1. Abstract

This document describes the best common practices characterized by the devices of which the Internet is built, which fulfill the network operator community’s requirements for configuration, management, and debugging user-interfaces. It thereby establishes a baseline of minimum functionality against which proposed device-management interfaces can be evaluated.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC-2119].

The word "device" is used herein to describe any piece of managed network equipment, for instance, an IP router, an Ethernet switch, or a fiber DACS. Network equipment which is not intended to be managed by a human professional, such as end-user-configurable firewall appliances, or cable-modems which are remotely configured by a central device, is outside the scope of this document.

The words "operator" and "operators" are used herein to describe representative members of the community of engineers who professionally design, configure, test, maintain, and debug the networks of which the Internet is built. This explicitly excludes end-user consumers of simple equipment, who are not professionals, and have a separate set of needs, requiring different optimizations.

The words "developer" and "developers" are used herein to describe representative members of the community of programmers who design and implement the configuration and management portions of devices’
operating systems.

The phrase "snippet" is used herein to describe a set of device configuration commands which are valid and sufficient to produce a desired change in the configuration of a device, yet do not constitute the whole configuration of the device. For example, the set of commands which configure a single interface are a snippet.

The phrase "console port" or "craft interface" is used herein to describe an RS-232 serial port with an RJ-45 female form-factor, offering access to the device’s command line via a login prompt at 9600 baud.
4. Introduction

Over the past decade, operators’ configuration practices and the tools and interfaces which developers have been creating have diverged. The result is inefficiency in the practices of operators who have no tools which meet their needs, and waste on the part of developers who bring tools to market which do not meet market needs.

On May 23 and 24, 2001, the Area Directors of Operations and Management held an interim meeting in Scottsdale, in conjunction with the North American Network Operators’ Group, to address and repair the divergence. Operators and developers met, discussed the divergence and its possible remedies, formed a mailing list (ops-nm@ops.ietf.org) for further discussion, and resolved to produce this document, a Best Common Practices RFC, clearly specifying the management interfaces which meet operators’ requirements and establishing a baseline of functionality for developers to implement in devices.

All of the requirements rest upon these basic principles:

- All interaction with devices should be uniform. Devices should not require different protocol, syntax, or encoding depending upon who manufactured the device, who is communicating with it, or upon the medium by which communication is achieved.

- Devices should behave safely and securely. That is, the default mode of operation of any device or portion of a device should be as inert, static, and secure as possible. Deviations from this state should be explicitly enabled by the operator.

- Interfaces should be optimized for worst-case operation, not best-case operation. Best-case is a group of programmers working alone in a fully-stocked lab environment with a testbed network of known and contained parameters and no time pressure. Worst case is a junior operator in the field, crouched in front of a rack of unidentified and undocumented equipment in a hot, dark, noisy machine room, with senior management yelling in both ears about thousands of customers out of service, and only a VT100
serial terminal at hand. Worst-case is far more common than best-case.

Note that this document addresses operators’ requirements of devices as defined above. Thus this document does not describe the needs of people who are not full-time professional configurers of devices, nor does it apply to devices which are not intended to require the attentions of such a professional. Thus consumer devices and the needs of amateur configurers are not addressed in this document.

5. Interactive Communications Methods

The majority of device configuration is performed interactively. That is, an operator logs into the device, and operates it by means of its command line: issuing queries and commands, receiving responses, and logging out when finished. This has far-reaching implications regarding user accounts, authentication, the establishment and tear-down of sessions, and many other issues, which device vendors have, by and large, addressed adequately in current market offerings. There are a small range of behaviors which are worthy of specification, however, due to their criticality or their basic nature.

Every device (where a "device" is a piece of network equipment intended for configuration by professional operators, as defined above) MUST feature a console port. This uniform interface allows the operator to perform initial configuration of the device, endowing it with user accounts, passwords, network interfaces, and so forth. It also provides a point at which the device can be connected to the operator’s out-of-band management network for later troubleshooting when the in-band network is inoperable.

Interaction between the device and the operator MUST be plain ASCII.

6. File Transfer Methods

In addition to interacting with operators through interactive ASCII typing, devices MUST present the appearance of being able to send, receive, and store files. This allows for batch upload of snippets or whole configurations, upgrading of the operating system image, and backup of configurations and other stored resources to central servers. We say "present the appearance" because some devices do not use a host-style file system internally, and this document does not presume to require that they do so, internally.

It is desirable, although not required, that devices be able to serve files to other devices, in order to assist in bootstrapping them.

7. Syntax and Semantics

Operators require that a common configuration language exist between platforms.

For example, an interface SHOULD be named by its position within its host chassis and by any logical subdivision of the physical interface, and the syntax by which it’s named SHOULD be exactly the same regardless of the vendor of the equipment.

The syntax of the management language MUST be portable to different transport methods, and not be coupled explicitly to a single
protocol. That is, one must be able to dictate commands over the telephone, or type them in on a serial port, or via SSH, all in the same syntax.

Reads and writes MUST, wherever possible, utilize the same name-space.

The syntax of data collected from displays of configuration MUST be, wherever possible, the same as the syntax which would be used to perform any corresponding write. An example of this is the command "show configuration" in IOS, which produces a file which can be re-entered into another router of similar hardware configuration to produce an identical configuration. Note that this process could work better in IOS, but the intent is correct. A counter-example is the IOS command "show access-list" which produces text which appears to be configuration, but cannot be re-entered into a router.

Devices MUST, by default, export their _complete_ configuration in plain ASCII. This configuration MUST be sufficient to create an identical environment when loaded into an unconfigured device of the same physical configuration. This configuration MUST contain any passwords or shared secrets associated with the device, in either unencrypted or strongly-encrypted form.

Devices MUST allow operators to associate comments or descriptive text with any object within their configuration, and this text MUST be retained and displayed in full in the saved configuration.

Operators recognize that 100% commonality of configuration language between devices is a non-goal, since some vendors offer unique functionality which requires unique descriptive terminology. However, operators require commonality of configuration language in all portions of device configuration which are common to multiple devices. For instance, no device should require different syntax for the application of an IP address to a VLAN subinterface of a 100Base-T port, as that is a feature shared by devices from many vendors.

8. Sanity-Checking, Commit, and Rollback

Parser interpretation and sanity-checking of entered configurations MUST be performed on an object-by-object basis as typed if the operator is in an object-by-object commit mode. Objects are currently commonly delimited by carriage returns or curly-braces.

Sanity-checking MUST be performed immediately prior to commit-time if in bulk mode, whether config was entered by a typist or file-transfer.

Configuration MUST be automatically revokable if it creates an operating environment which precludes the operator from being able to revoke it manually; in the event that a revoke occurs, object-by-object confirmation/feedback/errors MUST be logged.

If a device developer wishes to implement separate output modes for human and machine interaction, the branching, that is the point at which the code which generates the output differs, should occur as late as possible. Ideally, the difference should simply be one of object-by-object output for machine interaction, versus tabular output of the same data for human interaction.
9. Authentication & Permissions

In-band authentication systems such as RADIUS, kerberos, and the like are very fragile, and MUST be backed up by internal authentication against static user/password lists, else console port access to disconnected devices will fail. RSA/DSA keys should be accepted in lieu of interactively-prompted passwords over any transport which has standardized support for them. This is necessary to support login by automated systems to devices without storing unencrypted password strings within the automated systems.

New, unconfigured devices MUST NOT require or have knowledge of a "default" username or password at the time of the operator’s first login on the console port.

Any management system which results from this work SHOULD include provision for multiple permissions-levels of authenticated access, each with read and write access to different resources within the managed device or system. For instance, some users should have only read access, and only to specific portions of a device.

10. Security Considerations

Once the operator performs initial configuration of the device via its console port, further interaction is often in-band, via protocol across network interfaces. It is critical that such communication utilize cryptographically secure protocols whenever possible. At the time of this writing, the preferred and most widely-deployed protocol for this purpose is Secure Shell version 2, SSH2, and support of SSH2 client (originating side) and server (answering side) is REQUIRED of all device vendors by this document. Other cryptographically secure in-band communications protocols and combinations of protocols exist, and are OPTIONAL, but not recommended. It is the hope of the operator community that device vendors will concentrate their development efforts primarily upon support of a common core functionality. Examples of other cryptographically secure in-band communications protocols are Kerberized-rlogin, SSL, and telnet-over-IPSec.

Regrettably, one operating system vendor still does not include an SSH client in their default distribution, so it is still also REQUIRED that device vendors implement telnet client and server. It is the hope of the operator community that this requirement eventually be deprecated.

Many of the same security concerns hold true for file transfer as for interactive login. Encrypted, authenticated transfers are preferred, but unencrypted, unauthenticated transfers are also acceptable for things like operating system images, which do not contain secrets, can be procured from a trusted source, and can be checksummed to verify integrity. Thus devices MUST implement TFTP and kermit, which allow unauthenticated, unencrypted transfer over IP and serial, respectively, and scp, which provides authenticated and encrypted file transfer over IP. Devices MAY also implement bootp, xmodem, and zmodem. FTP SHOULD NOT be included, as it uses passwords, but passes them across the wire unencrypted.

Vendors MUST log as much information as possible regarding the source interface, neighbor MAC, IP address, et cetera, of access attempts, configuration changes, and operator-initiated state changes. Such logs SHOULD be both stored internally as memory
allows, and written out to an external log server as quickly as possible.

In general, device configurations are attractive targets to anyone who wishes to deny or subvert Internet services. This document recognizes this threat, and attempts to both document the limitations of both current management techniques and those which will be required in the future, as well as attempt to set a higher standard for support of secure management techniques in future devices.

Of principal concern are the observation by third parties of unencrypted configuration or authentication data as it is being transferred in-band across the network; the strength of encryption employed when storing or transmitting passwords; restricting access to device resources on a per-user and per-command basis; and the security of "password recovery" or factory-default-restoration procedures.

All of these are preexisting problem areas which this document attempts to either make explicit or solve.

11. Requirements of Future Devices

As a definition of best common practices, this document has not discussed operators' requirements which are not being fulfilled by products currently available on the market. However, there are a number of requirements which, while not depending on as-yet-undiscovered principles of physics, are not in fact being offered. To wit:

Versioned result codes. There is widespread consensus among the operator community that many results returned by devices through the CLI could be numerically encoded with a leading string of fixed length, which would generally identify both the nature of the following human-readable string, and the version of the result. This would facilitate easier machine parsing of results, and serve as a point of commonality between vendors who might choose to word human-readable strings in different ways. Full exposition of this topic is outside the scope of this document.

There is demand among the operator community for a "smart diff" utility which could take as input two configurations or snippets, and produce a snippet which, when applied to the first configuration, will result in the second configuration.

Operators have expressed a strong demand for a configuration-display mode which explicitly displays all default and assumed configuration parameters and values in full.

Operators desire that devices should support operator-created configuration templates. That is, an operator should be able to define a template for a generic interface, or routing peer, or any other repetitive task. Operators should be able to manually apply templates to whole ranges or lists of objects at a time. This ability is crucial as routers grow to support hundreds or thousands of interfaces or peers. Use of templates will eventually need to be governed by the user permissions regime. For example, some users might have the authority to create new instances of objects by applying a predefined template (turning up new BGP peers using a peer-group). More advanced users would have permission to apply a
template but also make specific individual-line overrides to the
predefined configuration of the template. Superusers would have the
ability to define new templates and destroy old ones.

Operators need to be able to define default configurations for new
resources before they become available or before they’re created.
That is, an operator needs to be able to define the default settings
(DNS servers, for instance) for any DHCP pool, and have those
defaults applied to any new pool which is created on a managed
device, or to any new pool which is created within a specific
address range on a managed device, or to any new pool which is
created by a specific user of a managed device. Similarly, an
operator needs to be able to define default settings (ESF, B8ZS,
frame relay encapsulation, for instance) for new serial interfaces
which may be subsequently inserted into the chassis, or subsequently
defined within a channelized resource.

12. Good Ideas Which Will be Excised

This section contains all of the good ideas which were too detailed
in nature (mostly too implementation-related) to be appropriate in a
BCP. This section will presumably be deleted prior to last call.

Once a connection has been established between the operator and the
device, the device MUST, if it has been configured, prompt the
operator for a login and then for a password. It is RECOMMENDED
that the device not identify itself prior by means of a banner prior
to successful authentication. Once the login has succeeded and any
banners displayed, the operator MUST be presented with a prompt. It
is RECOMMENDED that the prompt begin with an operator-configurable
string corresponding with the "hostname". Unless the width and
height of the operator’s terminal can be positively identified, or
the operator explicitly specifies different height and width, the
device MUST assume that the operator’s terminal displays monospaced
text 80 columns wide and 24 rows tall. Output SHOULD be displayed
one screen at a time, using the equivalent of "more" to allow the
operator to specify when additional data is desired. The "more"
function MUST be overridable with a single interactive command,
applying to the remainder of the session or until negated. When
data has been displayed, the device MUST immediately present a new
prompt to the operator. "Curses" or "menuing" user interfaces MUST
NOT be used, as they present operators with innumerable difficulties
of terminal emulation compatibility, difficulty of automation, and
are generally impossible to operate without a very narrow range of
terminal-emulation devices which are frequently unavailable.
Additionally, such interfaces are particularly vulnerable to serial
noise, and they present particular challenges to operators who use
voice-synthesis terminals.

Whether the physical transport is in-band or via console port, the
developer must provide sufficient input buffer to accept at a
minimum, a maximum-length configuration at line speed without
dropping any characters from the input or resorting to the use of
flow-control signalling. As of the time of this draft some vendors
drop characters from a "pasted" configuration via telnet, ssh, or
most notably, the craft port.

Developers should carefully consider the option of supporting
multiple simultaneous workspaces and separate locking of different
areas of the device configuration. Note that detecting the presence
of conflicting intent between changes made within different
workspaces is non-trivial, and support for multiple simultaneous workspaces or locks should not appear to be present without full consideration of these effects by the developer.

Note that time-delayed snippet applications is a special case of the above.

SMTP should be taken as a model of a good on-the-wire protocol for operational use. It’s easily implemented, whether in code or in scripts. It’s also easily typed manually, for debugging purposes. It contains machine-readable numeric result codes, as well as verbose text descriptions of the results which can be either observed by an operator performing interactive debugging, or communicated by an automated system to someone capable of interpreting them. It requires no special tools of the person attempting the debugging, as they can do so simply by telnetting to a well-known port. The commands are simple enough and memorable enough that a person of average intelligence can use them from memory, in the middle of the night, without enough coffee, and it contains a rudimentary interactive help system which allows an interactive user to determine what commands are available.

If a device-management protocol were to be implemented in a manner similar to SMTP, a "no verbose" command should be included so that automated console implementations could minimize the volume of return traffic from the device.

13. Authors’ Addresses

Bill Woodcock
Packet Clearing House
2355 Virginia Street
Berkeley, California  94709-1315
USA
Phone: +1 415 831 3100
Email: woody@pch.net