
```

Alternatively, we can change our current working directory to the location of interest and run the commands from there.

```
[admin@MBR1400-301: /]$ cd /config/system/logging/

[admin@MBR1400-301: /config/system/logging]$ ls
console         level           firewall        enabled
max_logs        modem_debug     remoteLogging/

[admin@MBR1400-301: /config/system/logging]$ get level "info"
```

The "." path name refers to the current working directory which can also be used as an argument to the get command. You will notice the output is very similar to the ls commands output, but is formatted as JSON and includes the contents of the sub-directories too.

```
[admin@MBR1400-301: /config/system/logging]$ get .
{
  "console": false,
  "enabled": true,
  "firewall": false,
  "level": "info",
  "max_logs": 1000,
  "modem_debug": false,
  "remoteLogging": {
        "enabled": false,
        "system_id": false,
        "utf8_bom": true
  }
}
```

If we change to the parent directory we can run the get command again but this time we must provide the relative path to the item we would like to see.

```
[admin@MBR1400-301: /config/system/logging]$ cd ..
```

```
[admin@MBR1400-301: /config/system]$ get logging/level        "info"
```

From those examples we can see that the log level is set to "info", which means only messages of this informational priority or higher will be stored in the system log. Let's say that we want to change this value to something else. We can use the *inspect* command to see what the valid values for this config setting are and then use the *set* command to update the value.

```
[admin@MBR1400-301: /]$ cd /config/system/logging/
```

```
[admin@MBR1400-301: /config/system/logging]$ inspect level     Path: /config/system/logging/level
```

Name	Type	Current value	Default Value	Options
level	select	info	info	debug, info, warning, error, critical

We can now see that the default value is "info" and the available options include a "debug" value, which we want to set the router to use. Just to verify that the current value is indeed set to "info", we run the get command prior to our change.

```
[admin@MBR1400-301: /config/system/logging]$ get level "info"
```

Now we simply run the "set" command with the new value. Remember this value MUST be valid JSON, so in this case the string value will be encapsulated in double quotes.

```
[admin@MBR1400-301: /config/system/logging]$ set level "debug"
```

Running the "get" command again verifies that the change was indeed accepted.

```
[admin@MBR1400-301: /config/system/logging]$ get level "debug"
```

As a short example of the internal error handling, we can purposely enter invalid values to test the system's validation procedures.

```
[admin@MBR1400-301: /config/system/logging]$ set level 3.14
Error: invalid value (3.14) at: ['config', 'system', 'logging', 'level']
Reason: invalid option

[admin@MBR1400-301: /config/system/logging]$ set level __not_json__
Invalid value: No JSON object could be decoded

[admin@MBR1400-301: /config/system/logging]$ set level "not_an_option"
Error: invalid value (not_an_option) at: ['config', 'system', 'logging', 'level']
Reason: invalid option
```

http://www.cradlepoint.com/