FireBrick FB6102

User Manual

FB6000 Versatile Network Appliance

FireBrick FB6102 User Manual

This User Manual documents Software version V1.27.001

Table of Contents

List of Figures

List of Tables

Preface

The FB6000 device is the result of several years of intensive effort to create products based on state of the art processing platforms, featuring an entirely new operating system and IPv6-capable networking software, written from scratch in-house by the FireBrick team. Custom designed hardware, manufactured in the UK, hosts the new software, and ensures FireBrick are able to maximise performance from the hardware, and maintain exceptional levels of quality and reliability.

The result is a product that has the feature set, performance and reliability to handle mission-critical functions, effortlessly handling huge volumes of traffic, supporting thousands of customer connections.

The software is constantly being improved and new features added, so please check that you are reading the manual appropriate to the version of software you are using. This manual is for version V1.27.001.

Chapter 1. Introduction

1.1. The FB6000

1.1.1. Where do I start?

The FB6000 is shipped in a *factory reset* state. This means it has a default configuration that allows the unit to be attached directly to a computer, or into an existing network, and is accessible via a web browser on a known IP address for further configuration.

Besides allowing initial web access to the unit, the factory reset configuration provides a starting point for you to develop a bespoke configuration that meets your requirements.

A printed copy of the QuickStart Guide is included with your FB6000 and covers the basic set up required to gain access to the web based user interface. If you have already followed the steps in the QuickStart guide, and are able to access the FB6000 via a web browser, you can begin to work with the factory reset configuration by referring to Chapter 3.

Initial set up is also covered in this manual, so if you have not already followed the QuickStart Guide, please start at Chapter 2.

Tip

The FB6000's configuration can be restored to the state it was in when shipped from the factory. The procedure requires physical access to the FB6000, and can be applied if you have made configuration changes that have resulted in loss of access to the web user interface, or any other situation where it is appropriate to start from scratch - for example, commissioning an existing unit for a different role, or where you've forgotten an administrative user password. It is also possible to temporarily reset the FB6000 to allow you to recover and edit a broken configuration (though you still need to know the password you had). You can also go back one step in the config.

The remainder of this chapter provides an overview of the FB6000's capabilities, and covers your product support options.

Tip

The latest version of the QuickStart guide for the FB6000 can be obtained from the FireBrick website at : http://www.firebrick.co.uk/pdfs/quickstart-6000.pdf

1.1.2. What can it do?

The FB6000 series of products is a family of high speed ISP/telco grade routers and firewalls providing a range of specific functions.

Key features of the FB6000 family:

- 1U 19" rack mount
- Very low power consumption (typical 20W) - all important with today's power charges in data centres
- Two small fans are the only moving parts for high reliability
- Dual 120/230V AC power feed
- IPv6 built in from the start

- Gigabit performance

1.1.2.1. FB6102 High capacity ping monitoring box

The FB6102 is a specialised device deisgned to perform large scale *ping* monitoring and graphing.

1.1.3. Ethernet port capabilities

The FB6000 has two Ethernet network ports that operate at 1Gb/s. The ports implement auto-negotiation by default, but operation can be fine-tuned to suit specific circumstances. The function of these ports is very flexible, and defined by the device's configuration. The ports implement one or more *interfaces*.

Multiple interfaces can be implemented on a single physical port via support for IEEE 802.1Q VLANs, ideal for using the FB6000 with VLAN-capable network switches. In this case, a single physical connection can be made between a VLAN-capable switch and the FB6000, and with the switch configured appropriately, this physical connection will carry traffic to/from multiple VLANs, and the FB6000 can do Layer 3 processing (routing/firewalling etc.) between nodes on two or more VLANs.

1.1.4. Product variants in the FB6000 series

- **FB6102** High capacity ping monitoring box
- **FB6202** Gigabit L2TP LNS with detailed monitoring of all lines
- **FB6302** Gigabit BGP router
- **FB6402** Gigabit stateful firewall
- **FB6502** Gigabit core VoIP SIP switch for ISTP use
- **FB6602** Mobile GTPv1 GGSN/L2TP gateway

1.2. About this Manual

1.2.1. Version

Every major FB6000 software release is accompanied by a release-specific version of this manual. This manual documents software version V1.27.001 - please refer to Section 4.3 to find out more about software releases, and to see how to identify which software version your FB6000 is currently running.

If your FB6000 is running a different version of system software, then please consult the version of this manual that documents that specific version, as there may be significant differences between the software versions. Also bear in mind that if you are not reading the latest version of the manual (and using the latest software release), references in this manual to external resources, such as the FireBrick website, may be out of date.

You can find the latest revision of a manual for a specific software version on the FB6000 software downloads website [http://www.firebrick.co.uk/software.php?PRODUCT=6000]. This includes the revision history for all software releases.

1.2.2. Intended audience

This manual is intended to guide FB6000 owners in configuring their units for their specific applications. We try to make no significant assumption about the reader's knowledge of FireBrick products, but as might be expected given the target market for the products, it is assumed the reader has a reasonable working knowledge of common IP and Ethernet networking concepts. So, whether you've used FireBrick products for years, or have purchased one for the very first time, and whether you're a novice or a network guru, this Manual sets out to be an easy to read, definitive guide to FireBrick product configuration for all FireBrick customers.

1.2.3. Technical details

There are a number of useful technical details included in the apendices. These are intended to be a reference guild for key features.

1.2.4. Document style

At FireBrick, we appreciate that different people learn in different ways - some like to dive in, hands-on, working with examples and tweaking them until they work the way they want, referring to documentation as required. Other people prefer to build their knowledge up from first principles, and gain a thorough understanding of what they're working with. Most people we suspect fall somewhere between these two learning styles.

This Manual aims to be highly usable regardless of your learning style - material is presented in an order that starts with fundamental concepts, and builds to more complex operation of your FireBrick. At all stages we hope to provide a well-written description of how to configure each aspect of the FireBrick, and - where necessary - provide enough insight into the FireBrick's internal operation that you understand *why* the configuration achieves what it does.

1.2.5. Document conventions

Various typefaces and presentation styles are used in this document as follows :-

- Text that would be typed as-is, for example a command, or an XML attribute name is shown in `monospaced_font`
- Program (including XML) listings, or fragments of listings are shown thus :-

```
/* this is an example program listing*/
printf("Hello World!\n");
```

- Text as it would appear on-screen is shown thus :-

```
This is an example of some text that would
appear on screen.
Note that for documentation purposes additional
line-breaks may be present that would not be in the on-screen text
```

- Notes of varying levels of significance are represented thus (colour schemes may differ depending on signficance) :-

  Note

  This is an example note.

  The significance is identified by the heading text and can be one of : Tip - general hints and tips, for example to point out a useful feature related to the current discussion ; Note - a specific, but not critical, point relating to the surrounding text ; Caution - a potentially critical point that you should pay attention to, failure to do so may result in *loss of data, security issues, loss of network connectivity etc.*

1.2.6. Comments and feedback

If you'd like to make any comments on this Manual, point out errors, make suggestions for improvement or provide any other feedback, we would be pleased to hear from you via e-mail at : `docs@firebrick.co.uk`.

1.3. Additional Resources

1.3.1. Technical Support

Technical support is available, in the first instance, via the reseller from which you purchased your FireBrick. FireBrick provide extensive training and support to resellers and you will find them experts in FireBrick products.

However, before contacting them, please ensure you have :-

- upgraded your FB6000 to the latest version of software (see Section 4.3) and
- are using the latest revision of the manual applicable to that software version and
- have attempted to answer your query using the material in this manual

Many FireBrick resellers also offer general IT support, including installation, configuration, maintenance, and training. You may be able to get your reseller to develop FB6000 configurations for you - although this will typically be chargeable, you may well find this cost-effective, especially if you are new to FireBrick products.

If you are not satisfied with the support you are getting from your reseller, please contact us [http://www.firebrick.co.uk/contact.php].

1.3.2. IRC Channel

A public IRC channel is available for FireBrick discussion - the IRC server is `irc.z.je`, and the channel is `#firebrick`.

1.3.3. Application Notes

FireBrick are building a library of Application Note documents that you can refer to - each Application Note describes how to use and configure a FireBrick in specific scenarios, such as using the device in a multi-tenant Serviced Office environment, or using the FireBrick to bond multiple WAN connections together.

1.3.4. White Papers

FireBrick White Papers cover topics that deserve specific discussion - they are not related to specific Applications, rather they aim to educate interested readers regarding networking protocols, common/best practice, and real-world issues encountered.

1.3.5. Training Courses

FireBrick provide training courses for the FB2x00 series products, and also training course on general IP networking that are useful if you are new to networking with IP.

To obtain information about upcoming courses, please contact us via e-mail at : `training@firebrick.co.uk`.

Chapter 2. Getting Started

2.1. IP addressing

You can configure your FireBrick using a web browser - to do this, you need IP connectivity between your computer and the FireBrick. For a new FB6000 or one that has been factory reset, there are three methods to set this up, as described below - select the method that you prefer, or that best suits your current network architecture.

- *Method 1* - use the FireBrick's DHCP server to configure a computer.

  If your computer is already configured (as many are) to get an IP address automatically, you can connect your computer to port 1 on the FireBrick, and the FireBrick's inbuilt DHCP server should give it an IPv4 and IPv6 address.

- *Method 2* - configure a computer with a fixed IP address.

  Alternatively, you can connect a computer to port 1 on the FireBrick, and manually configure your computer to have the fixed IP address(es) shown below :-

  Table 2.1. IP addresses for computer

IPv6	IPv4
`2001:DB8::2/64`	`10.0.0.2` ; subnet mask : `255.255.255.0`

- *Method 3* - use an existing DHCP server to configure the FireBrick.

  If your LAN already has a DHCP server, you can connect port 4 of your FireBrick to your LAN, and it will get an address. Port 4 is configured, by default, not to give out any addresses and as such it should not interfere with your existing network. You would need to check your DHCP server to find what address has been assigend to the FB6000.

2.2. Accessing the web-based user interface

If you used Method 1, you should browse to the FireBrick's web interface as follows, or you can use the IP addresses detailed:-

Table 2.2. IP addresses to access the FireBrick

URL
`http://my.firebrick.co.uk/`

If you used Method 2, you should browse to the FireBrick's IP address as listed below:-

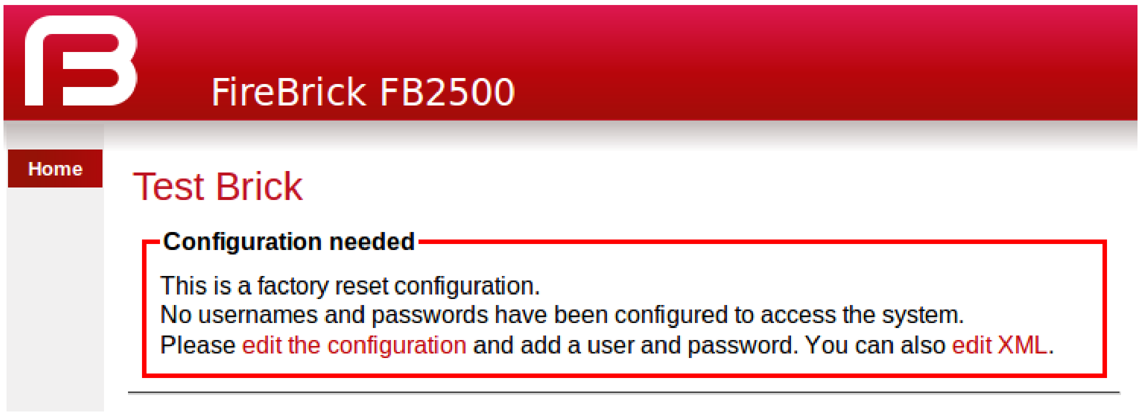
Table 2.3. IP addresses to access the FireBrick

IPv6	IPv4
`http://[2001:DB8::1]`	`http://10.0.0.1`

If you used Method 3, you will need to be able to access a list of allocations made by the DHCP server in order to identify which IP address has been allocated to the FB6000, and then browse this address from your computer. If your DHCP server shows the client name that was supplied in the DHCP request, then you will see FB6000 in the client name field (assuming a factory reset configuration) - if you only have one FB6000 in factory reset state on your network, then it will be immediately obvious via this client name. Otherwise, you will need to locate the allocation by cross-referring with the MAC address range used by the FB6000 you are interested in - if necessary, refer to Appendix B to see how to determine which MAC address you are looking for in the list of allocations.

Once you are connected to the FB6000, you should see a page with "Configuration needed" prominently displayed, as shown below :-

Figure 2.1. Initial web page in factory reset state



Test Brick

Configuration needed

This is a factory reset configuration.
No usernames and passwords have been configured to access the system.
Please edit the configuration and add a user and password. You can also edit XML.

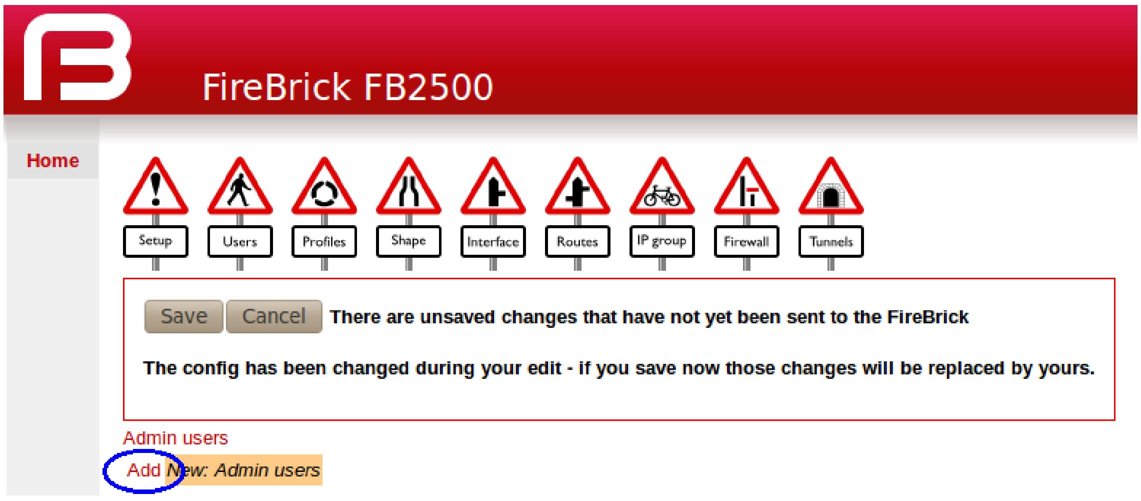
Click on the "edit the configuration" link (red text), which will take you to the main user interface page for managing the configuration.

2.2.1. Add a new user

You now need to add a new user with a password in order to gain full access to the FireBrick's user interface.

Click on the "Users" icon, then click on the "Add" link to add a user. The "Users" page is shown below, with the "Add" link highlighted:-

Figure 2.2. Initial "Users" page



There are unsaved changes that have not yet been sent to the FireBrick

The config has been changed during your edit - if you save now those changes will be replaced by yours.

Admin users

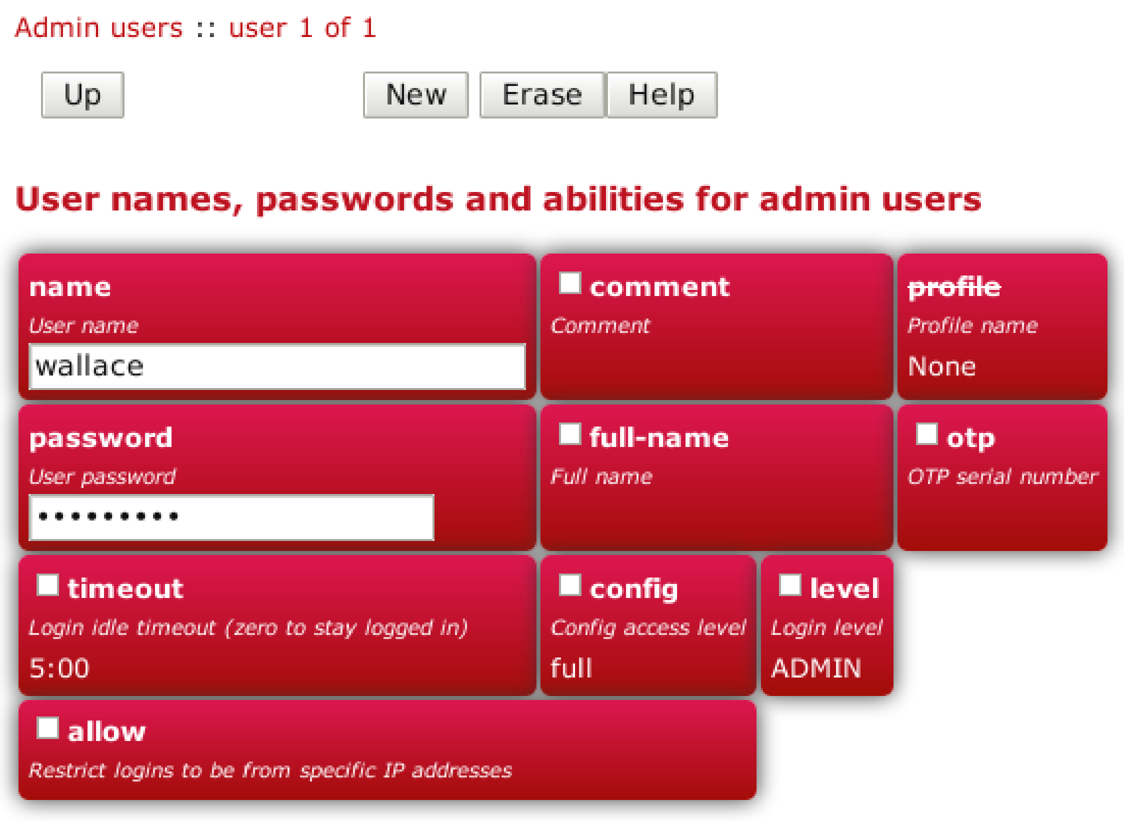
Add New: Admin users

Enter a suitable username in the "Name" box, and enter a password (passwords are mandatory), as shown below. Leave all other checkboxes un-ticked, but see the Tip below regarding the `timeout` setting.

Note

Take care to enter the password carefully, as the FB6000 does not prompt you for confirmation of the password.

Figure 2.3. Setting up a new user



User names, passwords and abilities for admin users

name
User name
wallace

☐ comment
Comment

~~profile~~
Profile name
None

password
User password
••••••••••

☐ full-name
Full name

☐ otp
OTP serial number

☐ timeout
Login idle timeout (zero to stay logged in)
5:00

☐ config
Config access level
full

☐ level
Login level
ADMIN

☐ allow
Restrict logins to be from specific IP addresses

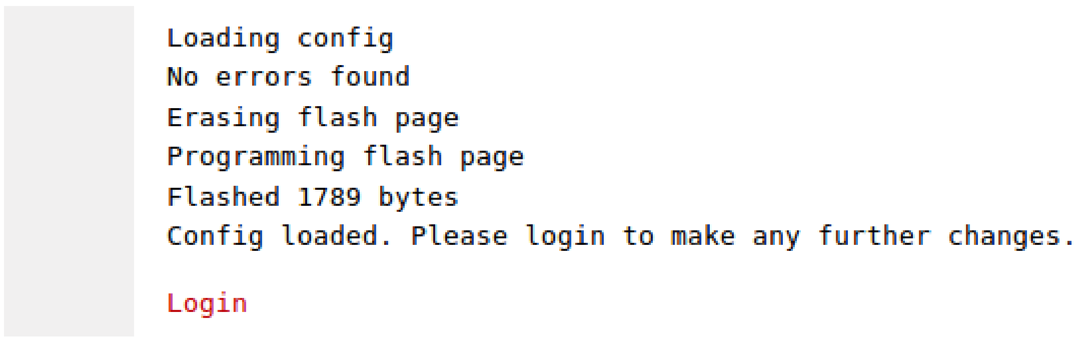
Tip

You may also want to increase the login-session idle time-out from the default of 5 minutes, especially if you are unfamiliar with the user-interface. To do that, tick the checkbox next to `timeout`, and enter an appropriate value as minutes, colon, and seconds, e.g. `15:00` for 15 minutes.

Click on the Save button near the top of the screen which will save a new configuration that includes your new user definition.

You should now see a page showing the progress of storing the new configuration in Flash memory :-

Figure 2.4. Configuration being stored



```
Loading config
No errors found
Erasing flash page
Programming flash page
Flashed 1789 bytes
Config loaded. Please login to make any further changes.

Login
```

On this page there is a "Login" link (in red text)- click on this link and then log in using the username and password you chose.

We recommend you read Chapter 3 to understand the design of the FB6000's user interface, and then start working with your FB6000's factory reset configuration. Once you are familiar with how the user interface is structured, you can find more detail on setting up users in Section 4.1.

Chapter 3. Configuration

3.1. The Object Hierarchy

The FB6000 has, at its core, a configuration based on a *hierarchy of objects*, with each object having one or more *attributes*. An object has a *type*, which determines its role in the operation of the FB6000. The values of the attributes determine how that object affects operation. Attributes also have a *type* (or *datatype*), which defines the type of data that attribute specifies. This in turn defines what the valid syntax is for a value of that datatype - for example some are numeric, some are free-form strings, others are strings with a specific format, such as a dotted-quad IP address. Some examples of attribute values are :-

- IP addresses, and subnet definitions in CIDR format e.g. 192.168.10.0/24
- free-form descriptive text strings, e.g. a name for a firewall rule
- Layer 4 protocol port numbers e.g. TCP ports
- data rates used to control traffic shaping
- enumerated values used to control a feature e.g. defining Ethernet port LED functions

The object hierarchy can be likened to a family-tree, with relationships between objects referred to using terms such as Parent, Child, Sibling, Ancestor and Descendant. This tree-like structure is used to :-

- group a set of related objects, such as a set of firewall rules - the parent object acts as a container for a group of (child) objects, and may also contribute to defining the detailed behaviour of the group
- define a context for an object - for example, an object used to define a locally-attached subnet is a child of an object that defines an interface, and as such defines that the subnet is accessible on that specific interface. Since multiple interfaces can exist, other interface objects establish different contexts for subnet objects.

Additional inter-object associations are established via attribute values that reference other objects, typically by name, e.g. a firewall rule can specify one of several destinations for log information to be sent when the rule is processed.

3.2. The Object Model

The term 'object model' is used here to collectively refer to :-

- the constraints that define a valid object hiearchy - i.e. which object(s) are valid child objects for a given parent object, how many siblings of the same type can exist etc.
- for each object type, the allowable set of attributes, whether the attributes are mandatory or optional, their datatypes, and permissible values of those attributes

The bulk of this User Manual therefore serves to document the object model and how it controls operation of the FB6000.

Tip

This version of the User Manual may not yet be complete in its coverage of the full object model. Some more obscure attributes may not be covered at all - some of these may be attributes that are not used under any normal circumstances, and used only under guidance by support personnel. If you encounter attribute(s) that are not documented in this manual, please refer in the first instance to the documentation described in Section 3.2.1 below. If that information doesn't help you, and you think the attribute(s) may be relevant to implementing your requirements, please consult the usual support channel(s) for advice.

3.2.1. Formal definition of the object model

The object model has a *formal* definition in the form of an XML Schema Document (XSD) file, which is itself an XML file, normally intended for machine-processing. A more readable version of this information is available in Appendix F.

Note, however, that this is *reference* material, containing only brief descriptions, and intended for users who are familiar with the product, and in particular, for users configuring their units primarily via XML.

The XSD file is also available on the software downloads website by following the "XSD" link that is present against each software release.

3.2.2. Common attributes

Most objects have a `comment` attribute which is free-form text that can be used for any purpose. Similarly, most objects have a `source` attribute that is intended for use by automated configuration management tools. Neither of these attributes have a direct effect on the operation of the FB6000.

Many objects have a `name` attribute which is non optional and often needs to be unique within the list of object. This allows the named object to be referenced from other attributes. The data type for these is typically an *NMTOKEN* which is a variant of a *string* type that does not allow spaces. If you include spaces then they are removed automatically. This helps avoid any problems referencing names in other places especially where the reference may be a space separated list.

Many objects have a `graph` attribute. This allows a graph name to be specified. However, the actual graph name will be *normalised* to avoide spaces and limit the number of characters. Try to keep graph names as basic characters (letters, numbers) to avoid confusion.

3.3. Configuration Methods

The configuration objects are created and manipulated by the user via one of two configuration methods :

- web-based graphical User Interface accessed using a supported web-browser
- an XML (eXtensible Markup Language) file representing the entire object hierarchy, editable via the web interface or can be uploaded to the FB6000

The two methods operate on the same underlying object model, and so it is possible to readily move between the two methods - changes made via the User Interface will be visible as changes to the XML, and vice-versa. Users may choose to start out using the User Interface, and - as experience with the object model and the XML language develops - increasingly make changes in the XML environment. For information on using XML to configure the FB6000, please refer to Section 3.5.

3.4. Web User Interface Overview

This section provides an overview of how to use the web-based User Interface. We recommend that you read this section if you are unfamiliar with the FB6000, so that you feel comfortable with the design of the User Interface. Later chapters cover specific functionality topics, describing which objects are relevant, any underlying operational principles that are useful to understand, and what effect the attributes (and their values) have.

The web-based User Interface provides a method to create the objects that control operation of the FB6000. Internally, the User Interface uses a formal definition of the object model to determine where (in the hierarchy) objects may be created, and what attributes may exist on each object, so you can expect the User Interface to always generate valid XML. [1]

[1] If the User Interface does not generate valid XML - i.e. when saving changes to the configuration the FireBrick reports XML errors, then this may be a bug - please check this via the appropriate support channel(s).

Additionally, the web User Interface provides access to the following items :-

- status information, such as DHCP server allocations, FB105 tunnel information and system logs
- network diagnostic tools, such as Ping and Traceroute ; there are also tools to test how the FB6000 will process particular traffic, allowing you to verify your firewalling is as intended
- traffic graphs

By default, access to the web user interface is available to all users, from any IP address. If you don't require such open access, you may wish to restrict access using the settings described in Section 10.3.

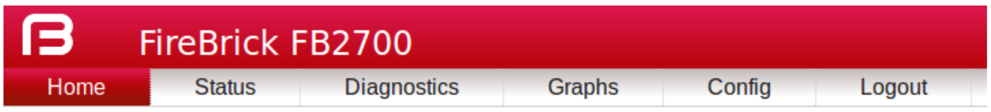
3.4.1. User Interface layout

The User Interface has the following general layout :-

- a 'banner' area at the top of the page, containing the FireBrick logo, model number and system name
- a main-menu, with sub-menus that access various parts of the user interface ; the main-menu can be shown vertically or horizontally - sub-menu appearance depends on this display style : if the main-menu is vertical, sub-menus are shown by 'expanding' the menu vertically ; if the main-menu is horizontal, sub-menus are shown as pull-down menus
- a 'footer' area at the bottom of the page, containing layout-control icons and showing the current software version
- the remaining page area contains the content for the selected part of the user-interface

Figure 3.1 shows the main menu when it is set to display horizontally. Note that the main-menu items themselves have a specific function when clicked - clicking such items displays a general page related to that item - for example, clicking on Status shows some overall status information, whereas sub-menu items under Status display specific categories of status information.

Figure 3.1. Main menu



The user interface pages used to change the device configuration are referred to as the 'config pages' in this manual - these pages are accessed by clicking on the "Edit" item in the sub-menu under the "Config" main-menu item.

Note

The config pages utilise JavaScript for their main functionality ; you must therefore have JavaScript enabled in your web browser in order to configure your FB6000 using the web interface.

3.4.1.1. Customising the layout

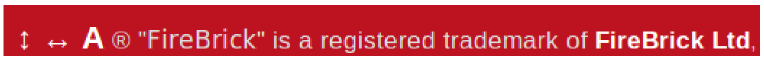
The following aspects of the user interface layout can be customised :-

- The banner area can be reduced in height, or removed all together
- The main menu strip can be positioned vertically at the left or right-hand sides, or horizontally at the top (under the banner, if present)

Additionally, you can choose to use the default fonts that are defined in your browser setup, or use the fonts specified by the user interface.

These customisations are controlled using three icons on the left-hand side of the page footer, as shown in Figure 3.2 below :-

Figure 3.2. Icons for layout controls


↕ ↔ A ® "FireBrick" is a registered trademark of FireBrick Ltd,

The first icon, an up/down arrow, controls the banner size/visibility and cycles through three settings : full size banner, reduced height banner, no banner. The next icon, a left/right arrow, controls the menu strip position and cycles through three settings : menu on the left, menu on the right, menu at the top. The last icon, the letter 'A', toggles between using browser-specified or user-interface-specified fonts.

Layout settings are stored in a cookie - since cookies are stored on your computer, and are associated with the DNS name or IP address used to browse to the FB6000, this means that settings that apply to a particular FB6000 will automatically be recalled next time you use the same computer/browser to connect to that FB6000.

It is also possible to configure an external CSS to use with the FireBrick web control pages which allows a great deal of control over the overall layout and appearance. This can be usful for dealers or IT support companies to set up FireBricks in a style and branding of their choice.

3.4.2. Config pages and the object hierarchy

The structure of the config pages mirrors the object hierachy, and therefore they are themselves naturally hierachical. Your postition in the hierachy is illustrated in the 'breadcrumbs' trail at the top of the page, for example :-

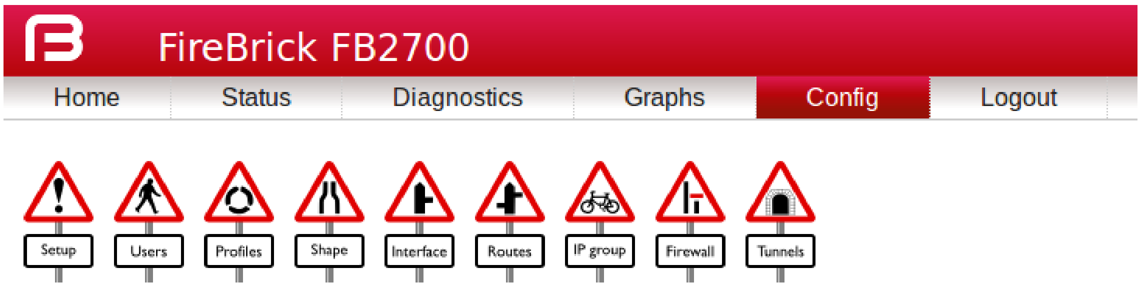
```
Firewall/mapping rules :: rule-set 1 of 3 (filters) :: rule 7 of 19 (ICMP)
```

This shows that the current page is showing a rule, which exists within a rule-set, which in turn is in the "Firewall/mapping rules" *category* (see below).

3.4.2.1. Configuration categories

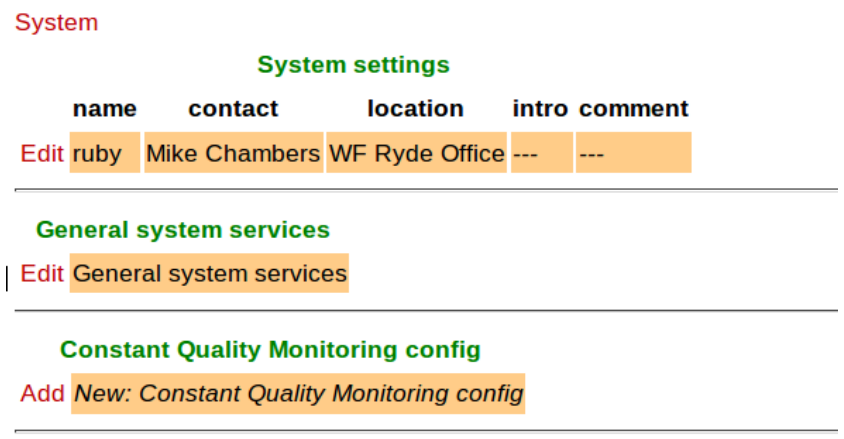
Configuration objects are grouped into a number of *categories*. At the top of the config pages is a set of icons, one for each category, as shown in Figure 3.3 :-

Figure 3.3. Icons for configuration categories



Within each category, there are one or more sections delimited by horizontal lines. Each of these sections has a heading, and corresponds to a particular type of *top-level* object, and relates to a major part of the configuration that comes under the selected category. See Figure 3.4 for an example showing part of the "Setup" category, which includes general system settings (the `system` object) and control of system services (network services provided by the FB6000, such as the web-interface web server, telnet server etc., controlled by the `services` object).

Figure 3.4. The "Setup" category



System

System settings

	name	contact	location	intro	comment
Edit	ruby	Mike Chambers	WF Ryde Office	---	---

General system services

| Edit | General system services |

Constant Quality Monitoring config

| Add | *New: Constant Quality Monitoring config* |

Each section is displayed as a tabulated list showing any existing objects of the associated type. Each row of the table corresponds with one object, and a subset (typically those of most interest at a glance) of the object's attributes are shown in the columns - the column heading shows the attribute name. If no objects of that type exist, there will be a single row with an "Add" link. Where the order of the objects matter, there will be an 'Add' link against each object - clicking an 'Add' link for a particular object will insert a new object *before* it. To add a new object after the last existing one, click on the 'Add' link on the bottom (or only) row of the table.

Tip

If there is no 'Add' link present, then this means there can only exist a limited number of objects of that type (possibly only one), and this many already exist. The existing object(s) may have originated from the factory reset configuration.

You can 'push-down' into the hierarchy by clicking the 'Edit' link in a table row. This takes you to a page to edit that specific object. The page also shows any child objects of the object being edited, using the same horizontal-line delimited section style used in the top-level categories. You can navigate back up the hierarchy using various methods - see Section 3.4.3.

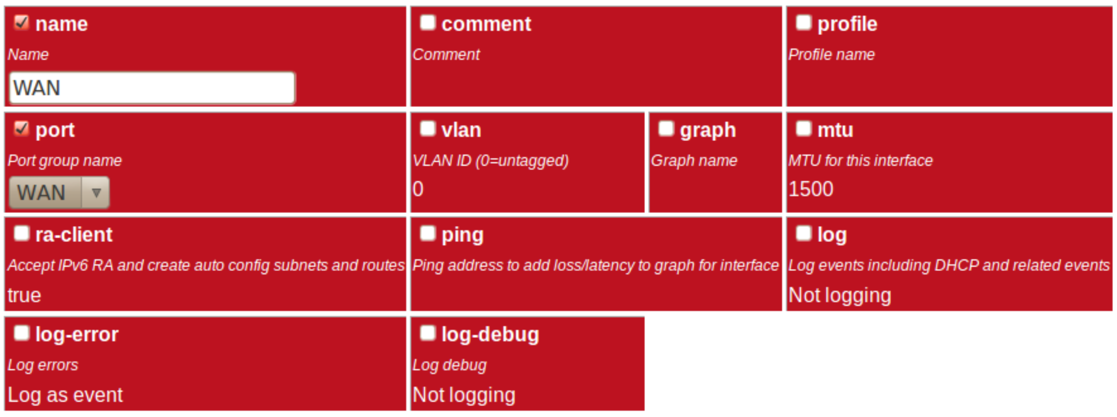
Caution

Clicking the "Add" link will create a new sub-object which will have blank/default settings. This can be useful to see what attributes an object can take, but if you do not want this blank object to be part of the configuration you later save you will need to click Erase. Simply going back "Up" or moving to another part of the config will leave this newly created empty object and that could have undesirable effects on the operation of your FireBrick if saved.

3.4.2.2. Object settings

The details of an object are displayed as a matrix of boxes (giving the appearance of a wall of bricks), one for each attribute associated with that object type. Figure 3.5 shows an example for an `interface` object (covered in Chapter 6) :-

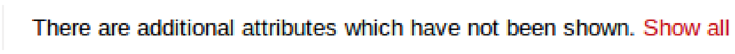
Figure 3.5. Editing an "Interface" object



✓ **name** *Name* WAN	**comment** *Comment*		**profile** *Profile name*
✓ **port** *Port group name* WAN ▾	**vlan** *VLAN ID (0=untagged)* 0	**graph** *Graph name*	**mtu** *MTU for this interface* 1500
ra-client *Accept IPv6 RA and create auto config subnets and routes* true	**ping** *Ping address to add loss/latency to graph for interface*		**log** *Log events including DHCP and related events* Not logging
log-error *Log errors* Log as event	**log-debug** *Log debug* Not logging		

By default, more advanced or less frequently used attributes are hidden - if this applies to the object being edited, you will see the text shown in Figure 3.6. The hidden attributes can be displayed by clicking on the link "Show all".

Figure 3.6. Show hidden attributes



There are additional attributes which have not been shown. Show all

Each brick in the wall contains the following :-

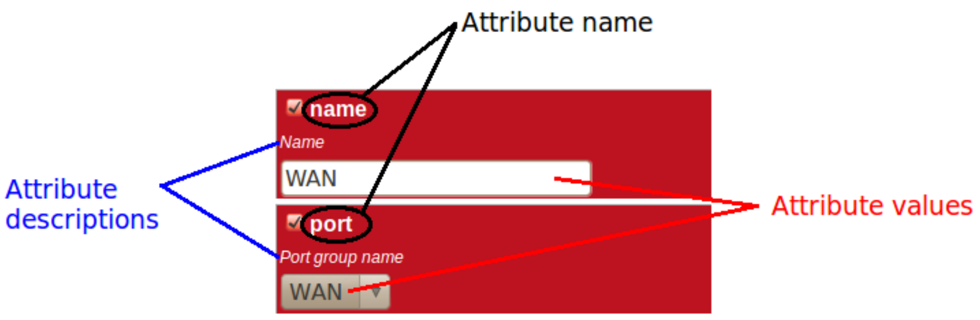
- a checkbox - if the checkbox is checked, an appropriate value entry widget is displayed, otherwise, a *default* value is shown and applied for that setting. If the attribute is not optional then no checkbox is show.
- the attribute name - this is a compact string that exactly matches the underlying XML attribute name
- a short description of the attribute

Tip

If there is no default shown for an attribute then its value, if needed, is zero, blank, null, empty string, false (internally it is zero bits!). In some cases the presence of an attribute will have meaning even if that attribute is an empty string or zero value. In some cases the default for an attribute will not be a fixed value but will depend on other factors, e.g. it may be "auto", or "set if using xyz...". The description of the default value should make this clear. Where an optional attribute is not ticked the attribute does not appear in the XML at all.

These can be seen in Figure 3.7 :-

Figure 3.7. Attribute definitions



If the attribute value is shown in a 'strike-through' font (with a horizontal line through it mid-way vertically), this illustrates that the attribute can't be set - this will happen where the attribute value would reference an instance of particular type of object, but there are not currently any instances of objects of that type defined.

Tip

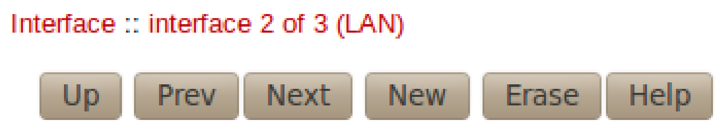
Since the attribute name is a compact, concise and un-ambiguous way of referring to an attribute, please quote attribute names when requesting technical support, and expect technical support staff to discuss your configuration primarily in terms of attribute (and object/element) names, rather than descriptive text, or physical location on your screen (both of which can vary between software releases).

3.4.3. Navigating around the User Interface

You navigate around the hierarchy using one or more of the following :-

- configuration category icons
- the breadcrumbs - each part of the breadcrumbs (delimited by the :: symbol) is a clickable link
- the *in-page* navigation buttons, shown in Figure 3.8 : "Up" - move one level up in the object hierachy, "Prev" - Previous object in a list, and "Next" - Next object in a list.

Figure 3.8. Navigation controls



Caution

The configuration pages are generated on-the-fly using JavaScript within your web browser environment (i.e. client-side scripting). As such, the browser is essentially unaware of changes to page content, and cannot track these changes - *this means the browser's navigation buttons (Back, Forward), will not correctly navigate through a series of configuration pages.*

Please take care not to use the browser's Back button whilst working through configuration pages - navigation between such pages must be done via the buttons provided on the page - "Prev", "Next" and "Up".

Navigating away from an object *using the supported navigation controls* doesn't cause any modifications to that object to be lost, even if the configuration has not yet been saved back to the FB6000. All changes are initially held in-memory (in the web browser itself), and are committed back to the FireBrick only when you press the Save button.

The navigation button area, shown in Figure 3.8, also includes three other buttons :-

- New : creates a new instance of the object type being edited - the new object is inserted after the current one ; this is equivalent to using the "Add" link one level up in the hierarchy
- Erase : deletes the object being edited - note that the object will not actually be erased until the configuration is saved
- Help : browses to the online reference material (as desribed in Section 3.2.1) for the object type being edited

Caution

If you *Add* a new object, but don't fill in any parameter values, the object will remain in existence should you navigate away. You should be careful that you don't inadvertently add incompletely setup objects this way, as they may affect operation of the FireBrick, possibly with a detrimental effect.

If you have added an object, perhaps for the purposes of looking at what attributes can be set on it, remember to delete the object before you navigate away -- the "Erase" button (see Figure 3.8) is used to delete the object you are viewing.

3.4.4. Backing up / restoring the configuration

To back up / save or restore the configuration, start by clicking on the "Config" main-menu item. This will show a page with a form to upload a configuration file (in XML) to the FB6000 - also on the page is a link "Download/save config" that will download the current configuration in XML format.

3.5. Configuration using XML

3.5.1. Introduction to XML

An XML file is a text file (i.e. contains human-readable characters only) with formally defined structure and content. An XML file starts with the line :-

```
<?xml version="1.0" encoding="UTF-8"?>
```

This defines the version of XML that the file complies with and the character encoding in use. The UTF-8 character coding is used everywhere by the FireBrick.

The XML file contains one or more *elements*, which may be nested into a hiearchy.

Note

In XML, the configuration objects are represented by *elements*, so the terms object and element are used interchangeably in this manual.

Each element consists of some optional content, bounded by two *tags* - a *start tag* AND an *end tag*.

A start tag consists of the following sequence of characters:-

- a < character
- the element name
- optionally, a number of *attributes*
- a > character

An end tag consists of the following sequence of characters:-

- a < character
- a / character
- the element name
- a > character

If an element needs no content, it can be represented with a more compact *self closing tag*. A self closing tag is the same as a start tag but ends with /> and then has no content or end tag.

Since the <, > and " characters have special meaning, there are special ('escape') character sequences starting with the ampersand character that are used to represent these characters. They are :-

Table 3.1. Special character sequences

Sequence	Character represented
`&lt;`	<
`&gt;`	>
`&quot;`	"
`&amp;`	&

Note that since the ampersand character has special meaning, it too has an escape character sequence.

Attributes are written in the form : `name="value"` or `name='value'`. Multiple attributes are separated by white-space (spaces and line breaks).

Generally, the content of an element can be other *child* elements or text. However, the FB6000 doesn't use text content in elements - all configuration data is specified via attributes. Therefore you will see that elements only contain one or more child elements, or no content at all. Whilst there is generally not any text between the tags, white space is normally used to make the layout clear.

3.5.2. The root element - <config>

At the top level, an XML file normally only has one element (the *root* element), which contains the entire element hierarchy. In the FB6000 the root element is `<config>`, and it contains 'top-level' configuration elements that cover major areas of the configuration, such as overall system settings, interface definitions, firewall *rule sets* etc.

In addition to this User Manual, there is reference material is available that documents the XML elements - refer to Section 3.2.1.

3.5.3. Viewing or editing XML

The XML representation of the configuration can be viewed and edited (in text form) via the web interface by clicking on "XML View" and "XML Edit" respectively under the main-menu "Config" item. Viewing the configuration is, as you might expect, 'read-only', and so is 'safe' in as much as you can't accidentally change the configuration.

3.5.4. Example XML configuration

An example of a simple, but complete XML configuration is shown below, with annotations pointing out the main elements

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://firebrick.ltd.uk/xml/fb2700/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://firebrick.ltd.uk/xml/fb2700/  ...
        timestamp="2011-10-14T12:24:07Z"
        patch="8882">
```

```
<system name="gateway"             ❶
        contact="Peter Smith"
        location="The Basement"
        log-panic="fb-support">
</system>

<user name="peter"                 ❷
      full-name="Peter Smith"
      password="FB105#4D42454D26F8BF5480F07DFA1E41AE47410154F6"
      timeout="PT3H20M"
      config="full"
      level="DEBUG"/>

<log name="default"/>           ❸
<log name="fb-support">
 <email to="crashlog@firebrick.ltd.uk"
        comment="Crash logs emailed to FireBrick Support"/>
</log>

<services>                      ❹
 <ntp timeserver="pool.ntp.org"/>
 <telnet log="default"/>
 <http />
 <dns domain="watchfront.co.uk"
      resolvers="81.187.42.42 81.187.96.96"/>
</services>

<port name="WAN"                   ❺
      ports="1"/>
<port name="LAN"
      ports="2"/>

<interface name="WAN"
           port="WAN">
 <subnet name="ADSL"
         ip="81.187.106.73/30"/>
</interface>

<interface name="LAN"               ❻
           port="LAN">
 <subnet name="LAN"
         ip="81.187.96.94/28"/>
 <dhcp name="LAN"
       ip="81.187.96.88-92"
       log="default"/>
</interface>

</config>
```

❶ sets some general system parameters (see Section 4.2)
❷ defines a single user with the highest level of access (DEBUG) (see Section 4.1)
❸ defines a log target (see Chapter 5)
❹ configures key system services (see Chapter 10)
❺ defines physical-port group (see Section 6.1)

❻ defines an interface, with one subnet and a DHCP allocation pool (see Chapter 6)

3.6. Downloading/Uploading the configuration

The XML file may be retrieved from the FireBrick, or uploaded to the FireBrick using HTTP transfers done via tools such as `curl`. Using these methods, configuration of the FB6000 can be integrated with existing administrative systems.

Note

Linebreaks are shown in the examples below for clarity only - they must not be entered on the command-line

3.6.1. Download

To download the configuration from the FB6000 you need to perform an HTTP `GET` of the following URL :-

`http://<FB6000 IP address or DNS name>/config/config`

An example of doing this using `curl`, run on a Linux box is shown below :-

```
curl http://<FB6000 IP address or DNS name>/config/config
   --user "username:password" --output "filename"
```

Replace *username* and *password* with appropriate credentials.

The XML configuration file will be stored in the file specified by *filename* - you can choose any file extension you wish (or none at all), but we suggest that you use `.xml` for consistency with the file extension used when saving a configuration via the User Interface (see Section 3.4.4).

3.6.2. Upload

To upload the configuration to the FB6000 you need to send the configuration XML file as if posted by a web form, using encoding MIME type `multi-part/form-data`.

An example of doing this using `curl`, run on a Linux box is shown below :-

```
curl http://<FB6000 IP address or DNS name>/config/config
   --user "username:password" --form config="@filename"
```

Chapter 4. System Administration

4.1. User Management

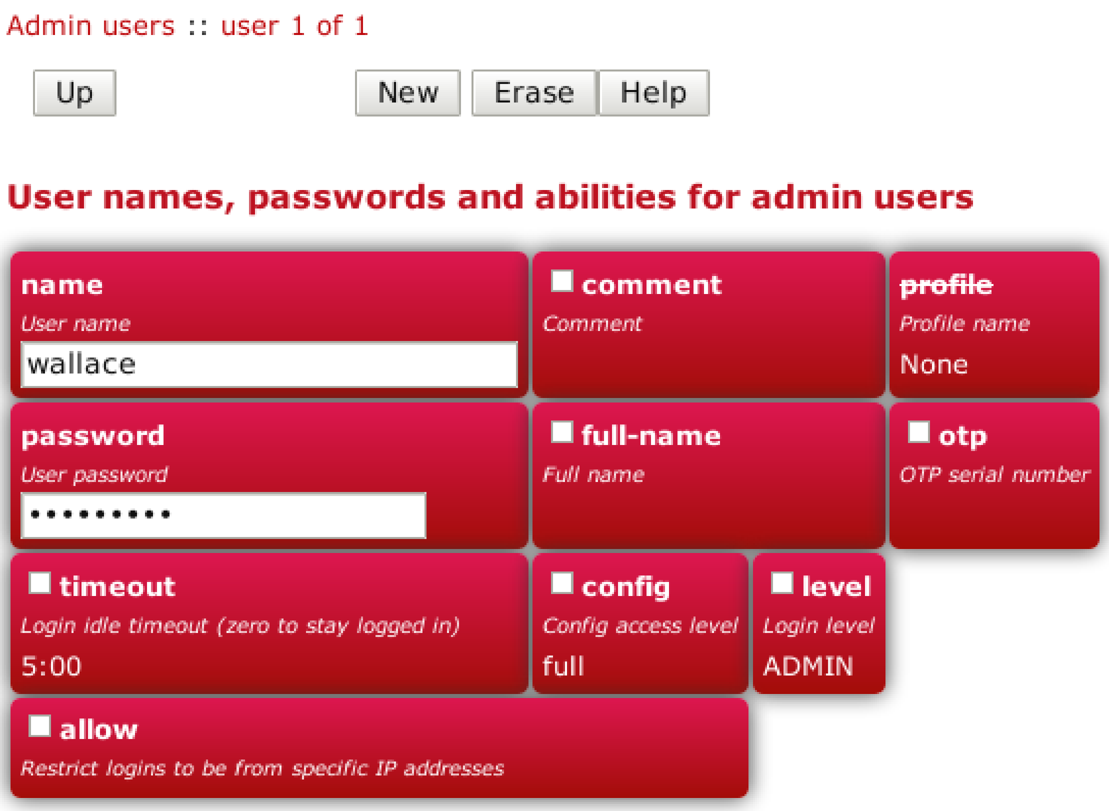
You will have created your first user as part of the initial setup of your FB6000, as detailed in either the QuickStart Guide or in Chapter 2 in this manual.

To create, edit or delete users, browse to the config pages by clicking the "Edit" item in the sub-menu under the "Config" main-menu item, then click on the "Users" category icon. Click on the "Edit" link adjacent to the user you wish to edit, or click on the "Add" link to add a user.

To delete a user, click the appropriate "Edit" link, then click the "Erase" button in the navigation controls - see Figure 3.8. As with any such object erase operation, the object will not actually be erased until the configuration is saved.

Once you have added a new user, or are editing an existing user, the object editing page will appear, as shown in Figure 4.1 :-

Figure 4.1. Setting up a new user



User names, passwords and abilities for admin users

name — User name: wallace

☐ comment — Comment

~~profile~~ — Profile name: None

password — User password: ••••••••••

☐ full-name — Full name

☐ otp — OTP serial number

☐ timeout — Login idle timeout (zero to stay logged in): 5:00

☐ config — Config access level: full

☐ level — Login level: ADMIN

☐ allow — Restrict logins to be from specific IP addresses

The minimum attributes that must be specified are `name`, which is the username that you type in when logging in, and `password` - passwords are mandatory on the FB6000.

You can optionally provide a *full name* for the specified username, and a general comment field value.

4.1.1. Login level

A user's *login level* is set with the `level` attribute, and determines what CLI commands the user can run. The default, if the `level` attribute is not specified, is `ADMIN` - you may wish to downgrade the level for users who are not classed as 'system administrators'.

Table 4.1. User login levels

Level	Description
`NOBODY`	No access to any menu items
`GUEST`	Guest user, access to some menu items
`USER`	Normal unprivileged user
`ADMIN`	System administrator
`DEBUG`	System debugging user

4.1.2. Configuration access level

The *configuration access level* determines whether a user has read-only or read-write access to the configuration, as shown in Table 4.2 below. This mechanism can also be used to deny all access to the configuration using the `none` level, but still allowing access to other menus and diagnostics.

This setting is distinct from, and not connected with, the *login level* described above. You can use the access level to define, for example, whether a USER login-level user can modify the configuration. Typically an ADMIN (or DEBUG) login-level user would always be granted full access, so for ADMIN or DEBUG level user's, the default of `full` is suitable.

Table 4.2. Configuration access levels

Level	Description
`none`	No access unless explicitly listed
`view`	View only access (no passwords)
`read`	Read only access (with passwords)
`full`	Full view and edit access - DEFAULT

4.1.3. Login idle timeout

To improve security, login sessions to either the web user interface, or to the command-line interface (via telnet, see Chapter 13), will time-out after a period of inactivity. This idle time-out defaults to 5 minutes, and can be changed by setting the `timeout` attribute value.

The time-out value is specified using the syntax for the XML *fb:duration* data type. The syntax is hours, minutes and seconds, or minutes and seconds or just seconds. E.g. `5:00`.

To set a user's time-out in the user interface, tick the checkbox next to `timeout`, and enter a value in the format described above.

Setting a timeout to 0 means *unlimited* and shoudl obviously be used with care.

4.1.4. Restricting user logins

4.1.4.1. Restrict by IP address

You can restrict logins by a given user to be allowed only from specific IP addresses, using the `allow` attribute. This restriction is per-user, and is distinct from, and applies in addition to, any restrictions specified on either the web or telnet (for command line interface access) services (see Section 10.3 and Section 10.4), or any firewall rules that affect web or telnet access to the FB6000 itself.

4.1.4.2. Logged in IP address

The FireBrick allows a general definition of *IP groups* which allow a name to be used in place of a range of IP addresses. This is a very general mechanism that can be used for single IP addresses or groups of ranges

IPs, e.g. *admin-machines* may be a list or range of the IP addresses from which you want to allow some access. The feature can also be useful even where only one IP is in the group just to give the IP a meaningful name in an access list.

These named IP groups can be used in the *allow list* for a user login, along with specific IP addresses or ranges if needed.

However, *IP groups* can also list one or more user names and implicitely include the *current IP address* from which those users are logged in to the web interface. This can be useful for firewall rules where you may have to log in to the FireBrick, even as a NOBODY level user, just to get your IP address in an access list to allow further access to a network from that IP.

4.1.4.3. Restrict by profile

By specifying a profile name using the `profile` attribute, you can allow logins by the user only when the profile is in the Active state (see Chapter 8). You can use this to, for example, restrict logins to be allowed only during certain times of the day, or you can effectively suspend a user account by specifying an always-Inactive profile.

4.2. General System settings

The `system` top-level object can specify attributes that control general, global system settings. The available attributes are described in the following sections, and can be configured in the User Interface by choosing the "Setup" category, then clicking the "Edit" link under the heading "System settings".

The software auto upgrade process is controlled by `system` objects attributes - these are described in Section 4.3.3.2.

4.2.1. System name (hostname)

The system name, also called the *hostname*, is used in various aspects of the FB6000's functions, and so we recommend you set the hostname to something appropriate for your network.

The hostname is set using the `name` attribute.

4.2.2. Administrative details

The attributes shown in Table 4.3 allow you to specify general administrative details about the unit :-

Table 4.3. General administrative details attributes

Attribute	Purpose
`comment`	General comment field
`contact`	Contact name
`intro`	Text that appears on the 'home' page - the home page is the first page you see after logging in to the FB6000. This text is also displayed immediately after you login to a command-line session.
`location`	Physical location description

4.2.3. System-level event logging control

The `log` and `log-...` attributes control logging of events related to the operation of the system itself. For details on event logging, please refer to Chapter 5, and for details on the logging control attributes on `system` object, please refer to Section 5.7.

4.2.4. Home page web links

The home page is the first page you see after logging in to the FB6000, or when you click the Home main-menu item. The home page displays the system name, and, if defined, the text specified by the `intro` attribute on the `system` object.

Additionally, you can define one or more web links to appear on the home page. These are defined using `link` objects, which are child objects of the `system` object.

To make a usable link, you must specify the following two attributes on the `link` object :-

- `text` : the text displayed as a hyperlink
- `url` : link destination URL

Additionally, you can name a link, specify a comment, and make the presence of the link on the home page conditional on a profile.

4.3. Software Upgrades

FB6000 users benefit from FireBrick's pro-active software development process, which delivers fast fixes of important bugs, and implementation of many customer enhancement requests and suggestions for improvement. As a matter of policy, FireBrick software upgrades are always free to download for all FireBrick customers.

To complement the responsive UK-based development process, the FB6000 is capable of downloading and installing new software *directly from Firebrick's servers*, providing the unit has Internet access.

This Internet-based upgrade process can be initiated manually (refer to Section 4.3.3.1), or the FB6000 can download and install new software automatically, without user intervention.

If the unit you want to upgrade does not have Internet access, then new software can be uploaded to the unit via a web browser instead - see Section 4.3.4.

Caution

Software upgrades are best done using the Internet-based upgrade process if possible - this ensures the changes introduced by *Breakpoint* releases are automatically accounted for (see Section 4.3.1.1)

Software upgrades will trigger an automatic reboot of your FB6000 - this will cause an outage in routing, and can cause connections that are using NAT to drop. However, the FB6000 reboots very quickly, and in many cases, users will be generally unaware of the event. You can also use a profile to restrict when software upgrades may occur - for example, you could ensure they are always done over night. For this reason, on the FB6000 factory reset config does not have automatic s/w upgrades enabled.

4.3.1. Software release types

There are three types of software release : factory, beta and alpha. For full details on the differences between these software releases, refer to the FB6000 software downloads website [http://www.firebrick.co.uk/software.php?PRODUCT=6000] - please follow the 'read the instructions' link that you will find just above the list of software versions.

Note

In order to be able to run alpha releases, your FB6000 must be enabled to run alpha software - this is done by changing the entry in the FireBrick capabilities database (hosted on FireBrick company servers) for your specific FB6000, as identified by the unit's Serial Number. Normally your FB6000 will be running factory or possibly beta software, with alpha software only used under advice and guidance of support personnel while investigating/fixing possible bugs or performance issues. You can see whether your FB6000 is able to run alpha releases by viewing the main Status page (click the Status main menu-item), and look for the row labelled "Allowed" - if the text shows "Alpha builds (for testing)" then your FB6000 can run alpha releases.

4.3.1.1. Breakpoint releases

Occasionally, a software release will introduce a change to the object model that means the way specific functionality is configured in XML also changes - for example an attribute may have been deprecated, and a replacement attribute should be used instead. A release where such an change has been made, and existing configurations will need modifying, are termed *Breakpoint* software releases.

Breakpoint releases are special as they are able to automatically update an existing configuration - used with the *previous* software release - so that it is compatible with the new release, and functionality is retained where-ever possible.

When using the Internet-based upgrade process, the FB6000 will always upgrade to the next available breakpoint version first, so that the configuration is updated appropriately. If your current software version is several breakpoint releases behind the latest version, the upgrade process will be repeated for each breakpoint release, and then to the latest version if that is later than the latest breakpoint release.

On the FB6000 software downloads website, breakpoint releases are labelled `[Breakpoint]` immediately under the version number.

Note

If you have saved copies of configurations for back-up purposes, always re-save a copy after upgrading to a breakpoint issue. If you use automated methods to configure your FB6000, check documentation to see whether those methods need updating.

4.3.2. Identifying current software version

The current software version is displayed on the main Status page, shown when you click the Status main menu-item itself (i.e. not a submenu item). The main software application version is shown next to the word "Software", e.g. :-

```
Software     FB2700 Hermia (V1.07.001 2011-11-15T10:22:48)
```

The software version is also displayed in the right hand side of the 'footer' area of each web page, and is shown immediately after you login to a command-line session.

4.3.3. Internet-based upgrade process

If automatic installs are allowed, the FB6000 will check for new software on boot up and approximately every 24 hours thereafter - your FB6000 should therefore pick up new software at most ~ 24 hours after it is released. You can choose to allow this process to install only new factory-releases, factory or beta releases, or any release, which then includes *alpha* releases (if your FB6000 is enabled for alpha software - see Section 4.3.1) - refer to Section 4.3.3.2 for details on how to configure auto upgrades.

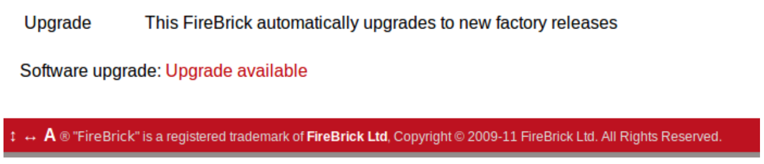
Caution

Alpha releases may be unstable, and so we do not generally recommend setting your FB6000 to automatically install alpha releases.

4.3.3.1. Manually initiating upgrades

Whenever you browse to the main Status page, the FB6000 checks whether there is newer software available, given the current software version in use, and whether alpha releases are allowed. If new software is available, you will be informed of this as shown in Figure 4.2 :-

Figure 4.2. Software upgrade available notification


Upgrade This FireBrick automatically upgrades to new factory releases

Software upgrade: Upgrade available

↕ ↔ A ® "FireBrick" is a registered trademark of **FireBrick Ltd**, Copyright © 2009-11 FireBrick Ltd. All Rights Reserved.

To see what new software is available, click on the "Upgrade available" link. This will take you to a page that will show *Release notes* that are applicable given your current software version, and the latest version available. On that page there is an "Upgrade" button which will begin the software upgrade process.

4.3.3.2. Controlling automatic software updates

There are two attributes on the `system` object (see Section 4.2) that affect the automatic software upgrade process :-

Table 4.4. Attributes controlling auto-upgrades

Attribute	Description
`sw-update`	Controls what types of software releases the auto-upgrade process will download/install. This attribute can also be used to disable the auto-upgrade process - use the value of `false` to achieve this. • `false` : Disables auto upgrades • `factory` : Only download/install factory releases - this is the default if the attribute is not specified • `beta` : Download/install factory or beta releases • `alpha` : Download/install factory, beta or alpha releases
`sw-update-profile`	Specifies the name of a profile to use to control when software upgrades are attempted (see Chapter 8 for details on profiles).

The current setting of `sw-update` (in descriptive form) can be seen on the main Status page, adjacent to the word "Upgrade", as shown in Figure 4.2 (in that example, `sw-update` is set to, or is defaulting to, `factory`).

4.3.4. Manual upgrade

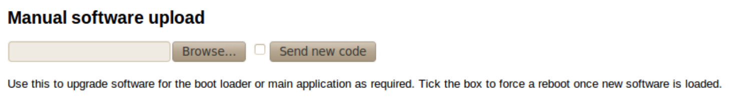
This method is entirely manual, in the sense that the brick itself does not download new software from the FireBrick servers, and responsibilty for loading breakpoint releases as required lies with the user.

In order to do this, you will first need to download the required software image file (which has the file extension `.img`) from the FB6000 software downloads website [http://www.firebrick.co.uk/software.php?PRODUCT=6000] onto your PC.

The next step is the same as you would perform when manually-initiating an Internet-based upgrade i.e. you should browse to the main Status page, where, if there is new software is available, you will be informed of this as shown in Figure 4.2.

This step is necessary since the manual upgrade feature currently shares the page used for Internet-based manual upgrades, which is reached by clicking "Upgrade available" link. After clicking this link, you will find the manual upgrade method at the bottom of the page, as shown in Figure 4.3 :-

Figure 4.3. Manual Software upload



Use this to upgrade software for the boot loader or main application as required. Tick the box to force a reboot once new software is loaded.

4.4. Boot Process

The FB6000 contains internal Flash memory storage that holds two types of software :-

- main application software (generally referred to as the *app*)
- a bootloader - runs immediately on power-up, initialises system, and then loads the app

It is possible for only one of these types of software, or neither of them, to be present in the Flash, but when shipped from the factory the unit will contain a bootloader and the latest factory-release application software. The FB6000 can store multiple app software images in the Flash, and this is used with an automatic fall-back mechanism - if a new software image proves unreliable, it is 'demoted', and the unit falls back to running older software. The `show flash contents` CLI command can be used to see what is stored in the Flash - see Appendix D.

4.4.1. LED indications

4.4.1.1. Power LED status indications

The green power LED has three defined states, as shown in Table 4.5 below :-

Table 4.5. Power LED status indications

Indication	Status
Off	No AC power applied to unit (or possibly hardware fault)
Flashing with approximately 1 second period	Bootloader running / waiting for network connection
On	Main application software running

After power-up, the normal power LED indication sequence is therefore to go through the ~1 second period flashing phase, and then - if at least one Ethernet port is connected to an active device - change to solid once the app is running.

From power-up, a FB6000 will normally boot and be operational in *under five seconds*.

4.4.1.2. Port LEDs

Whilst the bootloader is waiting for an active Ethernet connection, the green and yellow LEDs built into the physical port connectors flash in a continual left-to-right then right-to-left sequence. The port LEDs on the panel on the opposite side to the physical ports also flash, in a clock-wise sequence.

Note

The same port LED flashing sequences are observed if the app is running and none of the Ethernet ports are connected to an active link-partner. Note that the app continues to run, and the power LED will still be on solid.

When connected to an active link-partner, these flashing sequences will stop and the port LEDs will start indicating physical port status, with various status indications possible, controllable via the configuration (see Section 6.3).

Chapter 5. Event Logging

5.1. Overview

Many *events* in the operation of the FireBrick create a log entry. These are a one-line string of text saying what happened. This could be normal events such as someone logging in to the web interface, or unusual events such as a wrong password used, or DHCP not being able to find any free addresses to allocate.

5.1.1. Log targets

A *log target* is a named destination (initially internal to the FB6000) for log entries - you can have multiple log targets set up which you can use to separate out log event messages according to some criteria - for example, you could log all firewalling related log events to a log target specifically for that purpose. This makes it easier to locate events you are looking for, and helps you keep each log target uncluttered with un-related log events - this is particularly important when when you are logging a lot of things very quickly.

A log target is defined using a `log` top-level object - when using the web User Interface, these objects are in the "Setup" category, under the heading "Log target controls".

Every log entry is put in a buffer in RAM, which only holds a certain number of log entries (typically around 1MB of text) - once the buffer is full, the oldest entries are lost as new ones arrive. Since the buffer is stored in volatile memory (RAM), buffer contents are lost on reboot or power failure.

This buffer can be viewed via the web interface or command line which can show the history in the buffer and then *follow the log* in real time (even when viewing via a web browser, with some exceptions - see Section 5.6.1).

In some cases it is essential to ensure logged events can be viewed even after a power failure. You can flag a log target to log to the non-volatile Flash memory within the FB6000, where it will remain stored even after a power failure. You should read Section 5.5 before deciding to log events to Flash memory.

Each log target has various attributes and child objects defining what happens to log entries to this target. However, in the simplest case, where you do not require non-volatile storage, or external logging (see Section 5.3), the log object will only need a `name` attribute, and will have no child objects. In XML this will look something like :-

```
<log name="my_log"/>
```

5.1.1.1. Logging to Flash memory

The internal Flash memory logging system is separate from the external logging. It applies if the log target object has `flash="true"`. It causes each log entry to be written to the internal non-volatile Flash log as it is created.

The flash log is intended for urgent permanent system information only, and is visible using the `show flash log` CLI command (see Appendix D for details on using this command). Chapter 13 covers the CLI in general.

Caution

Flash logging slows down the system considerably - only enable Flash logging where absolutely necessary.

The flash log does have a limit on how much it can hold, but it is many thousands of entries so this is rarely an issue. Oldest entries are automatically discarded when there is no space.

5.1.1.2. Logging to the Console

The *console* is the command line environment described in Chapter 13. You can cause log entries to be displayed as soon as possible on the console (assuming an active console session) by setting `console="true"` on the log target.

You can stop the console logging with `troff` command or restart it with `tron` command.

The FB6000 also has a serial console to which *console* log entries are sent if logged in.

5.2. Enabling logging

Event logging is enabled by setting one of the attributes shown in Table 5.1 on the appropriate object(s) in the configuration, which depends on what event(s) you are interested in. The attribute value specifies the name of the log target to send the event message to. The events that cause a log entry will naturally depend on the object on which you enable logging. Some objects have additional attributes such as `log-error` for unusual events, and `log-debug` for extra detail.

Table 5.1. Logging attributes

Attribute	Event types
`log`	This is normal events. Note that if `log-error` is not set then this includes errors.
`log-error`	This is when things happen that should not. It could be something as simple as bad login on telnet. Note that if `log-error` is not set but `log` is set then errors are logged to the *log* target by default.
`log-debug`	This is extra detail and is normally only used when diagnosing a problem. Debug logging can be a lot of information, for example, in some cases whole packets are logged (e.g. PPP). It is generally best only to use debug logging when needed.

5.3. Logging to external destinations

Entries in the buffer can also be sent on to external destinations, such as via email or *syslog*. Support for triggering SNMP traps may be provided in a future software release.

You can set these differently for each log target. There is inevitably a slight lag between the event happening and the log message being sent on, and in some cases, such as email, you can deliberately delay the sending of logs to avoid getting an excessive number of emails.

If an external logging system cannot keep up with the rate logs are generated then log entries can be lost. The fastest type of external logging is using *syslog* which should manage to keep up in pretty much all cases.

5.3.1. Syslog

The FB6000 supports sending of log entries across a network to a *syslog server*. Syslog is described in RFC5424 [http://tools.ietf.org/html/rfc5424], and the FB6000 includes microsecond resolution time stamps, the hostname (from system settings) and a module name in entries sent via syslog. Syslog logging is very quick as there is no reply, and syslog servers can be easily setup on most operating systems, particularly Unix-like systems such as Linux.

Note

Older syslog servers will typically show time and hostname twice, and will need upgrading.

The module name refers to which part of the system caused the log entry, and is also shown in all other types of logging such as web and console.

To enable log messages to be sent to a syslog server, you need to create a `syslog` object that is a child of the log target (`log`) object. You must then specify the DNS name or IP address of your syslog server by setting the `server` attribute on the `syslog` object. You can also set the *facility* and/or *severity* values using these attributes :-

- `facility` : the 'facility' to be used in the syslog messages - when syslog entries are generated by subsystems or processes in a general-purpose operating system, the facility typically identifies the message source ; where the commonly used facility identifiers are not suitable, the "local0" thru "local7" identifiers can be used. If the `facility` attribute is not set, it defaults to `LOCAL0`
- `severity` : the severity value to be used in the syslog messages - if not set, the severity defaults to `NOTICE`

The FB6000 normally uses the 'standard' syslog port number of 514, but if necessary, you can change this by setting the `port` attribute value.

5.3.2. Email

You can cause logs to be sent by e-mail by creating an `email` object that is a child of the log target (`log`) object.

An important aspect of emailed logs is that they have a *delay* and a *hold-off*. The delay means that the email is not sent immediately because often a cluster of events happen over a short period and it is sensible to wait for several log lines for an event before e-mailing.

The hold-off period is the time that the FB6000 waits after sending an e-mail, before sending another. Having a hold-off period means you don't get an excessive number of e-mails ; since the logging system is initially storing event messages in RAM, the e-mail that is sent after the hold-off period will contain any messages that were generated during the hold-off period.

The following aspects of the e-mail process can be configured :-

- subject : you can either specify the subject, by setting the `subject` attribute value, or you can allow the FB6000 to create a subject based on the first line of the log message
- e-mail addresses : as to be expected, you must specify a target e-mail address, using the `to` attribute. You can optionally specify a From: address, by setting the `from` atttribute, or you can allow the FB6000 to create an address based on the unit's serial number
- outgoing mail server : the FB6000 normally sends e-mail directly to the Mail eXchanger (MX) host for the domain, but you can optionally specify an outgoing mail server ('smart host') to use instead, by setting the `server` attribute
- SMTP port number : the FB6000 defaults to using TCP port 25 to perform the SMTP mail transfer, but if necessary you can set the `port` attribute to specify which port number to use
- retry delay : if an attempt to send the e-mail fails, the FB6000 will wait before re-trying ; the default wait period is 10 minutes, but you can change this by setting the `retry` attribute

An example of a simple log target with e-mailing is available in a factory reset configuration - the associated XML is shown below, from which you can see that in many cases, you only need to specify the `to` attribute (the `comment` attribute is an optional, general comment field) :-

```
<log name="fb-support"
     comment="Log target for sending logs to FireBrick support team">
 <email to="crashlog@firebrick.ltd.uk"
        comment="Crash logs emailed to FireBrick Support team"/>
```

```
</log>
```

A profile can be used to stop emails at certain times, and when the email logging is back on an active profile it tries to catch up any entries still in the RAM buffer if possible.

5.3.2.1. E-mail process logging

Since the process of e-mailing can itself encounter problems, it is possible to request that the process itself be logged via the usual log target mechanism. This is done by specifying one or more of the `log`, `log-debug` and `log-error` attributes.

Note

We recommend that you avoid setting these attributes such that specify the log-target containing the `email` object, otherwise you are likely to continually receive e-mails as each previous e-mail process log will trigger another e-mail - the hold-off will limit the rate of these mails though.

5.4. Factory reset configuration log targets

A factory reset configuration has a log target named `default`, which only logs to RAM. Provided this log target has not been deleted, you can therefore simply set `log="default"` on any appropriate object to immediately enable logging to this 'default' log target, which can then be viewed from the web User Interface or via the CLI.

A factory reset configuration also has a log target named `fb-support` which is referenced by the `log-panic` attribute of the `system` object (see Section 5.7). This allows the FireBrick to automatically email the support team if there is a panic (crash) - you can, of course, change or delete this if you prefer.

Caution

Please only set things to log to `fb-support` if requested by support staff.

5.5. Performance

The FireBrick can log a lot of information, and adding logs can causes things to slow down a little. The controls in the config allow you to say what you log in some detail. However, logging to flash will always slow things down a lot and should only be used where absolutely necessary.

5.6. Viewing logs

5.6.1. Viewing logs in the User Interface

To view a log in the web User Interface, select the "Log" item in the "Status" menu. Then select which log target to view by clicking the appropriate link. You can also view a 'pseudo' log target "All" which shows log event messages sent to any log target.

The web page then continues showing log events on the web page in *real time i.e. as they happen.*

Note

This is an "open ended" web page which has been known to upset some browsers, but this is rare. However it does not usually work with any sort of web proxy which expects the page to actually finish.

All log targets can be viewed via the web User Interface, regardless of whether they specify any external logging (or logging to Flash memory).

5.6.2. Viewing logs in the CLI environment

The command line allows logs to be viewed, and you can select which log target, or all targets. The logging continues on screen until you press a key such as RETURN.

In addition, anything set to log to console shows anyway (see Section 5.1.1.2), unless disabled with the `troff` command.

5.7. System-event logging

Some aspects of the operation of the overall system have associated events and messages that can be logged. Logging of such events is enabled via the `system` object attributes shown in Table 5.2 below :-

Table 5.2. System-Event Logging attributes

`system` **object attribute**	**Event types**
`log`	General system events.
`log-debug`	System debug messages.
`log-error`	System error messages.
`log-eth`	General Ethernet hardware messages.
`log-eth-debug`	Ethernet hardware debug messages.
`log-eth-error`	Ethernet hardware error messages.
`log-panic`	System Panic events.
`log-stats`	"One second stats" messages

Specifying system event logging attributes is usually only necessary when diagnosing problems with the FB6000, and will typically be done under guidance from support staff. For example, `log-stats` causes a log message to be generated *every second* containing some key system statistics and state information, which are useful for debugging.

Note that there are some system events, such as startup and shutdown, which are always logged to all log targets, and to the console and flash by default, regardless of these logging attributes.

5.8. Using Profiles

The log target itself can have a profile which stops logging happening when the profile is disabled. Also, each of the external logging entries can have a profile. Some types of logging will catch up when their profile comes back on (e.g. email) but most are immediate (such as syslog and SMS) and will drop any entries when disabled by an Inactive profile.

Chapter 6. Interfaces and Subnets

This chapter covers the setup of *Ethernet* interfaces and the definition of subnets that are present on those interfaces.

6.1. Relationship between Interfaces and Physical Ports

The FB6000 features two Gigabit Ethernet (1Gb/s) ports. These ports only work at gigabit speeds.

Each port features a green and amber LED, the functions of which can be chosen from a range of options indicating link speed and/or traffic activity.

The exact function of the ports is flexible, and controlled by the configuration of the FB6000.

6.1.1. Port groups

As the FB6000 only has two phisical ports, the port group configuration on the FB6000 has no options, only two groups are possible, each with the one physical interface. Port group configuration is provided only for consistency and some degree of configuration file portability with the FB2500 and FB2700 products.

6.1.2. Interfaces

In the FB6000, an *interface* is a logical equivalent of a physical Ethernet interface adapter. Each interface normally exists in a distinct *broadcast domain*, and is associated with at most one port group.

Each port can operate simply as an *interface* with no VLANs, or can have one or more tagged VLANs which are treated as separate logical *interfaces*. Using VLAN tags and a VLAN capable switch you can effectively increase the number of physical ports.

If you are unfamiliar with VLANs or the concept of broadcast domains, Appendix C contains a brief overview.

By combining the FB6000 with a VLAN capable switch, using only a single physical connection between the switch and the FB6000, you can effectively expand the number of distinct physical interfaces, with the upper limit on number being determined by switch capabilities, or by inherent IEEE 802.1Q VLAN or FB6000 MAC address block size. An example of such a configuration is a multi-tenant serviced-office environment, where the FB6000 acts as an Internet access router for a number of tenants, firewalling between tenant networks, and maybe providing access to shared resources such as printers.

6.2. Defining an interface

To create or edit interfaces, select the Interface category in the top-level icons - under the section headed "Ethernet interface (port-group/vlan) and subnets", you will see the list of existing `interface` top-level objects (if any), and an "Add" link.

The primary attributes that define an interface are the name of the physical port group it uses, an optional VLAN ID, and an optional name. If the VLAN ID is not specified, it defaults to "0" which means only *untagged* packets will be received by the interface.

To create a new interface, click on the Add link to take you to a new interface defintion. Select one of the defined port groups. If the interface is to exist in a VLAN, tick the `vlan` checkbox and enter the VLAN ID in the text field.

Editing an existing interface works similarly - click the Edit link next to the interface you want to modify.

An `interface` object can have the following child objects :-

- One or more subnet definition objects
- Zero or more DHCP server settings objects
- Zero or more Virtual Router Redundancy Protocol (VRRP) settings objects (refer to Chapter 12)

6.2.1. Defining subnets

Each interface can have one *or more* subnets definitions associated with it. The ability to specify multiple subnets on an interface can be used where it is necessary to communicate with devices on two different subnets and it is acceptable that the subnets exist in the same broadcast domain. For example, it may not be possible to reassign machine addresses to form a single subnet, but the machines do not require firewalling from each other.

Note

As discussed in Section 6.1, an interface is associated with a broadcast domain ; therefore multiple subnets existing in a single broadcast domain are not 'isolated' (at layer 2) from each other. Effective firewalling (at layer 3) cannot be established between such subnets ; to achieve that, subnets need to exist in different broadcast domains, and thus be on different interfaces. An example of this is seen in the factory default configuration, which has two interfaces, "WAN" and "LAN", allowing firewalling of the LAN from the Internet.

You may also have both IPv4 and IPv6 subnets on an interface where you are also using IPv6 networking.

The primary attributes that define a subnet are the IP address range of the subnet, the IP address of the FB6000 itself on that subnet, and an optional name.

The IP address and address-range are expressed together using *CIDR notation* - if you are not familiar with this notation, please refer to Appendix A for an overview.

To create or edit subnets, select the Interface category in the top-level icons, then click Edit next to the appropriate interface - under the section headed "IP subnet on the interface", you will see the list of existing `subnet` child objects (if any), and an "Add" link.

Note

In a factory reset configuration, there are two temporary subnets defined on the "LAN" interface : `2001:DB8::1/64` and `10.0.0.1/24`. These subnet definitions provide a default IP address that the FB6000 can initially be accessed on, regardless of whether the FB6000 has been able to obtain an address from an existing DHCP server on the network. Once you have added new subnets to suit your requirements, and tested that they work as expected, these temporary definitions should be removed.

To create a new subnet, click on the Add link to take you to a new `subnet` object defintion. Tick the `ip` checkbox, and enter the appropriate CIDR notation.

Editing an existing subnet works similarly - click the Edit link next to the subnet you want to modify.

6.2.1.1. Using DHCP to configure a subnet

You can create a subnet that is configured via DHCP by clearing the `ip` checkbox - the absence of an IP address/prefix specification causes the FB6000 to attempt to obtain an address from a DHCP server (which must be in the same broadcast domain). It may help to use the Comment field to note that the subnet is configured via DHCP.

In its simplest form, a DHCP configured subnet is created by the following XML :- `<subnet />`

Tip

It is possible to specify multiple DHCP client subnets like this, and the FB6000 will reserve a separate MAC address for each. This allows the FB6000 to aquire multiple independant IP addresses by DHCP on the same interface if required.

6.2.2. Setting up DHCP server parameters

The FB6000 can act as a DHCP server to dynamically allocate IP addresses to clients. Optionally, the allocation can be accompanied by information such as a list of DNS resolvers that the client should use.

Since the DHCP behaviour needs to be defined for each interface (specifically, each broadcast domain), the behaviour is controlled by one or more `dhcp` objects, which are children of an `interface` object.

Address allocations are made from a *pool* of addresses - the pool is either explicitly defined using the `ip` attribute, or if `ip` is not specified, it consists of all addresses on the interface i.e. from all subnets, but excluding network or broadcast addresses, or any addresses that the FB6000 has seen ARP responses for (i.e. addresses already in use, perhaps through a static address configuration on a machine).

The XML below shows an example of an explicitly-specified DHCP pool :-

```
<interface ...>
...
  <dhcp name="LAN"
        ip="172.30.16.50-80"
        log="default"/>
...
</interface>
```

Tip

When specifying an explicit range of IP addresses, if you start at the *network* then the FB6000 will allocate that address. Not all devices cope with this so it is recommended that an explicit range is used, e.g. `192.168.1.100-199`. You do not, however, have to be careful of either the FireBrick's own addresses or subnet broadcast addresses as they are automatically excluded. When using the default (0.0.0.0/0) range network addresses are also omitted, as are any other addresses not within a subnet on the same interface.

Every allocation made by the DHCP server built-in to the FB6000 is stored in non-volatile memory, and as such will survive power-cycling and/or rebooting. The allocations can be seen using the "DHCP" item in the "Status" menu, or using the `show dhcp` CLI command.

If a client does not request renewal of the lease before it expires, the allocation entry will show "expired". Expired entries remain stored, and are used to lease the same IP address again if the same client (as identified by its MAC address) requests an IP address. However, if a new MAC address requests an allocation, and there are no available IPs (excluding expired allocations) in the allocation pool, then the oldest expired allocation IP address is re-used for the new client.

6.2.2.1. Fixed/Static DHCP allocations

'Fixed' (or 'static') allocations can be achieved by creating a separate `dhcp` object for each such allocation, and specifying the client MAC address via the `mac` attribute on the `dhcp` object.

The XML below shows an example of a fixed allocation - note the MAC address is written without any colons, and is therefore a string of twelve hexadecimal digits (48-bits). This allocation also supplies DNS resolver information to the client.

```
<interface ...>
...
  <dhcp name="laptop"
        ip="81.187.96.81"
        mac="0090F59E4F12"
        dns="81.187.42.42 81.187.96.96"
        log="default"/>
...
</interface>
```

Tip

If you are setting up a static allocation, but your client has already obtained an address (from your FB6000) from a pool, you will need to clear the allocation and then force the client to issue another DHCP request (e.g. unplug ethernet cable, do a software 'repair connection' procedure or similar etc.). See the `show dhcp` and `clear dhcp` CLI commands in the Appendix D for details on how to clear the allocation. Chapter 13 covers the CLI in general.

You can also *lock* an existing dynamic allocation to prevent it being re-used for a different MAC address even if it has expired.

6.2.2.1.1. Special DHCP attributes

For each pool you can list specific DHCP attributes, specified as a string, IPv4 address, or number, or even as raw data in hexadecimal. You can force sending of an attribute even if not requested.

For vendor specific attributes (ID 43) you can either specify in hex as ID 43, or you can specify the code to use and set the vendor flag, this adds an attribute type 43 with the code and length for the attribute which can be string, IPv4 address, number, or hexadecimal.

6.2.2.2. Partial-MAC-address based allocations

In addition to specifying a full 48-bit (12 hexadecimal character) MAC address in a `dhcp` object, it is also possible to specify part of a MAC address, specifically some number of *leading* bytes. The `dhcp` object will then apply for any client whose MAC address has the same leading bytes.

For example, as discussed in Appendix B, the first three octets (bytes) of a MAC address identify the organization (often the end product manufacturer) that can allocate that MAC address to an Ethernet device. By specifying only these first three bytes (six hexadecimal characters, no colon delimiters), in the `mac` attribute, you could ensure that all devices from the associated manufacturer are allocated addresses from a particular address pool. This is helpful if you have some common firewalling requirements for such a group of devices - for example, if all your VoIP phones are from one manufacturer - as you can have appropriate firewall rule(s) that apply to addresses in that pool.

6.3. Physical port settings

The detailed operation of each physical port can be controlled by creating `ethernet` top-level objects, one for each port that you wish to define different behaviour for vs. default behaviour.

To create a new `ethernet` object, or edit an existing object, select the Interface category from the top-level icons. Under the section headed "Ethernet port settings", you will see the list of existing `ethernet` objects (if any), and an "Add" link.

In a factory reset configuration, there are no `ethernet` objects, and all ports assume the following defaults :-

- Link auto-negotiation is enabled - both speed and duplex mode are determined via auto-negotiation, which should configure the link for highest performance possible for the given link-partner (which will need to be capable of, and participating in, auto-negotiation for this to happen)

- Auto-crossover mode is enabled - the port will swap Receive and Transmit pairs if required to adapt to cable / link-partner configuration
- The green port LED is configured to show combined Link Status and Activity indication - the LED will be off if no link is established with a link-partner. When a link is established (at any speed), the LED will be on steady when there is no activity, and will blink when there is activity.
- The yellow port LED is configured to show Transmit activity.

When you first create an `ethernet` object you will see that none of the attribute checkboxes are ticked, and the defaults described above apply. Ensure that you set the `port` attribute value correctly to modify the port you intended to.

The FB6000 configuration contains a number of port settings which are not possible and will not save, e.g. 10M and 100M modes. These are included for compatibility with FB2500 and FB2700 products. The FB6000 only operates at gigabit port speeds.

6.3.1. Setting duplex mode

If auto-negotiation is enabled, the FB6000 port will normally advertise that it is capable of either half- or full-duplex operation modes - if you have reason to restrict the operation to either of these modes, you can set the `duplex` attribute to either `half` or `full`. This will cause the port to only advertise the specified mode - if the (auto-negotiate capable) link-partner does not support that mode, the link will fail to establish.

If auto-negotiation is disabled, the `duplex` attribute simply sets the port's duplex mode.

Note

If you do not set the `autoneg` attribute (checkbox is unticked), and you set *both* port speed and duplex mode to values other than `auto`, auto-negotiation will be disabled ; this behaviour is to reduce the potential for duplex mis-match problems that can occur when connecting the FB6000 to some vendors' (notably Cisco) equipment that has auto-negotation disabled by default.

6.3.2. Defining port LED functions

Each port has options to control the way the yellow and green LEDs are displayed based on the state of the port. The default is yellow for Tx and green for link/activity.

Chapter 7. Routing

7.1. Routing logic

The routing logic in the FB6000 operates primarily using a conventional routing system of *most specific prefix*, which is commonly found in many IP stacks in general purpose computers and routers.

Conventional routing determines where to send a packet based *only* on the packet's *destination* IP address, and is applied on a 'per packet' basis - i.e. each packet that arrives is processed independently from previous packets.

Note that with this routing system, it does not matter where the packet came *from*, either in terms of source IP address or which interface/tunnel etc. the packet arrived on.

A *route* consists of :-

- a 'target' specifying where to send the packet to - this may be a specialised action, such as silently dropping the packet (a 'black-hole')
- an IP address range that this routing information applies to - the *routing destination*

A *routing table* consists of one or more routes. Unlike typical IP stacks, the FB6000 supports multiple independent routing tables.

Routing destinations are expressed using CIDR notation - if you are not familiar with this notation, please refer to Appendix A for an overview. Note that ip-groups cannot be used when defining subnets or routes. IP-groups allow arbitrary ranges and not just prefixes, but routes can only use prefixes.

There are two cases that deserve special attention :-

- A routing destination may be a single IP address, in which case it is a "/32" in CIDR notation (for IPv4). The /32 part (for IPv4) or /128 (for IPv6) is not shown when displaying such prefixes.
- A routing destination may encompass the entire IPv4 (or IPv6) address space, written as `0.0.0.0/0` (for IPv4) or `::/0` (for IPv6) in CIDR notation. Since the prefix is zero-length, all destination IP addresses will match this route - however, it is always the shortest-prefix route present, and so will only match if there are no more specific routes. Such routes therefore acts as a *default* route.

The decision of where to send the packet is based on matching the packet's destination IP address to one or more routing table entries. If more than one entry matches, then the longest (most specific) prefix entry is used. The longest prefix is assumed to be associated with the optimal route to the destination IP address, since it is the 'most specific', i.e. it covers a smaller IP address range than any shorter matching prefix.

For example, if you have two routes, one for 10.0.1.32/27 , and another for 10.0.0.0/8 (which encompasses 10.0.1.32/27), then a destination IP address of 10.0.1.35 will match the longest-prefix (smallest address range) "/27" route.

The order in which routes are created does not normally matter as you do not usually have two routes that have the same prefix. However, there is an attribute of every route called the `localpref` which decides between identical routes - the *higher* `localpref` being the one which applies. If you have identical routes with the same `localpref` then one will apply (you cannot rely on which one) but it can, in some cases, mean you are bonding multiple links.

Tip

You can show the route(s) that apply for a specific destination IP address or address range using the CLI command `show route`. You can also see a list of all routes in a routing table using the CLI command `show routes`. There is also a routing display on the Diagnostics control web pages.

7.2. Routing targets

A route can specify various targets for the packet :-

Table 7.1. Example route targets

Target	Notes
an Ethernet interface (locally-atached subnet)	requires ARP or ND to find the device on the LAN to which the traffic is to be sent.
a specific IP address (a "gateway")	the packet is forwarded to another router (gateway) ; routing is then determined based on the gateway's IP address instead
tunnel interface such as L2TP, PPPoE or FB105 tunnels.	such routes are created as part of the config for the interface and relate to the specific tunnel.
special targets	e.g. the FB6000 itself, or to a *black hole* (causes all traffic to be dropped)

These are covered in more detail in the following sections.

7.2.1. Subnet routes

Whenever you define a subnet or one is created dynamically (e.g. by DHCP), an associated route is automatically created for the associated prefix. Packets being routed to a subnet are sent to the Ethernet `interface` that the subnet is associated with. Traffic routed to the subnet will use ARP or ND to find the final MAC address to send the packet to.

In addition, a subnet definition creates a very specific single IP (a "/32" for IPv4, or a "/128" for IPv6) route for the IP address of the FB6000 itself on that subnet. This is a separate *loop-back* route which effectively internally routes traffic back into the FB6000 itself - i.e. it never appears externally.

A subnet can also have a *gateway* specified, either in the config or by DHCP or RA. This gateway is just like creating a route to 0.0.0.0/0 or ::/0 as a specific route configuration. It is mainly associated with the subnet for convenience. If defined by DHCP or RA then, like the rest of the routes created by DHCP or RA, it is removed when the DHCP or RA times out.

Example: `<subnet ip="192.168.0.1/24"/>` creates a route for destination `192.168.0.0/24` to the `interface` associated with that subnet. A loop-back route to `192.168.0.1` (the FB6000's own IP address on that subnet) is also created.

7.2.2. Routing to an IP address (gateway route)

Routes can be defined to forward traffic to another IP address, which will typically be another router (often also called a *gateway*) For such a routing target, the gateway's IP address is then used to determine how to route the traffic, and another routing decision is made. This subsequent routing decision usually identifies an `interface` or other data link to send the packet via - in more unusual cases, the subsequent routing decision identifies another gateway, so it is possible for the process to be 'recursive' until a 'real' destination is found.

Example: `<route ip="0.0.0.0/0" gateway="192.168.0.100"/>` creates a default IPv4 route that forwards traffic to `192.168.0.100`. The routing for `192.168.0.100` then has to be looked up to find the final target, e.g. it may be to an Ethernet interface, in which case an ARP is done for `192.168.0.100` to find the MAC to send the traffic.

There is logic to ensure that the *next-hop* is valid - the gateway specified must be routable somewhere and if that is via an Ethernet interface then the endpoint must be answering ARP or ND packets. If not, then the route using the gateway is *supressed* and other less specific routes may apply.

7.2.3. Special targets

It is possible to define two special targets :-

- 'black-hole' : packets routed to a black-hole are silently dropped. 'Silent' refers to the lack of any ICMP response back to the sender.
- 'nowhere' (also called *Dead End*) : packets routed to 'nowhere' are also dropped but the FB6000 generates ICMP error responses back to the sender.

The `blackhole` and `nowhere` top-level objects are used to specify prefixes which are routed to these special targets. In the User Interface, these objects can be found under the Routes category icon.

7.3. Dynamic route creation / deletion

For data links that have an Up/Down state, such as L2TP or FB105 tunnels, or PPP links, the ability to actually send traffic to the route target will depend on the state of the link. For such links, you can specify route(s) to automatically create each time the link comes up - when the link goes down these routes are removed automatically.

This can be useful where a link such as PPPoE is defined with a given `localpref` value, and a separate route is defined with a *lower* `localpref` value (i.e. less preferred), and therefore acts as a fallback route if the PPPoE link drops.

7.4. Routing tables

The conventional routing logic described above operates using one of possibly many routing tables that the FB6000 can support simultaneously. Routing tables are numbered, with the default being routing table 0 (zero).

The various ways to add routes allow the routing table to be specified, and so allow completely independent routing for different routing tables. The default table (table zero) is used when optional routing-table specification attributes or CLI command parameters are omitted.

Each `interface` is logically in a routing table and traffic arriving on it is processed based on the routes in that routing table. Tunnels like FB105 and L2TP allow the wrapped tunnel packets to work on one routing table and the tunnel payload packets to be on another.

Routing tables can be very useful when working with tunnels of any sort - placing the *wrappers* in one routing table, allowing DHCP clients and so on, without taking over the default route for all traffic. The payload can then be in the normal routing table 0.

7.5. Bonding

A key feature of the FB6000 is the ability to bond multiple links at a per packet level.

Bonding works with routing and shapers together. (See Chapter 9 for details of shapers.)

The basic principle is that you have two or more routes that are identical (same target IP prefix) and have the same localpref, so that there is nothing to decide between them. As described above this normally means one of the routes is picked.

However, where the two (or more) routes are the same type of interface, and there are shapers applied to those routes, then a decisions is made on a per packet basis as to which interface to used. The shapers are used to decide which link is least *far ahead*. This means that traffic is sent down each link at the speed of that link.

To make this work to the best effect, set the tx speed of the shapers on the links to match the actual link speed. E.g. for broadband lines, set the speed to match the uplink from the FB6000.

Chapter 8. Profiles

Profiles allow you to enable/disable various aspects of the FB6000's configuration (and thus functionality) based on things such as time-of-day or presence/absence of Ping responses from a specified device.

8.1. Overview

A profile is a two-state control entity - it is either Active or Inactive ("On" or "Off", like a switch). Once a profile is defined, it can be referenced in various configuration objects where the profile state will control the behaviour of that object.

A profile's state is determined by one or more defined *tests*, which are performed periodically. If multiple tests are specified, then the overall test result will be pass only if all the individual test results are pass. Assuming the profile's state is Active, then when the overall test result has been 'fail' for a specified duration, the profile transitions to Inactive. Similary, once the overall test result has been 'pass' for a specified duration, the profile transitions to Active. These two durations are controlled by attributes and provide a means to 'filter' out short duration 'blips' that are of little interest.

An example of a test that can be performed is a Ping test - ICMP echo request packets are sent, and replies are expected. If replies are not being received, the test fails.

Profiles can be logically combined using familiar boolean terminology i.e. AND, OR and NOT, allowing for some complex profile logic to be defined that determines a final profile state from several conditions.

By combining profiles with the FB6000's event logging facilities, they can also be used for automated monitoring and reporting purposes, where profile state changes can be e-mailed direct from the FB6000. For example, a profile using a Ping test can be used to alert you via e-mail when a destination is unreachable.

The current state of all the profiles configured on your FB6000 can be seen by choosing the "Profiles" item in the "Status" menu.

Tip

You can also define dummy profiles that are permanantly Active or Inactive, which can be useful if you wish to temporarily disable some functionality without deleting configuration object(s). For example, you can force an FB105 tunnel to be Down, preventing traffic from being routed through it. Refer to Section 8.2.4 for details.

8.2. Creating/editing profiles

In the web user interface, profiles are created and edited by clicking on the "Profiles" category icon. A profile is defined by a `profile` top-level object.

8.2.1. Timing control

The following timing control parameters apply :-

- `interval` : the interval between tests being performed
- `timeout` : the duration that the overall test must have been failing for before the profile state changes to Inactive
- `recover` : the duration that the overall test must have been passing for before the profile state changes to Active

The `timeout` and `recover` parameters do not apply to manually set profiles (see Section 8.2.4) and those based on time-of-day (see Section 8.2.2.2).

8.2.2. Tests

8.2.2.1. General tests

'General' tests are provided for the following :-

- Routable addresses : the `route` attributes lists one or more IP addresses (full addresses, not CIDR prefix ranges) - only if *all* the addresses are 'routable' - i.e. there is an entry in the routing table that will match that address - will this test pass. Refer to Chapter 7 for discussion of routing tables and the routing logic used by the FB6000
- VRRP state : the `vrrp` attribute lists one or more Virtual Router group membership definitions (see Chapter 12) by name - if the FB6000 is not the master device in any of these Virtual Routers, this test will fail

If more than one of these general tests is selected (corresponding attribute specified), then they must all pass (along with all other tests defined) for the overall result to be pass.

8.2.2.2. Time/date tests

Time and/or date tests are specified by `date` and/or `time` objects, which are child objects of the `profile` object.

You can define multiple date ranges via multiple `date` objects - the date test will pass if the current date is within *any* of the defined ranges. Similarly, you can define multiple time ranges via multiple `time` objects - the time test will pass if the current time is within *any* of the defined ranges.

Tip

Unlike other tests the chanhe of state because of a date/time test takes effect immediately rather than waiting for several seconds to confirm it is still Saturday or some such.

8.2.2.3. Ping tests

Like time/date tests, a Ping test is specified by a `ping` object, as a child of the `profile` object. At most one Ping test can be defined per profile - logical combinations of profiles can be used to combine Ping tests if necessary.

8.2.3. Inverting overall test result

The tests described in the previous section are used to form an overall test result. Normally this overall result is used to determine the profile state using the mapping Pass > Active and Fail > Inactive. By setting the `invert` attribute to `true`, the overall result is inverted (Pass changed to Fail and vice-versa) first before applying the mapping.

8.2.4. Manual override

You can manually override all tests, and force the profile state using the `set` attribute - a value of `true` forces the state to Active, and `false` forces it to Inactive.

You can also configure the `set` attribute with a value of `control-switch`. This causes the profile to be set manually based on a *control switch* which is not stored in the configuration itself. The *switch* appears on the home web page allowing it to be turned on or off with one click. It can also be changed from the command line. You can restrict each switch to one or more specific users to define who has control of the switch. This control applies even if the user has no access to make configuration changes as the switch is not part of the config. The switch state is automatically stored in the *dynamic peristent data* (along with DHCP settings, etc), so survives a power cycle / restart.

Note that the value of the `invert` attribute is ignored when manual override is requested.

These fixed-state profiles can be used as simple on/off controls for configuration objects. The following shows an example of two such profiles, expressed in XML :-

```
<profile name="Off" set="false"/>
<profile name="On" set="true"/>
<profile name="IT-Support"
         comment="Allow IT support company access to server"
  set="control-switch"/>
```

Chapter 9. Traffic Shaping

The FB6000 includes traffic shaping functionality that allows you to control the speed of specific traffic flows through the FB6000. The FB6000 also provides *graphing* functionality, allowing specific traffic flows to be plotted on a graph image (PNG format) that the FB6000 generates. Within the FB6000, traffic shaping and graphing are closely associated, and this is reflected in how you configure traffic shaping - in order to be able to perform traffic shaping, you must first graph the traffic flow.

9.1. Graphs and Shapers

9.1.1. Graphs

Several objects in the FB6000's configuration allow you to specify the name of a *graph*, by setting the value of the `graph` attribute. This causes the traffic flow that is associated with that object (a firewall rule, an interface, or whatever the attribute is attached to) to be recorded on a graph with the specified name. For connections that have a defined state, such as a PPP link, the graph will also show the link state history. Other information, such as packet loss and latency may also be displayed, depending on whether it can be provided by the type of object you are graphing.

For example, the XML snippet below shows the `graph` attribute being set on an `interface`. As soon as you have set a `graph` attribute (and saved the configuration), a new graph with the specified name will be created.

```
<interface name="LAN"
           port="LAN"
           graph="LAN">
```

The graph is viewable directly (as a PNG image) from the FB6000 via the web User Interface - to view a graph, click the "PNG" item in the "Graphs" menu. This will display all the graphs that are currently configured - it is not currently possible to show a single graph within the web User Interface environment.

It is possible to access the graph data in many ways, using the URL to control what information is shown, labels, and colours, and also allowing graphs to be archived. See Appendix E for more details.

Note

You may find images shown for graph names that are no longer specified anywhere in the configuration. Over time, these graphs will disappear automatically.

Alternatively, the underlying graph data is available in XML format, again via the FB6000's built-in HTTP server. The XML version of the data can be viewed in the web User Interface by clicking the "XML" item in the "Graphs" menu, and then clicking on the name of the graph you're interested in.

Both directions of traffic flow are recorded, and are colour-coded on the PNG image generated by the FB6000. The directions are labelled "tx" and "rx", but the exact meaning of these will depend on what type of object the graph was referenced from - for example, on a graph for an `interface`, "tx" will be traffic leaving the FB6000, and "rx" will be traffic arriving at the FB6000.

Each data point on a graph corresponds to a 100 second interval ; where a data point is showing a traffic rate, the rate is an average over that interval. For each named graph, the FB6000 stores data for the last 24 hours.

Note

Specifying a graph does not itself cause any traffic shaping to occur, but is a pre-requisite to specifying how the associated traffic flow should be shaped.

9.1.2. Shapers

Once you have graphed a (possibly bi-directional) traffic flow, you can then also define speed restrictions on those flows. These can be simple "Tx" and "Rx" speed limits or more complex settings allowing maximum average speeds over time.

You define the speed controls associated with the graphed traffic flow(s) by creating a `shaper` top-level object. To create or edit a `shaper` object in the web User Interface, first click on the "Shape" category icon. To create a new object, click the "Add" link. To edit an existing object, click the appropriate "Edit" link instead.

The `shaper` object specifies the parameters (primarily traffic rates) to use in the traffic shaping process, and the `shaper` is *associated* with the appropriate existing graph by specifying the `name` attribute of the `shaper` object to be the *same* as the name of the graph.

9.1.3. Ad hoc shapers

You can define a `shaper` object and set the speed controls for that shaper, and then define the `graph` attribute on something, e.g. an interface, to apply that shaper to the interface.

It is also possible, in most cases, to simply set a `speed` attribute on some object. This creates an un-named shaper (so no graph) which has the specified speed for egress (tx). This is unique to that object unlike named shapers which are shared between all objects using the same named shaper.

It is also possible to set `graph` and `speed` attributes to create a named shaper with the specified speed, without having to create a separate `shaper` object.

If you set a `graph` attribute without a `speed` attribute or creating a `shaper` object then that simply creates a graph without traffic shaping. Multiple objects can share the same graph.

9.1.4. Long term shapers

If defining a shaper using the `shaper` object there are a number of extra options which allow a long term shaper to be defined. A long term shaper is one that changes the actual speed applied dynamlically to ensure a long term usage level that is within a defined setting.

The key parameters for the long term shaper are the target speed (e.g. `tx`), the minimum speed (e.g. `tx-min`) and maximum speed (e.g. `tx-max`). The target speed is what is normally used if nothing else is set, but if a min and max are set then the shaper will actually use the max speed normally.

However, if the usage exceeds the target speed then this is considered to be *bursting* and this continues until the average speed since the bursting started drops below the target speed.

When bursting, initiall a time is allowed with no change of speed (e.g. `tx-min-burst`) and after that the speed drops. This can be automatic, or using a rate of drop per hour (e.g. `tx-step`). The rate will drop down to the defined minimum speed.

Once the average, since bursting started, drops below the target and the restrictions are lifted, returning to the maximum speed. If the minimum speed is below the target speed then this will happen eventually even if the link is used solidly at the maximum it is allowed. If the minimum is at the target or higher then the usage will have to drop below the target for a time before the average speed drops low enough to restore full speed.

The overall effect of this means that you can burst up to a specified maximum, but ultimately you cannot transfer more than if the target speed had been applied the whole time.

9.2. Multiple shapers

A packet that passes through the FB6000 can pass through multiple shapers, for example

- The ingress interface can have a defined shaper
- It is possible to create a bonded gateway route where multiple routes exist for the same target (typically a default gateway) and each route as a speed set, which is itself a shaper. This is used to control how much traffic goes via each of the bonded routes. (You simply create more than one `route` object with a `speed` or `graph` setting).
- The egress interface can have a defined shaper

9.3. Basic principles

Each shaper tracks how *far ahead* the link has got with traffic that has been recently sent. This depends on the length of packets sent and the speed of the shaper. This is, essentially, tracking how much is likely to be queued at a bottleneck further on. The FB6000 does not delay sending packets and assumes something with a lower speed is probably queuing them up later.

This record of how far ahead the traffic is gets used in two ways:

- If the shaper is too far ahead, then packets are dropped, causing the link to be rate limited to the selected speed. Exactly how much is *too far* depends on the packet size, with small packets (less than 1000 bytes) allowed more margin than large packets. This has the effect of prioritising DNS, interactive traffic, VoIP, etc.
- Where there are two or more links with shapers a link is picked based on which is the least *far ahead*. This has the effect of balancing the traffic levels between multiple links based on the speed of each link exactly.

Chapter 10. System Services

A *system service* provides general functionality, and runs as a separate concurrent process alongside normal traffic handling.

Table 10.1 lists the services that the FB6000 can provide :-

Table 10.1. List of system services

Service	Function
SNMP server	provides clients with access to management information using the Simple Network Management Protocol
NTP client	automatically synchronises the FB6000's clock with an NTP time server (usually using an Internet public NTP server)
Telnet server	provides an administration command-line interface accessed over a network connection
HTTP server	serves the web user-interface files to a user's browser on a client machine
DNS	relays DNS requests from either the FB6000 itself, or client machines to one or more DNS *resolvers*

Services are configured under the "Setup" category, under the heading "General system services", where there is a single services object (XML element : `<services>`). The services object doesn't have any attributes itself, all configuration is done via child objects, one per service. If a service object is not present, the service is disabled. Clicking on the Edit link next to the services object will take you to the lists of child objects. Where a service object is not present, the table in that section will contain an "Add" link. A maximum of one instance of each service object type can be present.

10.1. Protecting the FB6000

The FB6000 does not have a firewall as such. However, the design of the FB6000 is that it should be able to protect itself sensibly without the need for a separate firewall.

Each service has specific access control settings, and these default to not allowing external access (i.e. traffic not from locally Ethernet connected devices. You can also lock down access to a specific routing table, and restrict the source IP addresses from which connections are accepted.

In the case of the web interface, you can also define trusted IP addresses which are given priority access to the login page even.

10.2. Common settings

Most system service have common access control attributes as follows.

Tip

You can verify whether the access control performs as intended using the diagnostic facility described in Section 11.1

Table 10.2. List of system services

Attribute	Function

`table`	If specified, then the service only accepts requests/connections on the specified routing table. If not specified then the service works on any routing table. Where the service is also a client then this specifies the routing table to use (default 0).
`allow`	If specified then this is a list of ranges of IP addresses and ip group names from which connections are allowed. If specified as an empty list then no access is allowed. If omitted then access is allowed from everywhere. Note that if `local-only` is specified, the allow list allows access from addresses that are not local, if they are in the `allow` list.
`local-only`	This normally defaults to `true`, but not in all cases. If *true* then access is only allowed from machines on IPs on the local subnet[a] (and any addresses in the `allow` list, if specified).
`log`	The standard `log`, `log-error`, and `log-debug` settings can be used to specified levels of logging for the service.

[a]A *locally-attached* subnet is one which can be directly reached via one of the defined interfaces, i.e. is not accessed via a gateway.

Tip

Address ranges in `allow` can be entered using either <first address>-<last_address> syntax, or using CIDR notation : <start address>/<prefix length>. If a range entered using the first syntax can be expressed using CIDR notation, it will be automatically converted to that format when the configuration is saved. You can also use name(s) of defined IP address group(s), which are pre-defined ranges of IPs.

10.3. HTTP Server configuration

The HTTP server's purpose is to serve the HTML and supporting files that implement the web-based user-interface for the FB6000. It is not a general-purpose web server that can be used to serve user documents, and so there is little to configure.

10.3.1. Access control

By default, the FB6000 will allow access to the user interface from any machine, although obviously access to the user interface normally requires the correct login credentials to be provided. However, if you have no need for your FB6000 to be accessed from arbitrary machines, then you may wish to 'lock-down' access to the user interface to one or more client machines, thus removing an 'attack vector'.

Access can be restricted using `allow` and `local-only` controls as with any service. If this allows access, then a user can try and login. However, access can also be restricted on a per user basis to IP addresses and using profiles, which block the login even if the passord is correct.

Additionally, access to the HTTP server can be completely restricted (to all clients) under the control of a *profile*. This can be used, for example, to allow access only during certain time periods.

10.3.1.1. Trusted addresses

Trusted addresses are those from which additional access to certain functions is available. They are specified by setting the `trusted` attribute using address ranges or IP address group names. This *trusted* access allows visibility of graphs without the need for a password, and is mandatory for packet dump access.

10.4. Telnet Server configuration

The Telnet server allows standard telnet-protocol clients (available for most client platforms) to connect to the FB6000 and access a command-line interface (CLI). The CLI is documented in Chapter 13 and in the Appendix D.

10.4.1. Access control

Access control can be restricted in the same way as the HTTP (web) service, including per user access restrictions.

Note

By default, the FB6000 will only allow telnet access from machines that are on one of the locally-attached Ethernet subnets[a]. This default is used since the CLI offers a degree of system control that is not available via the web interface - for example, software images stored in the on-board Flash memory can be deleted via the CLI.

The example XML below shows the telnet service configured this way :-

```
<telnet allow="10.0.0.0/24 10.1.0.3-98 10.100.100.88 10.99.99.0/24"
          comment="telnet service access restricted by IP address"
          local-only="false"/>
```

10.5. DNS configuration

The DNS service provides name resolution service to other tasks within the app software, and can act as a relay for requests received from client machines. DNS typically means converting a name, like `www.firebrick.co.uk` to one or more IP addresses, but it can also be used for *reverse DNS* finding the name of an IP address. DNS service is normally provided by your ISP.

The DNS service on the FB6000 simply relays requests to external DNS servers and caches replies. You can configure a list of external DNS servers using the `resolvers` attribute. However, DNS resolvers are also learned automatically via various systems such as DHCP In most cases you do not need to set the resolvers.

10.5.1. Blocking DNS names

You can configure names such that the FB6000 issues an *NXDOMAIN* response making it appear that the domain does not exist. This can be done using a *wildcard*, e.g. you could block `*.xxx`.

Tip

You can also restrict responses to certain IP addresses on your LAN, making it that some devices get different responses. You can also control when responses are given using a *profile*, e.g. time of day.

10.5.2. Local DNS responses

Instead of blocking names, you can also make some names return pre-defined responses. This is usually only used for special cases, and there is a default for `my.firebrick.co.uk` which returns *the FireBrick's own IP*. Faking DNS responses will not always work, and new security measures such as DNSSEC will mean these faked responses will not be accepted.

10.5.3. Auto DHCP DNS

The FB6000 can also look for specific matching names and IP addresses for forward and reverse DNS that match machines on your LAN. This is done by telling the FireBrick the `domain` for your local network. Any name that is within that *domain* which matches a *client name* of a DHCP allocation that the FireBrick has made will return the IP address assigned by DHCP. This is applied in reverse for *reverse DNS* mapping an IP address back to a name. You can enable this using the `auto-dhcp` attribute.

10.6. NTP configuration

The NTP service automatically sets the FB6000's real-time-clock using time information provided by a Network Time Protocol (NTP) server. There are public NTP servers available for use on the Internet, and a factory reset configuration does not specify an NTP server which means a default of `ntp.firebrick.ltd.uk`. You can set your preferred NTP server instead.

The NTP service is currently only an NTP client. A future software version is likely to add NTP server functionality, allowing other NTP clients (typically those in your network) to use the FB6000 as an NTP server.

Configuration of the NTP (client) service typically only requires setting the `timeserver` attribute to specify one or more NTP servers, using either DNS name or IP address.

10.7. SNMP configuration

The SNMP service allows other devices to query the FB6000 for management related information, using the Simple Network Management Protocol (SNMP).

As with the HTTP server, access can be restricted to :-

- specific client IP addresses, and/or
- clients connecting from locally-attached Ethernet subnets only.

See Section 10.3.1 for details. The SNMP service defaults to allowing access from anywhere.

The remaining SNMP service configuration attributes are :-

- `community` : specifies the SNMP *community* name, with a default of `public`
- `port` : specifies the port number that the SNMP service listens on - this typically does not need setting, as the default is the standard SNMP port (161).

Chapter 11. Network Diagnostic Tools

Various network diagnostic tools are provided by the FB6000, accessible through either the web user interface or the CLI :-

- Packet dump : low level diagnostics to for detailed examination of network traffic passing through the FB6000
- Ping : standard ICMP echo request/reply ping mechanism
- Traceroute : classical traceroute procedure - ICMP echo request packets with increasing TTL values, soliciting "TTL expired" responses from routers along the path
- Access check : check whether a specific IP address is allowed to access the various network services described in Chapter 10

Each tool produces a textual result, and can be accessed via the CLI, where the *same* result text will be shown.

Caution

The diagnostic tools provided are *not* a substitute for external penetration testing - they are intended to aid understanding of FB6000 configuration, assist in development of your configuration, and for diagnosing problems with the behaviour of the FB6000 itself.

11.1. Access check

For each network service implemented by the FB6000 (see Chapter 10), this command shows whether a specific IP address will be able to access or utilise the service, based on any access restrictions configured on the service.

For example, the following shows some service configurations (expressed in XML), and the access check result when checking access for an external address, `1.2.3.4` :-

```
  <http local-only="false"/>
```

```
Web control page access via http:-
This address is allowed access to web control pages subject to
username/password being allowed.
```

```
  <telnet allow="admin-ips"
          local-only="false"/>
```

```
Telnet access:-
This address is not allowed access due to the allow list on telnet
service.
```

(in this example, `admin-ips` is the name of an IP address group that does not include `1.2.3.4`)

```
  <dns local-only="true"/>
```

```
DNS resolver access:-
This address is not on a local Ethernet subnet and so not allowed access.
```

11.2. Packet Dumping

The FireBrick includes the ability to capture packet dumps for diagnostic purposes. This might typically be used where the behaviour of the FB6000 is not as expected, and can help identify whether other devices are correctly implementing network protocols - if they are, then you should be able to determine whether the FB6000 is responding appropriately. The packet dumping facility may also be of use to you to debug traffic (and thus specific network protocols) between two hosts that the brick is routing traffic between.

This feature is provided via the FB6000's HTTP server and provides a download of a *pcap* format file (old format) suitable for use with `tcpdump` or *Wireshark*.

A packet dump can be performed by either of these methods :-

- via the user interface, using a web-page form to setup the dump - once the capture data has been downloaded it can be analysed using `tcpdump` or *Wireshark*
- using an HTTP client on another machine (typically a command-line client utility such as `curl`)

The output is streamed so that, when used with `curl` and `tcpdump`, you can monitor traffic in real time.

Limited filtering is provided by the FB6000, so you will normally apply any additional filtering you need via `tcpdump`.

11.2.1. Dump parameters

Table 11.1 lists the parameters you can specify to control what gets dumped. The "Parameter name" column shows the exact parameter name to specify when constructing a URL to use with an HTTP client. The "Web-form field" column shows the label of the equivalent field on the user interface form.

Table 11.1. Packet dump parameters

Parameter name	Web-form field	Function
interface	Interface	One or more interfaces, as the name of the interface. e.g. interface=WAN, also applies for name of PPPoE on an interface
snaplen	Snaplen	The maximum capture length for a packet can be specified, in bytes. Default 0 (auto). See notes below.
timeout	Timeout	The maximum capture time can be specified in seconds. Default 10.
ip	IP address *(2-off)*	Up to two IPs can be specified to filter packets
self	Include my IP	By default any traffic to or from the IP which is connecting to the web interface to access pcap is excluded. This option allows such traffic. Use with care else you dump your own dump traffic.

11.2.2. Security settings required

The following criteria must be met in order to use the packet dump facility :-

- You must be accessing from an IP listed as *trusted* in the HTTP service configuration (see Section 10.3).
- You must use a user and password for a "DEBUG level" user - the user level is set with the `level` attribute on the `user` object.

Note

These security requirements are the most likely thing to cause your attempts to packet dump to fail. If you are getting a simple "404" error response, and think you have specified the correct URL (if using an HTTP client), please check security settings are as described here.

11.2.3. IP address matching

You may optionally specify upto two IP address to be checked for a match in packets on the interface(s) and/or L2TP session(s) specified. If you do not specify any IP addresses, then all packets are returned. If you specify one IP address then all packets containing that IP address (as source or destination) are returned. If you specify two IP addresses then only those packets containing both addresses (each address being either as source or destination) are returned.

IP matching is only performed against ARP, IPv4 or IPv6 headers and not in encapsulated packets or ICMP payloads.

If capturing too much, some packets may be lost.

11.2.4. Packet types

The capture can collect different types of packets depending on where the capture is performed. All of these are presented as Ethernet frames, with faked Ethernet headers where the packet type is not Ethernet.

Table 11.2. Packet types that can be captured

Type	Notes
Ethernet	Interface based capture contains the full Ethernet frame with any VLAN tag removed.
IP	IP only, currently not possible to capture at this level. An Ethernet header is faked.
PPP	PPP from the protocol word (HDLC header is ignored if present). An Ethernet header is faked and also a PPPoE header. The PPPoE header has the session PPPoE ID that is the local end L2TP session ID.

The faked protocol header has target MAC of 00:00:00:00:00:00 and source MAC of 00:00:00:00:00:01 for received packets, and these reversed for sent packets.

11.2.5. Snaplen specification

The `snaplen` argument specifies the maximum length captured, but this applies at the protocol level. As such PPP packets will have up to the `snaplen` from the PPP protocol bytes and then have fake PPPoE and Ethernet headers added.

A `snaplen` value of 0 has special meaning - it causes logging of just IP, TCP, UDP and ICMP headers as well as headers in ICMP error payloads. This is primarily to avoid logging data carried by these protocols.

11.2.6. Using the web interface

The web form is accessed by selecting the "Packet dump" item under the "Diagnostics" main-menu item. Setup the dump parameters with reference to Table 11.1 and click the "Dump" button. Your browser will ask you to save a file, which will take time to save as per the timeout requested.

11.2.7. Using an HTTP client

To perform a packet dump using an HTTP client, you first construct an appropriate URL that contains standard HTTP URL form-style parameters from the list shown in Table 11.1. Then you retreive the dump from the FB6000 using a tool such as `curl`.

The URL is `http://`*`<FB6000 IP address or DNS name>`*`/pcap?parameter_name=value`*`[`*`&parameter_name=value ...`*`]`*

The URL may include as many *parameter name* and *value* pairs as you need to completely specify the dump parameters.

Packet capturing stops if the output stream (HTTP transfer) fails. This is useful if you are unable to determine a suitable timeout period, and would like to run an ongoing capture which you stop manually. This is achieved by specifying a very long duration, and then interrupting execution of the HTTP client using Ctrl+C or similar.

Only one capture can operate at a time. The HTTP access fails if no valid interfaces or sessions etc. are specified or if a capturing is currently running.

11.2.7.1. Example using curl and tcpdump

An example of a simple real-time dump and analysis run on a Linux box is shown below :-

```
curl --silent --no-buffer --user name:pass
   'http://1.2.3.4/pcap?interface=LAN&amp;timeout=300&amp;snaplen=1500'
   | /usr/sbin/tcpdump -r - -n -v
```

Note

Linebreaks are shown in the example for clarity only - they must not be entered on the command-line

In this example we have used username *name* and password *pass* to log-in to a FireBrick on address 1.2.3.4 - obviously you would change the IP address (or host name) and credentials to something suitable for your FB6000.

We have asked for a dump of the interface named `LAN`, with a 5 minute timeout and capturing 1500 byte packets. We have then fed the output in real time (hence specifying `--no-buffer` on the `curl` command) to `tcpdump`, and asked it to take capture data from the standard input stream (via the `-r -` options). We have additionally asked for no DNS resolution (`-n`) and verbose output (`-v`).

Consult the documentation provided with the client (e.g. Linux box) system for details on the extensive range of `tcpdump` options - these can be used to filter the dump to better locate the packets you are interested in.

Chapter 12. VRRP

The FB6000 supports VRRP (Virtual Router Redundancy Protocol), which is a system that provides routing redundancy, by enabling more than one hardware device on a network to act as a gateway for routing traffic. Hardware redundancy means VRRP can provide resilience in the event of device failure, by allowing a backup device to *automatically* assume the role of actively routing traffic.

12.1. Virtual Routers

VRRP abstracts a group of routers using the concept of a *virtual router*, which has a *virtual IP address*. The IP address is virtual in the sense that it is associated with more than one hardware device, and can 'move' between devices automatically.

The virtual IP address normally differs from the real IP address of any of the group members, but it can be the real address of the master router if you prefer (e.g. if short of IP addresses).

You can have multiple virtual routers on the same LAN at the same time, so there is a Virtual Router Identifier (VRID) that is used to distinguish them. The default VRID used by the FB6000 is `42`. You must set all devices that are part of the same group (virtual router) to the same VRID, and this VRID must differ from that used by any other virtual routers *on the same LAN*. Typically you would only have one virtual router on any given LAN, so the default of `42` does not normally need changing.

Note

You can use the same VRID on different VLANs without a clash in any way in the FB6000, however you may find some switches and some operatings systems do not work well and get confused about the same MAC appearing on different interfaces and VLANs. As such it is generally a good idea to avoid doing this unless you are sure your network will cope. i.e. use different VRIDs on different VLANs.

At any one time, one physical device is the *master* and is handling all the traffic sent to the virtual IP address. If the master fails, a backup takes over, and this process is transparent to other devices, which do not need to be aware of the change.

The members of the group communicate with each other using multicast IP packets.

The transparency to device failure is implemented by having group members all capable of receiving traffic addressed to the *same* single MAC address. A special MAC address is used, `00-00-5E-00-01-XX`, where `XX` is the VRID or VRRPv2, and `00-00-5E-00-02-XX` for VRRPv3.

The master device will reply with this MAC address when an ARP request is sent for the virtual router's IP address.

Since the MAC address associated with the virtual IP address does not change, ARP cache entries in other devices remain valid throughout the master / backup switch-over, and other devices are not even aware that the switch has happened, apart from a short 'black-hole' period until the backup starts routing.

When there is a switch-over, the VRRP packets that are multicast are sent from this special MAC, so network switches will automatically modify internal MAC forwarding tables, and start switching traffic to the appropriate physical ports for the physical router that is taking up the active routing role.

Note

You can disable the use of the special MAC if you wish, and use a normal FireBrick MAC. However, this can lead to problems in some cases.

12.2. Configuring VRRP

VRRP operates within a layer 2 broadcast domain, so VRRP configuration on the FB6000 comes under the scope of an `interface` definition. As such, to set-up your FB6000 to participate in a Virtual Router group, you need to create a `vrrp` object, as a child object of the `interface` that is in the layer 2 domain where the VRRP operates.

12.2.1. Advertisement Interval

A master indicates that it still 'alive' by periodically sending an advertisement multicast packet to the group members. A failure to receive a multicast packet from the master router for a period longer than three times the advertisement interval timer causes the backup routers to assume that the master router is down.

The interval is specified in multiples of 10ms, so a value of 100 represents one second. The default value, if not specified, is one second. If you set lower than one second then VRRP3 is used by default (see below). VRRP2 only does whole seconds, and must have the same interval for all devices. VRRP3 can have different intervals on different devices, but typically you would set them all the same.

The shorter the advertisement interval, the shorter the 'black hole' period, but there will be more (multicast) traffic in the network.

Note

For IPv6 VRRP3 is used by default, whereas for IPv4 VRRP2 is used by default. Devices have to be using the same version. IPv4 and IPv6 can co-exist with one using VRRP2 and the other VRRP3. Setting the same config (apart from priority) on all devices ensures they have the same version.

12.2.2. Priority

Each device is assigned a priority, which determines which device becomes the master, and which devices remain as backups. The (working) device with the highest priority becomes the master.

If using the real IP of the master, then the master should have priority 255. Otherwise pick priorities from 1 to 254. It is usually sensible to space these out, e.g. using 100 and 200. We suggest not setting priority 1 (see profiles and test, below).

12.3. Using a virtual router

A virtual router is used by another device simply by specifying the virtual-router's virtual IP address as the gateway in a route, rather than using a router's real IP address. From an IP point-of-view, the upstream device is completely unaware that the IP address is associated with a group of physical devices, and will forward traffic to the virtual IP address as required, exactly as it would with a single physical gateway.

12.4. VRRP versions

12.4.1. VRRP version 2

VRRP version 2 works with IPv4 addresses only (i.e. does not support IPv6) and whole second advertisement intervals only. The normal interval is one second - since the timeout is three times that, this means the fastest a backup can take over is just over 3 seconds. You should configure all devices in a VRRP group with the same settings (apart from their priority).

12.4.2. VRRP version 3

VRRP version 3 works in much the same way, but allows the advertisement interval to be any multiple of 10ms (1/100th of a second). The default interval is still 1 second, but it can now be set much faster - so although the timeout is still 3 times the interval, this means the backup could take over in as little as 30ms.

VRRP3 also works with IPv6. Whilst IPv4 and IPv6 VRRP are completely independent, you can configure both at once in a single `vrrp` object by listing one or more IPv4 addresses *and* one or more IPv6 addresses.

VRRP3 is used by default for any IPv6 addresses or where an interval of below one second is selected. It can also be specifically set in the config by setting the attribute `version3` to the value `"true"`.

Caution

If you have devices that are meant to work together as VRRP but one is version 2 and one is version 3 then they will typically not see each other and both become master. The FB6000's VRRP Status page shows if VRRP2 or VRRP3 is in use, and whether the FireBrick is master or not.

12.5. Compatibility

VRRP2 and VRRP3 are standard protocols and so the FB6000 can work alongside other devices that support VRRP2 or VRRP3.

Note that the FB6000 has non-standard support for some specific packets sent to the VRRP virtual addresses. This includes answering pings (configurable) and handling DNS traffic. Other VRRP devices may not operate in the same way and so may not work in the same way if they take over from the FireBrick.

Chapter 13. Command Line Interface

The FB6000 provides a traditional command-line interface (CLI) environment that can be used to check status information, and control some aspects of the unit's operation.

The CLI is accessed via the 'telnet' protocol - the FB6000 implements a telnet server, which you can connect to using any common telnet client program. To learn how to enable the telnet server, and to set-up access restrictions, please refer to Section 10.4.

The CLI is also available via the serial interface on the rear of the unit. This is normally set to 9k600 1N8. A USB serial lead is supplied.

Note

The CLI is not normally used to change the *configuration* of the unit - that must be done via the web interface. Whilst most commands can be carried out via the web interface, there are a few that can only be performed via the CLI.

The CLI has the following features :-

- full line-editing capabilities - that cursor-keys, backspace key and delete key function as expected allowing you to go back and insert/delete characters. You can press Enter at any point in the command-line text, and the full command text will be processed.
- command history memory - the CLI remembers a number of previously typed commands, and these can be recalled using the Up and Down cursor keys. Once you've located the required command, you can edit it if needed, and then press Enter.
- supports entering abbreviated commands - you only need to type sufficient characters to make the command un-ambiguous ; for example, 'show dhcp' and 'show dns' can be abbreviated to 'sh dh' and 'sh dn' respectively - 'show' is the only command word that begins "sh", and two characters of the second command word are sufficient to make it un-ambiguous.
- built-in command help - you can list all the available commands, and the CLI will also show the synopsis for each command. Typing the ? character at the command-prompt immediately displays this list (you do not have to press Enter). Alternatively, you can list all the possible completions of a part-typed command - in this case, typing the ? character after typing part of a command will list only commands that begin with the already-typed characters, for example, typing `tr ?` causes the CLI to respond as shown below :-

```
marty> tr

traceroute <IPNameAddr> [table=<routetable>] [source=<IPAddr>] ...
troff
tron

marty> tr
```

After listing the possible commands, the CLI re-displays the command line typed so far, which you can then complete.

Please refer to Appendix D for command details.

Appendix A. CIDR and CIDR Notation

Classless Inter-Domain Routing (CIDR) is a strategy for IP address assignment originally specified in 1993 that had the aims of "conserving the address space and limiting the growth rate of global routing state". The current specification for CIDR is in RFC4632 [http://tools.ietf.org/html/rfc4632].

The pre-CIDR era

CIDR replaced the original class-based organisation of the IP address space, which had become wasteful of address space, and did not permit *aggregation* of routing information.

In the original scheme, only three sizes of network were possible :

- Class A : 128 possible networks each with 16,777,216 addresses
- Class B : 16384 possible networks each with 65,536 addresses
- Class C : 2097152 possible networks each with 256 addresses

Every network, including any of the large number of possible Class C networks, required an entry in global routing tables - those used by core Internet routers - since it was not possible to aggregate entries that had the same routing information. The inability to aggregate routes meant global routing table size was growing fast, which meant performance issues at core routers.

The position and size of the network ID and host ID bitfields were implied by the bit pattern of some of the most significant address bits, which segmented the 32-bit IPv4 address space into three main blocks, one for each class of network.

CIDR

The prefix notation introduced by CIDR was, in the simplest sense, "to make explicit which bits in a 32-bit IPv4 address are interpreted as the network number (or prefix) associated with a site and which are the used to number individual end systems within the site". In this sense, the 'prefix' is the N most significant bits that comprise the network ID bitfield.

CIDR notation is written as :-

IPv4 : Traditional IPv4 'dotted-quad' number, followed by the "/" (slash) character, followed by a decimal prefix-length value between 0 and 32 (inclusive)

IPv6 : IPv6 address, followed by the "/" (slash) character, followed by a decimal prefix-length value between 0 and 128 (inclusive)

Where formerly only three network sizes were available, CIDR prefixes may be defined to describe *any* power of two-sized block of between one and 2^32 end system addresses, that begins at an address that is a multiple of the block size. This provides for far less wasteful allocation of IP address space. The size of the range is given by 2^M, where *M = 32 - prefix_length*

Routing destinations

As well as being used to define a network (subnet), the CIDR notation is used to define the *destination* in a routing table entry, which may encompass multiple networks (with longer prefixes) that are reachable by using the associated routing information. This, therefore, provides the ability to create aggregated routing table entries.

For example, a routing table entry with a destination of `10.1.2.0/23` specifies the address range `10.1.2.0` to `10.1.3.255` inclusive. As an example, it might be that in practice two /24 subnets are reachable via this routing table entry - 10.1.2.0/24 and 10.1.3.0/24 - routing table entries for these subnets would appear in a downstream router.

Note that in either a network/subnet or routing destination specification, the address will be the starting address of the IP address range being expressed, such that there will be M least significant bits of the address set to zero, where *M = 32 - prefix_length*

Combined interface IP address and subnet definitions

Another common use of the CIDR notation is to combine the definition of a network with the specification of the IP address of an end system on that network - this form is used in subnet definitions on the FB6000, and in many popular operating systems.

For example, the default IPv4 subnet on the `LAN` interface after factory reset is `10.0.0.1/24` - the address of the FB6000 on this subnet is therefore `10.0.0.1`, and the prefix length is 24 bits, leaving 8 bits for host addresses on the subnet. The subnet address range is therefore `10.0.0.0` to `10.0.0.255`

A prefix-length of 32 is possible, and specifies a block size of just one address, equivalent to a plain IP address specification with no prefix notation. This is not the same as a combined subnet and interface-IP-address definition, as it only specifies a single IP address.

General IP address range specifications

CIDR notation can also be used in the FB6000 to express general IP address ranges, such as in session-rules, trusted IP lists, access control lists etc. In these cases, the notation is the same as for routing destinations or subnets, i.e. the address specified is the starting address of the range, and the prefix-length determines the size of the range.

Appendix B. MAC Addresses usage

Ethernet networks use 48 bit MAC addresses. These are globally unique and allocated by the equipment manufacturer from a pool of addresses that is defined by the first three octets (bytes), which identify the organization, and are known as the Organizationally Unique Identifier (OUI). OUIs are issued by the IEEE - more information, and a searchable database of existing OUIs are available at http://standards.ieee.org/develop/regauth/oui/

MAC addresses are commonly written as six groups of two hexadecimal digits, separated by colons or hyphens.

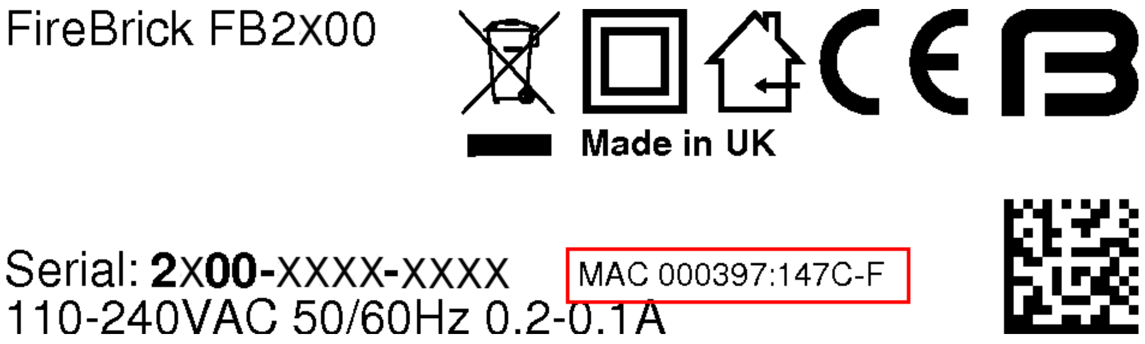
FB6000s currently ship with an OUI value of 00:03:97.

In principle the FireBrick could have a single MAC address for all operations. However, practical experience has led to the use of multiple MAC addresses on the FireBrick. A unique block of addresses is assigned to each FireBrick, with the size of the block dependent on the model.

Most of the time, FB6000 users do not need to know what MAC addresses the product uses. However, there are occasions where this information is useful, such as when trying to identify what IP address a DHCP server has allocated to a specific FB6000. For information on how MAC addresses are used by the FB6000, please refer to this article on the FireBrick website [http://www.firebrick.co.uk/fb2700/mac.php]

The label attached to the bottom of the FB6000 shows what MAC address range that unit uses, using a compact notation, as highlighted in Figure B.1 :-

Figure B.1. Product label showing MAC address range



In this example, the range is specified as :-

`000397:147C-F`

this is interpreted as :-

- All addresses in the range start with `00:03:97:14:7`
- the next digit then ranges from "C" through to "F"
  - the first address in the range has zero for the remaining digits (`C:00`)
  - the last address in the range has F for the remaining digits (`F:FF`)

Therefore this range spans `00:03:97:14:7C:00` to `00:03:97:14:7F:FF` inclusive (1024 addresses). If you trying to identify an IP address allocation, note that the exact address used within this range depends on a number of factors ; generally you should look for an IP address allocation against *any* of the addresses in the range.

Alternatively, if the range specification doesn't include a hyphen, it specifies that all addresses in the range start with this 'prefix' - the first address in the range will have zero for all the remaining digits, and the last address in the range will have F for all the remaining digits. For example :-

```
000397:147C
```

is interpreted as :

- All addresses in the range start with `00:03:97:14:7C`
  - the first address in the range has zero for the remaining digits (00)
  - the last address in the range has F for the remaining digits (FF)

Therefore this range spans `00:03:97:14:7C:00` to `00:03:97:14:7C:FF` inclusive (256 addresses).

If your DHCP server shows the name of the client (FB6000) that issued the DHCP request, then you will see a value that depends on whether the *system name* is set on the FB6000, as shown in Table B.1. Refer to Section 4.2.1 for details on setting the system name.

Table B.1. DHCP client names used

System name	Client name used
not set (e.g. factory reset configuration)	FB6000
set	Main application software running

If the FB6000's system name is set, and your DHCP server shows client names, then this is likely to be the preferred way to locate the relevant DHCP allocation in a list, rather than trying to locate it by MAC address. If the FB6000 is in a factory-reset state, then the system name will not be set, and you will have to locate it by MAC address.

Appendix C. VLANs : A primer

An Ethernet (Layer 2) broadcast domain consists of a group of Ethernet devices that are interconnected, typically via switches, such that an Ethernet broadcast packet (which specifies a reserved broadcast address as the destination Ethernet address of the packet) sent by one of the devices is always received by all the other devices in the group. A broadcast domain defines the boundaries of a single 'Local Area Network'.

When Virtual LANs (VLANs) are not in use, a broadcast domain consist of devices (such as PCs and routers), physical cables, switches (or hubs) and possibly bridges. In this case, creating a distinct Layer 2 broadcast domain requires a distinct set of switch/hub/bridge hardware, not physically interconnected with switch/hub/bridge hardware in any other domain.

A network using Virtual LANs is capable of implementing multiple distinct Layer 2 broadcast domains with *shared* physical switch hardware. The switch(es) used must support VLANs, and this is now common in cost-effective commodity Ethernet switches. Inter-working of VLAN switch hardware requires that all hardware support the same VLAN standard, the dominant standard being IEEE 802.1Q.

Such switches can seggregate physical switch ports into user-defined groups - with one VLAN associated with each group. Switching of traffic only occurs between the physical ports in a group, thus isolating each group from the others. Where more than one switch is used, with an 'uplink' connection between switches, VLAN *tagging* is used to multiplex packets from different VLANs across these single physical connections.

A IEEE 802.1Q VLAN tag is a small header prefixed to the normal Ethernet packet payload, includes a 12-bit number (range 1-4095) that identifies the tagged packet as belonging to a specific VLAN.

When a tagged packet arrives at another switch, the tag specifies which VLAN it is in, and switching to the appropriate physical port(s) occurs.

In addition to VLAN support in switches, some end devices incorporate VLAN support, allowing them to send and receive tagged packets from VLAN switch infrastructure, and use the VLAN ID to map packets to multiple logical interfaces, whilst only using a single physical interface. Such VLAN support is typically present in devices that are able to be multi-homed (have more than one IP interface), such as routers and firewalls, and general purpose network-capable operating systems such as Linux.

The FB6000 supports IEEE 802.1Q VLANs, and will accept (and send) packets with 802.1Q VLAN tags. It can therefore work with any Ethernet switch (or other) equipment that also supports 802.1Q VLANs, and therefore allows multiple logical interfaces to be implemented on a single physical port.

VLAN tagged switching is now also used in Wide-Area Layer 2 Ethernet networks, where a Layer 2 'circuit' is provided by a carrier over shared physical infrastructure. The conventional concept of a LAN occupying a small geographic area is thus no longer necessarily true.

Appendix D. Command line reference

D.1. General commands

D.1.1. Trace off

```
troff
```

Stop interactive logging to this CLI session, lasts until logout or **tron**.

D.1.2. Trace on

```
tron
```

Restart interactive logging to this CLI session. Some types of logging can be set to log to *console* which shows on the CLI.

D.1.3. Uptime

```
uptime
show uptime
```

Shows how long since the FB6000 restarted.

D.1.4. General status

```
show status
```

Shows general status information, including uptime, who owns the FireBrick, etc. This is the same as the Status on the web control pages.

D.1.5. Memory usage

```
show memory
```

Shows memory usage summary.

D.1.6. Process/task usage

```
show tasks
```

Shows internal task list. This is mainly for diagnostics purposes.

D.1.7. Login

```
login
```

Normally when you connect you are prompted for a username and password. If this is incorrect you can use the **login** to try again.

D.1.8. Logout

```
logout
quit
exit
```

You can also use **Ctrl-D** to exit, or close the connection (if using telnet)

D.1.9. See XML configuration

```
show run
show configuration
```

Dumps the full XML configuration to the screen

D.1.10. Load XML configuration

```
import configuration
```

You then send the XML configuration, ending with a blank line. You would not normally import a configuration using this command, as you can use the web interface, and tools like **curl** to load configtations. This command is provided as a last resort for emergency use, so use with care.

D.1.11. Show profile status

```
show profiles
```

Shows profiles and current status.

D.1.12. Enable profile control switch

```
enable profile <string>
```

Turns a named profile control switch on.

D.1.13. Disable profile control switch

```
disable profile <string>
```

Turns a named profile control switch off.

D.1.14. Show RADIUS servers

```
show radius
show radius <IPAddr>
```

Shows details of RADIUS servers currently in use

D.1.15. Show DNS resolvers

```
show dns
```

Shows current DNS resolver list and status.

D.2. Networking commands

D.2.1. Subnets

```
show subnets
show subnet <integer>
```

You can list all current subnets, or details of a specific subnet. This shows the same information as the web status pages for subnets.

D.2.2. Ping and trace

```
ping <IPNameAddr> [table=<routetable>] [source=<IPAddr>]
     [gateway=<IPAddr>] [flow=<unsignedShort>]
     [count=<positiveInteger>] [ttl=<unsignedByte>]
     [size=<unsignedShort>] [xml=<boolean>]
traceroute <IPNameAddr> [table=<routetable>] [source=<IPAddr>]
           [gateway=<IPAddr>] [flow=<unsignedShort>]
    [count=<positiveInteger>] [ttl=<unsignedByte>]
    [size=<unsignedShort>] [xml=<boolean>]
```

This sends a series of ICMP echo requests (ping) to a specified destination and confirms a response is received and the round trip time. For the **traceroute** variant, the TTL/Hopcount is increased by one each time to show a series of response hops. There are a number of controls allowing you to fine tune what is sent. Obviously you should only send from a *source* address that will return to the FB6000 correctly. You can also ask for the results to be presented in an XML format.

Where possible, the reverse DNS name is shown next to replies, but there is (deliberately) no delay waiting for DNS responses, so you may find it useful to run a trace a second time as results from the first attempt will be cached.

D.2.3. Show a route from the routing table

```
show route <IPPrefix> [table=<routetable>]
```

Shows details of a route in the routing table. Where an individual IP is used, the route that would be used is shown, but if a specifiy prefix is used then that specific route is shown even if there may be more specific routes in use.

D.2.4. List routes

```
show routes [<IPFilter>] [table=<routetable>]
```

Lists routes in the routing table, limited to those that match the filter if specified.

D.2.5. List routing next hops

```
show route nexthop [<IPAddr>]
```

List the next hop addresses currently in use and their status.

D.2.6. See DHCP allocations

```
show dhcp [<IP4Addr>] [table=<routetable>]
```

Shows DHCP allocations, with option to show details for specific allocation.

D.2.7. Clear DHCP allocations

```
clear dhcp [ip=<IP4Range>] [table=<routetable>]
```

Allows you to remove one or more DHCP allocations.

D.2.8. Lock DHCP allocations

```
lock dhcp ip=<IP4Addr> [table=<routetable>]
```

Locks a DHCP allocation. This stops the allocation being used for any other MAC address even if long expired.

D.2.9. Unlock DHCP allocations

```
unlock dhcp ip=<IP4Addr> [table=<routetable>]
```

Unlocks a DHCP allocation, allowing the address to be re-used if the expired.

D.2.10. Name DHCP allocations

```
name dhcp ip=<IP4Addr> [name=<string>] [table=<routetable>]
```

Allows you to set a name for a DHCP allocation, overridding the clientname that was sent.

D.2.11. Show ARP/ND status

```
show arp
show arp <IPAddr>
```

Shows details of ARP and Neighbour discovery cache.

D.2.12. Show VRRP status

```
show vrrp
```

Lists all VRRP in use and current status.

D.2.13. Send Wake-on-LAN packet

```
wol interface=<string> mac=<hexBinary>
```

Send a wake-on-LAN packet to a specific interface.

D.2.14. Check access to services

```
check access <IPAddr> [table=<routetable>]
```

Reports access control checks for a source address to various internal services. This is separate from any firewalling.

D.3. Advanced commands

Some commands are only available when logged in as a user set with *DEBUG* level access.

D.3.1. Panic

```
panic [<string>] [confirm=<string>]
```

This causes the FB6000 to crash, causing a *panic* event with a specified message. You need to specify **confirm=yes** for the command to work. This can be useful to test fallback scenarios by simulating a fatal error. Note that panic crash logs are emailed to the FireBrick support by default, so please use a meaningful string. e.g. **panic "testing fallback" confirm=yes**

D.3.2. Reboot

```
reboot [<unsignedInt>] [hard] [confirm=<string>]
```

A reboot is a more controlled shutdown and restart, unlike the **panic** command. The first argument is a block number (see **show flash contents**) and forces reboot to run a specific software stored in flash. Normally the reboot will run the latest valid code. The **hard** option forces the reboot to clear the Ethernet ports and other hardware so takes a couple of seconds. You must specify **confirm=yes** for this to work.

D.3.3. Screen width

```
set command screen width <unsignedInt>
```

This allows you to set the screen width.

D.3.4. Make outbound command session

```
start command session <IPAddr> [port=<unsignedShort>] [table=<routetable>]
```

This allows a *reverse telnet* connection to be made. A TCP connection is made to the IP address (and port) where a user can login. This can be useful where a firewall policy prevents incoming access to allow someone to have access from outside, e.g. the FireBrick support team.

D.3.5. Show command sessions

```
show command sessions
```

The FB6000 can have multiple telnet connections at the same time. This lists all of the current connections.

D.3.6. Kill command session

```
kill command session <IPAddr>
```

You can kill a command session by IP address. This is useful if you know you have left a telnet connected from somewhere else. Telnet sessions usually have a timeout, but this can be overridden in the configuration for each user.

D.3.7. Flash memory list

```
show flash contents
```

Lists the content of flash memory - this includes various *files* such as software releases, configuration, and so on. Multiple copies are usually stored allowing you to delete a later version if needed, and *roll-back* to an older version.

D.3.8. Delete block from flash

```
delete config <unsignedInt> [confirm=<string>]
delete data <unsignedInt> [confirm=<string>]
delete image <unsignedInt> [confirm=<string>]
```

Delete a block from flash memory. This cannot be undone. You have to specify the correct type of block, and specify **confirm=yes** for the command to work.

D.3.9. Boot log

```
show boot log [<unsignedInt>]
```

Show log of recent boots. You can specify the number of bytes of recent log to show.

D.3.10. Flash log

```
show flash log [<unsignedInt>]
```

The logging system can log to flash for a permanent record. This is done automatically for some system events and when booting. You can specify the number of bytes of recent log to show..

Appendix E. Constant Quality Monitoring - technical details

The FireBrick provides constant quality monitoring. The main purpose of this is to provide a graphical representation of the performance of an interface or traffic shaper

- 100 second interval statistics available graphically as png and in text as csv covering at least the last 25 hours (one day)
- Loss latency stats where available including minimum, average, and maximum latency for the 100 second sample, and percentage packet loss.
- Throughput stats where available (e.g. interfaces, shapers) including average tx and rx rate for 100 second sample

Graphs can be loss/latency or throughput of both. A ping only system would only have loss/latency. An interface or shaper normally has only throughput data.

E.1. Access to graphs and csvs

Graphs can be accessed by http using the normal web management interface. This can be used as a direct link from a web browser, or using common tools such as curl and wget.

The web management interface (services/http) define the port, and allowed user list and also a trusted IP access list. The CQM config defines a secret which is used to authorise untrusted access using an SHA1 hash in the URL.

All CQM URLs are in the /cqm/ path.

E.1.1. Trusted access

To access a graph you simply need to request the URL that is the graph name, followed by the file extension. E.g. http://host:port/cqm/circuit.png.

Table E.1. File types

Extn	Format
png	PNG image
csv	COMMA separated values list
tsv	TAB separated values list
txt	SPACE separated values list
xml	XML data

E.1.2. Dated information

Without any date the data returned is the latest. For csv it is all data points available. For graph it is the last 24 to 25 hours.

You can display data for a specific date. This only makes sense for *today*, and during the first couple of hours of the day you can get *yesterday* in full.

The syntax is that of a date first in the form YYYY-MM-DD/, e.g. http://host:port/cqm/YYYY-MM-DD/circuit.png.

E.1.3. Authenticated access

Authenticate access requires a prefix of a hex sha1 string. e.g. http://host:port/cqm/longhexsha1/circuit.png or http://host:port/cqm/longhexsha1/YYYY-MM-DD/circuit.png.

The SHA1 is 40 character hex of the SHA1 hash made from the graph name, the date, and the http-secret. The date is in the form YYYY-MM-DD, and is today's date for undated access (based on local time).

This means a graph URL can be composed that is valid for a specific graph name for a specific day.

Note that an MD5 can also be used instead but the SHA1 is the preferred method.

E.2. Graph display options

The graphs can have a number of options which define the colours, text and layout. These are defined as http form get attributes on the URL, e.g. http://host:port/cqm/circuit.png?H=a+heading.

Note that they can also be included in the path before the graph name, e.g. http://host:port/cqm/H=a+heading/circuit.png in which case they can be separated by / rather than &.

The attributes are processed in order.

E.2.1. Data points

The data point controls can be included as either fieldname or fieldname=colour. To make a valid URL either escape the # prefix or omit it. If any of these are included, then only those that are included are shown. If just fieldname is specified then the default colour is applied. The text on the right shows what fields are included and their colour key.

Table E.2. Colours

Key	Colour
M	Defines colour for minimum latency
A	Defines colour for average latency
X	Defines colour for max latency
U	Defines colour for upload rate
D	Defines colour for download rate
S	Defines colour for sent echos
J	Defines colour for rejected echos
F	Defines colour for failed (no response) echos
O	Defines colour for off-line

E.2.2. Additional text

Additional text is shown on the graph based on the values in the configuration if not specified. There are 4 lines on the top left in small text and two heading lines top right in large text.

Table E.3. Text

Key	Text
z	Clean output, clears all additional text fields

Z	Clean and clear, as *z* but also sets inside background and off-line colours to transparent so graphs are easy to merge with those other LNSs
C	Line 1 top left text, default if not set in config is system name
c	Line 2 top left text
N	Line 3 top left text
n	Line 4 top left text
H	Main heading text, default if not set in config is graph name
h	Sub heading text

E.2.3. Other colours and spacing

Colours can be in the form of RGB, RRGGBB, RGBA, RRGGBBAA defining red/green/blue/alpha, or some simple colour names.

Table E.4. Text

Key	Meaning
L	Defines a number of pixels to be provided on the left of the graph. Bandwidth and scale axis shown based on space provided left and right.
R	Defines a number of pixels to be provided on the right of the graph. Bandwidth and scale axis is shown based on space provided left and right.
T	Defines a number of pixels to be provided on the top of the graph. Time axes is show based on space at top and bottom.
B	Defines a number of pixels to be provided on the bottom of the graph. Time axes is show based on space at top and bottom.
Y	Defines Y bandwidth scale starting point (0 is lowest, 1 is next, etc).
y	Defines Y ms scale max level (in ms).
I	Defines colour for graticule
i	Defines colour for axis lines
g	Defines colour for background within axis
G	Defines colour for background outside axis
W	Defines colour for writing (text)

E.3. Overnight archiving

The system is designed to make it easy to archive all graphs or png, xml, etc files over night. The graphs hold 1000 data points, which is 27 hours 46 minutes. This means you can access a full day's data for the previous day in the first 3 hours 46 minutes of the new day (2 hours 46 or 4 hours 46 when clocks change in previous day). As such it is recommended that over night archiving is done of the previous day just after midnight.

The recommended command to run just after midnight is `wget -m http://host:port/cqm/`date +%F`/z/` as this will create a directory for the server, cqm, date, and z, and then the files. The use of `z` clears text off the graphs to make them clean.

E.3.1. Full URL format

The full URL format allows several variations. These are mainly to allow sensible directory structures in overnight archiving.

Table E.5. URL formats

URL	Meaning
/cqm/	All CQM URLs start with this
32-hex-characters/	Optional authentication string needed for untrusted access. Can be used with trusted access to test the authentication is right
YYYY-MM-DD/	Optional date to restrict output. Can also be in the form YYYY/MM/DD, YYYY-MM/DD, YYYY/MM-DD if preferred. Can also have /HH or -HH on the end to get data for just one hour, and /HH-HH, or -HH-HH on the end for a specific range of hours. Can end / HH:MM:SS or -MM:MM:SS for data for one hour from a specific time.
options/	Optional graph colour control options. Useful when extracting a list of images as the all must have the same options as the list is just graphname.png as a relative link thereby ensuring all graphs appear in this directory. The options list can include / separators rather & separators to make apparent subdirectories.
ext/	The file extension can be included on the end of the options, this is used only for making the index of all graphs for that type (see below)
graphname	Graph name. For XML this can be just * to produce one XML file with all graphs.
.ext	Extension for file type required
?options	Options can alternatively be included as a html form get field list

Where no graph name or ext are provided, i.e. the index page of a directory then an html page is served. An ext/ can be included after any options to make a list of files of that type. Otherwise the index is an html page explaining the options.

A blank graph is available by accessing simply `.png` (i.e. no graph name).

An xml list of all graphs is available as `.xml`.

A csv list of graph name and score is available as `.csv` and similarly for txt and tsv.

A special case exists for extracting the xml files for all graphs in one request, using the name `*.xml`.

E.3.2. load handling

The graphs and csv files are generated on the fly, and only one is generated at a time. Connection requests are queued. As part of the normal web management system, the trusted IPs queue is always processed first so constant access from untrusted sources will not stop access from trusted sources. If the queue is full the connection is not accepted. The most load applies when archiving, but tools like wget fetch one linked file at a time which is ideal.

E.4. Graph scores

Graphs are scored based on settings in the config. Each 100 second sample has a score which is included in the csv and xml lists for any graph. The score is also totalled for a graph as a whole and included in the csv and xml list of all graphs. This total is done by multiplying the last score by 864, the previous by 863, and so on for the previous 24 hours.

E.5. Creating graphs, and graph names

Graph names are text and up to 20 characters. Only letters, numbers, @, -, and . are allowed. All other characters are removed. It is recommended that names complying with this are used. Any graph name that you try and use that is too long will be replaced with one that uses part of the name and a hash to try and ensure a consistent unique graph name is applied.

Graphs can be defined in some configuration settings such as interface names.

The number of graphs is limited depending on memory, but the design is to allow for 100,000 distinctly named graphs. Dynamic circuits simply do not have graphs on them if this number is exceeded. Graphs not used for more than the data retention time are discarded automatically.

E.6. Ping

Models of the FB6000 that are designed to do constant pinging, e.g. the FB6102, allow graphs for pings using `<ping.../>` in the config, but they also allow pings to be started and stopped using a web interface. The Ping menu will show under config as well allowing input using a web form.

To start a ping, send a GET or POST to the `/ping` URL with form fields. HTTP Basic auth can be used instead of a normal session tracked login. The field `graph` must be specified with the graph name for the pinging. To start a ping also specify `ip` and optionally `table`. This can be used on an existing graph to change the ping target. To stop a ping omit `ip`.

There is also a means to define a bulk ping for the FB6102. The config allows a URL to be specified. This is fetched at startup and defined intervals (default 1 hour). It is expected to contain plain text which has on each line the graph name and IP address to ping. Any pings previously included but not now included are automatically stopped. The web interface also allows for the URL to be fetched manually if needed.

Appendix F. Configuration Objects

This appendix defines the object definitions used in the FireBrick FB6102 Ping monitoring configuration. Copyright © 2008-13 FireBrick Ltd.

F.1. Top level

F.1.1. config: Top level config

The top level config element contains all of the FireBrick configuration data.

Table F.1. config: Attributes

Attribute	Type	Default	Description
patch	integer	-	Internal use, for s/w updates that change config syntax
timestamp	dateTime	-	Config store time, set automatically when config is saved

Table F.2. config: Elements

Element	Type	Instances	Description
blackhole	blackhole	Optional, unlimited	Black hole (dropped packets) networks
cqm	cqm	Optional	Constant Quality Monitoring config
ethernet	ethernet	Optional, unlimited	Ethernet port settings
interface	interface	Optional, up to 8192	Ethernet interface (port-group/vlan) and subnets
ip-group	ip-group	Optional, unlimited	Named IP groups
log	log	Optional, up to 50	Log target controls
loopback	loopback	Optional, unlimited	Extra local addresses
nowhere	blackhole	Optional, unlimited	Dead end (icmp error) networks
ping	ping	Optional, up to 100	Base ping graph settings
port	portdef	Optional, up to 2	Port grouping and naming
route	route	Optional, unlimited	Static routes
services	services	Optional	General system services
system	system	Optional	System settings
user	user	Optional, unlimited	Admin users

F.2. Objects

F.2.1. system: System settings

The system settings are the top level attributes of the system which apply globally.

Table F.3. system: Attributes

Attribute	Type	Default	Description

comment	string	-	Comment
contact	string	-	Contact name
dos-delay	unsignedInt	2	Interrupt DoS restoration counter, leave at default
dos-limit	unsignedInt	1000	Interrupt DoS packet limit, leave at default
intro	string	-	Home page text
location	string	-	Location description
log	NMTOKEN	Web/console	Log system events
log-debug	NMTOKEN	Not logging	Log system debug messages
log-error	NMTOKEN	Web/Flash/console	Log system errors
log-eth	NMTOKEN	Web/console	Log Ethernet messages
log-eth-debug	NMTOKEN	Not logging	Log Ethernet debug
log-eth-error	NMTOKEN	Web/Flash/console	Log Ethernet errors
log-panic	NMTOKEN	Web logs	Log system panic messages
log-stats	NMTOKEN	Not logging	Log one second stats
name	string	-	System hostname
source	string	-	Source of data, used in automated config management
sw-update	autoloadtype	false	Load new software automatically

Table F.4. system: Elements

Element	Type	Instances	Description
link	link	Optional, unlimited	Home page links

F.2.2. link: Web links

Links to other web pages

Table F.5. link: Attributes

Attribute	Type	Default	Description
comment	string	-	Comment
name	string	-	Link name
source	string	-	Source of data, used in automated config management
text	string	-	Link text
url	string	-	Link address

F.2.3. user: Admin users

User names, passwords and abilities for admin users

Table F.6. user: Attributes

Attribute	Type	Default	Description

allow	*List of* IPNameRange	-	Restrict logins to be from specific IP addresses
comment	string	-	Comment
config	config-access	full	Config access level
full-name	string	-	Full name
level	user-level	ADMIN	Login level
name	*(NMTOKEN)* username	*Not optional*	User name
otp	string	-	OTP serial number
password	Password	*Not optional*	User password
source	string	-	Source of data, used in automated config management
table	*(unsignedByte 0-99)* routetable	0	Restrict login to specific routing table
timeout	duration	5:00	Login idle timeout (zero to stay logged in)

F.2.4. log: Log target controls

Named logging target

Table F.7. log: Attributes

Attribute	Type	Default	Description
colour	Colour	-	Colour used in web display
comment	string	-	Comment
console	boolean	-	Log immediately to console
flash	boolean	-	Log immediately to slow flash memory (use with care)
jtag	boolean	-	Log immediately jtag (development use only)
name	NMTOKEN	*Not optional*	Log target name
source	string	-	Source of data, used in automated config management

Table F.8. log: Elements

Element	Type	Instances	Description
email	log-email	Optional, unlimited	Email settings
syslog	log-syslog	Optional, unlimited	Syslog settings

F.2.5. log-syslog: Syslog logger settings

Logging to a syslog server

Table F.9. log-syslog: Attributes

Attribute	Type	Default	Description
comment	string	-	Comment

facility	syslog-facility	LOCAL0	Facility setting
port	unsignedShort	514	Server port
server	IPNameAddr	*Not optional*	Syslog server
severity	syslog-severity	NOTICE	Severity setting
source	string	-	Source of data, used in automated config management
source-ip	IPAddr	-	Use specific source IP
table	*(unsignedByte 0-99)* routetable	0	Routing table number for sending syslogs

F.2.6. log-email: Email logger settings

Logging to email

Table F.10. log-email: Attributes

Attribute	Type	Default	Description
comment	string	-	Comment
delay	duration	1:00	Delay before sending, since first event to send
from	string	One made up using serial number	Source email address
hold-off	duration	1:00:00	Delay before sending, since last email
log	NMTOKEN	Not logging	Log emailing process
log-debug	NMTOKEN	Not logging	Log emailing debug
log-error	NMTOKEN	Not logging	Log emailing errors
port	unsignedShort	25	Server port
retry	duration	10:00	Delay before sending, since failed send
server	IPNameAddr	-	Smart host to use rather than MX
source	string	-	Source of data, used in automated config management
subject	string	From first line being logged	Subject
table	*(unsignedByte 0-99)* routetable	0	Routing table number for sending email
to	string	*Not optional*	Target email address

F.2.7. services: System services

System services are various generic services that the system provides, and allows access controls and settings for these to be specified. The service is only active if the corresponding element is included in services, otherwise it is disabled.

Table F.11. services: Elements

Element	Type	Instances	Description
dns	dns-service	Optional	DNS service settings

http	http-service	Optional	HTTP server settings
ntp	ntp-service	Optional	NTP client settings (server not implemented yet)
snmp	snmp-service	Optional	SNMP server settings
telnet	telnet-service	Optional	Telnet server settings

F.2.8. snmp-service: SNMP service settings

The SNMP service has general service settings and also specific attributes for SNMP such as community

Table F.12. snmp-service: Attributes

Attribute	Type	Default	Description
allow	*List of* IPNameRange	Allow from anywhere	List of IP ranges from which service can be accessed
comment	string	-	Comment
community	string	public	Community string
local-only	boolean	false	Restrict access to locally connected Ethernet subnets only
log	NMTOKEN	Not logging	Log events
log-debug	NMTOKEN	Not logging	Log debug
log-error	NMTOKEN	Log as event	Log errors
port	unsignedShort	161	Service port
source	string	-	Source of data, used in automated config management
table	*(unsignedByte 0-99)* routetable	0	Routing table number

F.2.9. ntp-service: NTP service settings

The NTP settings define how the system clock is set, from what servers, and controls for daylight saving (summer time). The defaults are those that apply to the EU

Table F.13. ntp-service: Attributes

Attribute	Type	Default	Description
allow	*List of* IPNameRange	Allow from anywhere	List of IP ranges from which service can be accessed
comment	string	-	Comment
local-only	boolean	true	Restrict access to locally connected Ethernet subnets only
log	NMTOKEN	Not logging	Log events
log-debug	NMTOKEN	Not logging	Log debug
log-error	NMTOKEN	Log as event	Log errors
ntpserver	*List of* IPNameAddr	ntp.firebrick.ltd.uk	List of time servers (IP or hostname) from which time may be set by ntp
poll	duration	1:00:00	NTP poll rate

source	string	-	Source of data, used in automated config management
table	*(unsignedByte 0-99)* routetable	0	Routing table number
tz1-name	string	GMT	Timezone 1 name
tz1-offset	duration	0	Timezone 1 offset from UTC
tz12-date	*(unsignedByte 1-31)* datenum	25	Timezone 1 to 2 earliest date in month
tz12-day	day	Sun	Timezone 1 to 2 day of week of change
tz12-month	month	Mar	Timezone 1 to 2 month
tz12-time	time	01:00:00	Timezone 1 to 2 local time of change
tz2-name	string	BST	Timezone 2 name
tz2-offset	duration	1:00:00	Timezone 2 offset from UTC
tz21-date	*(unsignedByte 1-31)* datenum	25	Timezone 2 to 1 earliest date in month
tz21-day	day	Sun	Timezone 2 to 1 day of week of change
tz21-month	month	Oct	Timezone 2 to 1 month
tz21-time	time	02:00:00	Timezone 2 to 1 local time of change

F.2.10. telnet-service: Telnet service settings

Telnet control interface

Table F.14. telnet-service: Attributes

Attribute	Type	Default	Description
allow	*List of* IPNameRange	Allow from anywhere	List of IP ranges from which service can be accessed
comment	string	-	Comment
local-only	boolean	true	Restrict access to locally connected Ethernet subnets only
log	NMTOKEN	Not logging	Log events
log-debug	NMTOKEN	Not logging	Log debug
log-error	NMTOKEN	Log as event	Log errors
port	unsignedShort	23	Service port
source	string	-	Source of data, used in automated config management
table	*(unsignedByte 0-99)* routetable	0	Routing table number

F.2.11. http-service: HTTP service settings

Web management pages

Table F.15. http-service: Attributes

Attribute	Type	Default	Description

access-control-allow-origin	string	-	Additional header for cross site javascript
allow	*List of* IPNameRange	Allow from anywhere	List of IP ranges from which service can be accessed
comment	string	-	Comment
css-url	string	-	Additional CSS for web control pages
local-only	boolean	true	Restrict access to locally connected Ethernet subnets only
log	NMTOKEN	Not logging	Log events
log-debug	NMTOKEN	Not logging	Log debug
log-error	NMTOKEN	Log as event	Log errors
port	unsignedShort	80	Service port
source	string	-	Source of data, used in automated config management
table	*(unsignedByte 0-99)* routetable	0	Routing table number
trusted	*List of* IPNameRange	-	List of allowed IP ranges from which additional access to certain functions is available

F.2.12. dns-service: DNS service settings

DNS forwarding resolver service

Table F.16. dns-service: Attributes

Attribute	**Type**	**Default**	**Description**
allow	*List of* IPNameRange	Allow from anywhere	List of IP ranges from which service can be accessed
auto-dhcp	boolean	-	Forward and reverse DNS for names in DHCP using this domain
comment	string	-	Comment
domain	string	-	Our domain
local-only	boolean	true	Restrict access to locally connected Ethernet subnets only
log	NMTOKEN	Not logging	Log events
log-debug	NMTOKEN	Not logging	Log debug
log-error	NMTOKEN	Log as event	Log errors
resolvers	*List of* IPAddr	-	Recursive DNS resolvers to use
source	string	-	Source of data, used in automated config management
table	*(unsignedByte 0-99)* routetable	0	Routing table number

Table F.17. dns-service: Elements

Element	**Type**	**Instances**	**Description**

block	dns-block	Optional, unlimited	Fixed local DNS host blocks
host	dns-host	Optional, unlimited	Fixed local DNS host entries

F.2.13. dns-host: Fixed local DNS host settings

DNS forwarding resolver service

Table F.18. dns-host: Attributes

Attribute	Type	Default	Description
comment	string	-	Comment
ip	*List of* IPAddr	Our IP	IP addresses to serve (or our IP if omitted)
name	*List of* string	*Not optional*	Host names (can use * as a part of a domain)
restrict	*List of* IPNameRange	-	List of IP ranges to which this is served
reverse	boolean	-	Map reverse DNS as well
source	string	-	Source of data, used in automated config management
ttl	unsignedInt	60	Time to live

F.2.14. dns-block: Fixed local DNS blocks

DNS forwarding resolver service

Table F.19. dns-block: Attributes

Attribute	Type	Default	Description
comment	string	-	Comment
name	*List of* string	*Not optional*	Host names (can use * as a part of a domain)
restrict	*List of* IPNameRange	-	List of IP ranges to which this is served
source	string	-	Source of data, used in automated config management
ttl	unsignedInt	60	Time to live

F.2.15. ethernet: Physical port controls

Physical port attributes

Table F.20. ethernet: Attributes

Attribute	Type	Default	Description
autoneg	boolean	auto negotiate unless manual 10/100 speed and duplex are set	Perform link auto-negotiation
clocking	LinkClock	prefer-slave	Gigabit clock setting
crossover	Crossover	auto	Port crossover configuration
duplex	LinkDuplex	auto	Duplex setting for this port

flow	LinkFlow	none	Flow control setting
green	LinkLED-g	Link/Activity	Green LED setting
optimise	boolean	true	enable PHY optimisations
port	port	*Not optional*	Physical port
power-saving	LinkPower	full	enable PHY power saving
send-fault	LinkFault	-	Send fault status
shutdown	boolean	false	Power down this port
speed	LinkSpeed	auto	Speed setting for this port
yellow	LinkLED-y	Tx	Yellow LED setting

F.2.16. portdef: Port grouping and naming

Port grouping and naming

Table F.21. portdef: Attributes

Attribute	**Type**	**Default**	**Description**
comment	string	-	Comment
name	NMTOKEN	*Not optional*	Name
ports	*Set of* port	*Not optional*	Physical port(s)
source	string	-	Source of data, used in automated config management

F.2.17. interface: Port-group/VLAN interface settings

The interface definition relates to a specific physical port group and VLAN. It includes subnets and VRRP that apply to that interface.

Table F.22. interface: Attributes

Attribute	**Type**	**Default**	**Description**
comment	string	-	Comment
graph	*(token)* graphname	-	Graph name
link	NMTOKEN	-	Interface to which this is linked at layer 2
log	NMTOKEN	Not logging	Log events including DHCP and related events
log-debug	NMTOKEN	Not logging	Log debug
log-error	NMTOKEN	Log as event	Log errors
mtu	*(unsignedShort 576-2000)* mtu	1500	MTU for this interface
name	NMTOKEN	-	Name
ping	IPAddr	-	Ping address to add loss/latency to graph for interface
port	NMTOKEN	*Not optional*	Port group name
ra-client	boolean	true	Accept IPv6 RA and create auto config subnets and routes
restrict-mac	boolean	-	Use only one MAC on this interface

source	string	-	Source of data, used in automated config management
source-filter	sfoption	-	Source filter traffic received via this interface
table	*(unsignedByte 0-99)* routetable	0	Routing table applicable
vlan	*(unsignedShort 0-4095)* vlan	0	VLAN ID (0=untagged)

Table F.23. interface: Elements

Element	Type	Instances	Description
dhcp	dhcps	Optional, unlimited	DHCP server settings
subnet	subnet	Optional, unlimited	IP subnet on the interface
vrrp	vrrp	Optional, unlimited	VRRP settings

F.2.18. subnet: Subnet settings

Subnet settings define the IP address(es) of the FireBrick, and also allow default routes to be set.

Table F.24. subnet: Attributes

Attribute	Type	Default	Description
accept-dns	boolean	true	Accept DNS servers specified by DHCP
arp-timeout	unsignedShort	60	Max lifetime on ARP and ND
broadcast	boolean	false	If broadcast address allowed
comment	string	-	Comment
gateway	*List of* IPAddr	-	One or more gateways to install
ip	*List of* IPSubnet	Automatic by DHCP	One or more IP/len
localpref	unsignedInt	4294967295	Localpref for subnet (highest wins)
mtu	*(unsignedShort 576-2000)* mtu	As interface	MTU for subnet
name	string	-	Name
proxy-arp	boolean	false	Answer ARP/ND by proxy if we have routing
ra	ramode	false	If to announce IPv6 RA for this subnet
ra-dns	*List of* IP6Addr	-	List of recursive DNS servers in route announcements
ra-managed	dhcpv6control	-	RA 'M' (managed) flag
ra-max	*(unsignedShort 4-1800)* ra-max	600	Max RA send interval
ra-min	*(unsignedShort 3-1350)* ra-min	-	Min RA send interval
ra-mtu	unsignedShort	As subnet	MTU to use on RA
ra-other	dhcpv6control	-	RA 'O' (other) flag
source	string	-	Source of data, used in automated config management

test	IPAddr	-	Test link state using ARP/ND for this IP
ttl	unsignedByte	64	TTL for originating traffic via subnet

F.2.19. vrrp: VRRP settings

VRRP settings provide virtual router redundancy for the FireBrick. Profile inactive does not disable vrrp but forces vrrp low priority.

Table F.25. vrrp: Attributes

Attribute	Type	Default	Description
answer-ping	boolean	true	Whether to answer PING to VRRP IPs when master
comment	string	-	Comment
delay	unsignedInt	60	Delay after routing established before priority returns to normal
interval	unsignedShort	100	Transit interval (centiseconds)
ip	*List of* IPAddr	*Not optional*	One or more IP addresses to announce
log	NMTOKEN	Not logging	Log events
log-error	NMTOKEN	log as event	Log errors
low-priority	unsignedByte	1	Lower priority applicable until routing established
name	NMTOKEN	-	Name
preempt	boolean	true	Whether pre-empt allowed
priority	unsignedByte	100	Normal priority
source	string	-	Source of data, used in automated config management
test	*List of* IPAddr	-	List of IPs to which routing must exist else low priority (deprecated)
use-vmac	boolean	true	Whether to use the special VMAC or use normal MAC
version3	boolean	v2 for IPv4, v3 for IPv6	Use only version 3
vrid	unsignedByte	42	VRID

F.2.20. dhcps: DHCP server settings

Settings for DHCP server

Table F.26. dhcps: Attributes

Attribute	Type	Default	Description
boot	IP4Addr	-	Next/boot server
boot-file	string	-	Boot filename
class	string	-	Class match
client-name	string	-	Client name match
comment	string	-	Comment

dns	*List of* IP4Addr	Our IP	DNS resolvers
domain	string	From system settings	DNS domain
force	boolean	-	Send all options even if not requested
gateway	*List of* IP4Addr	Our IP	Gateway
ip	*List of* IP4Range	0.0.0.0/0	Address pool
lease	duration	2:00:00	Lease length
log	NMTOKEN	Not logging	Log events (allocations)
mac	*List up to 12 (hexBinary)* macprefix	-	Partial or full MAC addresses
name	string	-	Name
ntp	*List of* IP4Addr	From system settings	NTP server
source	string	-	Source of data, used in automated config management
syslog	*List of* IP4Addr	-	Syslog server
time	*List of* IP4Addr	Our IP	Time server

Table F.27. dhcps: Elements

Element	Type	Instances	Description
send	dhcp-attr-hex	Optional, unlimited	Additional attributes to send (hex)
send-ip	dhcp-attr-ip	Optional, unlimited	Additional attributes to send (IP)
send-number	dhcp-attr-number	Optional, unlimited	Additional attributes to send (numeric)
send-string	dhcp-attr-string	Optional, unlimited	Additional attributes to send (string)

F.2.21. dhcp-attr-hex: DHCP server attributes (hex)

Additional DHCP server attributes (hex)

Table F.28. dhcp-attr-hex: Attributes

Attribute	Type	Default	Description
comment	string	-	Comment
force	boolean	-	Send even if not requested
id	unsignedByte	*Not optional*	Attribute type code
name	string	-	Name
value	hexBinary	*Not optional*	Value
vendor	boolean	-	Add as vendor specific option (under option 43)

F.2.22. dhcp-attr-string: DHCP server attributes (string)

Additional DHCP server attributes (string)

Table F.29. dhcp-attr-string: Attributes

Attribute	Type	Default	Description

comment	string	-	Comment
force	boolean	-	Send even if not requested
id	unsignedByte	*Not optional*	Attribute type code
name	string	-	Name
value	string	*Not optional*	Value
vendor	boolean	-	Add as vendor specific option (under option 43)

F.2.23. dhcp-attr-number: DHCP server attributes (numeric)

Additional DHCP server attributes (numeric)

Table F.30. dhcp-attr-number: Attributes

Attribute	Type	Default	Description
comment	string	-	Comment
force	boolean	-	Send even if not requested
id	unsignedByte	*Not optional*	Attribute type code
name	string	-	Name
value	unsignedInt	*Not optional*	Value
vendor	boolean	-	Add as vendor specific option (under option 43)

F.2.24. dhcp-attr-ip: DHCP server attributes (IP)

Additional DHCP server attributes (IP)

Table F.31. dhcp-attr-ip: Attributes

Attribute	Type	Default	Description
comment	string	-	Comment
force	boolean	-	Send even if not requested
id	unsignedByte	*Not optional*	Attribute type code
name	string	-	Name
value	IP4Addr	*Not optional*	Value
vendor	boolean	-	Add as vendor specific option (under option 43)

F.2.25. route: Static routes

Static routes define prefixes which are permanently in the routing table, and whether these should be announced by routing protocols or not.

Table F.32. route: Attributes

Attribute	Type	Default	Description

comment	string	-	Comment
gateway	*List of* IPAddr	*Not optional*	One or more target gateway IPs
graph	*(token)* graphname	-	Graph name
ip	*List of* IPPrefix	*Not optional*	One or more network prefixes
localpref	unsignedInt	4294967295	Localpref of network (highest wins)
name	string	-	Name
source	string	-	Source of data, used in automated config management
speed	unsignedInt	-	Egress rate limit (b/s)
table	*(unsignedByte 0-99)* routetable	0	Routing table number

F.2.26. blackhole: Dead end networks

Networks that go nowhere

Table F.33. blackhole: Attributes

Attribute	Type	Default	Description
comment	string	-	Comment
ip	*List of* IPPrefix	*Not optional*	One or more network prefixes
localpref	unsignedInt	4294967295	Localpref of network (highest wins)
name	string	-	Name
source	string	-	Source of data, used in automated config management
table	*(unsignedByte 0-99)* routetable	0	Routing table number

F.2.27. loopback: Locally originated networks

Loopback addresses define local IP addresses

Table F.34. loopback: Attributes

Attribute	Type	Default	Description
comment	string	-	Comment
ip	*List of* IPAddr	*Not optional*	One or more local network addresses
localpref	unsignedInt	4294967295	Localpref of network (highest wins)
name	string	-	Name
source	string	-	Source of data, used in automated config management
table	*(unsignedByte 0-99)* routetable	0	Routing table number

F.2.28. cqm: Constant Quality Monitoring settings

Constant quality monitoring (graphs and data) have a number of settings. Most of the graphing settings can be overridden when a graph is collected so these define the defaults in many cases.

Table F.35. cqm: Attributes

Attribute	Type	Default	Description
ave	Colour	#08f	Colour for average latency
axis	Colour	black	Axis colour
background	Colour	white	Background colour
bottom	unsignedByte	11	Pixels space at bottom of graph
dateformat	string	%Y-%m-%d	Date format
dayformat	string	%a	Day format
fail	Colour	red	Colour for failed (dropped) seconds
fail-level	unsignedInt	1	Fail level not expected on low usage
fail-level1	unsignedByte	3	Loss level 1
fail-level2	unsignedByte	50	Loss level 2
fail-score	unsignedByte	200	Score for fail and low usage
fail-score1	unsignedByte	100	Score for on/above level 1
fail-score2	unsignedByte	200	Score for on/above level 2
fail-usage	unsignedInt	128000	Usage below which fail is not expected
fblogo	Colour	#bd1220	Colour for logo
graticule	Colour	grey	Graticule colour
heading	string	-	Heading of graph
hourformat	string	%H	Hour format
key	unsignedByte	90	Pixels space for key
label-ave	string	Av	Label for average latency
label-fail	string	%Fail	Label for seconds (%) failed
label-latency	string	Latency	Label for latency
label-max	string	Max	Label for maximum latency
label-min	string	Min	Label for minimum latency
label-off	string	Off	Label for off line seconds
label-period	string	Period	Label for period
label-poll	string	Polls	Label for polls
label-rej	string	%Reject	Label for rejected seconds
label-rx	string	Rx	Label for Rx traffic level
label-score	string	Score	Label for score
label-sent	string	Sent	Label for seconds polled
label-time	string	Time	Label for time
label-traffic	string	Traffic (bit/s)	Label for traffic level
label-tx	string	Tx	Label for Tx traffic level
latency-level	unsignedInt	100000000	Latency level not expected on low usage
latency-level1	unsignedInt	100000000	Latency level 1 (ns)
latency-level2	unsignedInt	500000000	Latency level 2 (ns)
latency-score	unsignedByte	200	Score for high latency and low usage
latency-score1	unsignedByte	10	Score for on/above level 1

latency-score2	unsignedByte	20	Score for on/above level 2
latency-usage	unsignedInt	128000	Usage below which latency is not expected
left	unsignedByte	0	Pixels space left of main graph
log	NMTOKEN	Not logging	Log events
max	Colour	green	Colour for maximum latency
min	Colour	#008	Colour for minimum latency
ms-max	positiveInteger	500	ms max height
off	Colour	#c8f	Colour for off line seconds
outside	Colour	transparent	Colour for outer border
ping-update	duration	1:00:00	Interval for periodic updates
ping-url	string	-	URL for ping list
rej	Colour	#f8c	Colour for off line seconds
right	unsignedByte	50	Pixels space right of main graph
rx	Colour	#800	Colour for Rx traffic level
secret	Secret	-	Secret for MD5 coded URLs
sent	Colour	#ff8	Colour for polled seconds
subheading	string	-	Subheading of graph
text	Colour	black	Colour for text
text1	string	-	Text line 1
text2	string	-	Text line 2
text3	string	-	Text line 3
text4	string	-	Text line 4
timeformat	string	%Y-%m-%d %H:%M:%S	Time format
top	unsignedByte	4	Pixels space at top of graph
tx	Colour	#080	Colour for Tx traffic level

F.2.29. ping: Ping/graph definition

Base ping config - additional ping targets set via web API or other means

Table F.36. ping: Attributes

Attribute	Type	Default	Description
comment	string	-	Comment
graph	*(token)* graphname	*Not optional*	Graph name
ip	IPNameAddr	*Not optional*	Far end IP
name	string	-	Name
size	*(unsignedInt 0-1472)* ping-size	0	Payload size
slow	boolean	Auto	Slow polling
source	string	-	Source of data, used in automated config management

table	*(unsignedByte 0-99)* routetable	0	Routing table number for sending pings

F.2.30. ip-group: IP Group

Named IP group

Table F.37. ip-group: Attributes

Attribute	Type	Default	Description
comment	string	-	Comment
ip	*List of* IPRange	-	One or more IP ranges or IP/len
name	string	*Not optional*	Name
source	string	-	Source of data, used in automated config management
users	*List of* NMTOKEN	-	Include IP of (time limited) logged in web users

F.3. Data types

F.3.1. autoloadtype: Type of s/w auto load

Table F.38. autoloadtype: Type of s/w auto load

Value	Description
false	Do no auto load
factory	Load factory releases
beta	Load beta test releases
alpha	Load test releases

F.3.2. config-access: Type of access user has to config

Table F.39. config-access: Type of access user has to config

Value	Description
none	No access unless explicitly listed
view	View only access (no passwords)
read	Read only access (with passwords)
full	Full view and edit access

F.3.3. user-level: User login level

User login level - commands available are restricted according to assigned level.

Table F.40. user-level: User login level

Value	Description
NOBODY	Unknown or not logged in user

GUEST	Guest user
USER	Normal unprivileged user
ADMIN	System administrator
DEBUG	System debugger

F.3.4. syslog-severity: Syslog severity

Log severity - different loggable events log at different levels.

Table F.41. syslog-severity: Syslog severity

Value	Description
EMERG	System is unstable
ALERT	Action must be taken immediately
CRIT	Critical conditions
ERR	Error conditions
WARNING	Warning conditions
NOTICE	Normal but significant events
INFO	Informational
DEBUG	Debug level messages
NO-LOGGING	No logging

F.3.5. syslog-facility: Syslog facility

Syslog facility, usually used to control which log file the syslog is written to.

Table F.42. syslog-facility: Syslog facility

Value	Description
KERN	Kernel messages
USER	User level messges
MAIL	Mail system
DAEMON	System Daemons
AUTH	Security/auth
SYSLOG	Internal to syslogd
LPR	Printer
NEWS	News
UUCP	UUCP
CRON	Cron deamon
AUTHPRIV	private security/auth
FTP	File transfer
12	Unused
13	Unused
14	Unused
15	Unused

LOCAL0	Local 0
LOCAL1	Local 1
LOCAL2	Local 2
LOCAL3	Local 3
LOCAL4	Local 4
LOCAL5	Local 5
LOCAL6	Local 6
LOCAL7	Local 7

F.3.6. month: Month name (3 letter)

Table F.43. month: Month name (3 letter)

Value	Description
Jan	January
Feb	February
Mar	March
Apr	April
May	May
Jun	June
Jul	July
Aug	August
Sep	September
Oct	October
Nov	November
Dec	December

F.3.7. day: Day name (3 letter)

Table F.44. day: Day name (3 letter)

Value	Description
Sun	Sunday
Mon	Monday
Tue	Tuesday
Wed	Wednesday
Thu	Thursday
Fri	Friday
Sat	Saturday

F.3.8. port: Physical port

Table F.45. port: Physical port

Value	Description

0	Port 0 (left)
1	Port 1 (right)

F.3.9. Crossover: Crossover configuration

Physical port crossover configuration.

Table F.46. Crossover: Crossover configuration

Value	Description
auto	Crossover is determined automatically
MDI	Force no crossover

F.3.10. LinkSpeed: Physical port speed

Table F.47. LinkSpeed: Physical port speed

Value	Description
10M	10Mbit/sec
100M	100Mbit/sec
1G	1Gbit/sec
auto	Speed determined by autonegotiation

F.3.11. LinkDuplex: Physical port duplex setting

Table F.48. LinkDuplex: Physical port duplex setting

Value	Description
half	Half-duplex
full	Full-duplex
auto	Duplex determined by autonegotiation

F.3.12. LinkFlow: Physical port flow control setting

Table F.49. LinkFlow: Physical port flow control setting

Value	Description
none	No flow control
symmetric	Can support two-way flow control
send-pauses	Can send pauses but does not support pause reception
any	Can receive pauses and may send pauses if required

F.3.13. LinkClock: Physical port Gigabit clock master/ slave setting

Table F.50. LinkClock: Physical port Gigabit clock master/slave setting

Value	Description
prefer-master	Master status negotiated; preference for master

prefer-slave	Master status negotiated; preference for slave
force-master	Master status forced
force-slave	Slave status forced

F.3.14. LinkLED-y: Yellow LED setting

Table F.51. LinkLED-y: Yellow LED setting

Value	Description
Duplex/ Collision	On when full-duplex; blink when half-duplex and collisions detected
Activity	Blink when Tx or Rx activity
Fault	On when autonegotiation mismatch
Tx	Blink when Tx activity
Off	Permanently off
On	Permanently on
Cycling	Cycling pattern

F.3.15. LinkLED-g: Green LED setting

Table F.52. LinkLED-g: Green LED setting

Value	Description
Link/Activity	On when link up; blink when Tx or Rx activity
Duplex/ Collision	On when full-duplex; blink when half-duplex and collisions detected
Rx	Blink when Rx activity
Off	Permanently off
On	Permanently on
Cycling	Cycling pattern

F.3.16. LinkPower: PHY power saving options

Table F.53. LinkPower: PHY power saving options

Value	Description
none	No power saving
full	Full power saving

F.3.17. LinkFault: Link fault type to send

Table F.54. LinkFault: Link fault type to send

Value	Description
false	No fault
true	Send fault
off-line	Send offline fault (1G)

ane	Send ANE fault (1G)

F.3.18. ramode: IPv6 route announce level

IPv6 route announcement mode and level

Table F.55. ramode: IPv6 route announce level

Value	Description
false	Do not announce
low	Announce as low priority
medium	Announce as medium priority
high	Announce as high priority
true	Announce as default (medium) priority

F.3.19. dhcpv6control: Control for RA and DHCPv6 bits

Table F.56. dhcpv6control: Control for RA and DHCPv6 bits

Value	Description
false	Don't set bit or answer on DHCPv6
true	Set bit but do not answer on DHCPv6
dhcpv6	Set bit and do answer on DHCPv6

F.3.20. sfoption: Source filter option

Table F.57. sfoption: Source filter option

Value	Description
false	No source filter checks
blackhole	Check replies have any valid route
true	Check replies down same port/vlan

F.4. Basic types

Table F.58. Basic data types

Type	Description
string	text string
token	text string
hexBinary	hex coded binary data
integer	integer (-2147483648-2147483647)
positiveInteger	positive integer (1-4294967295)
unsignedInt	unsigned integer (0-4294967295)
unsignedShort	unsigned short integer (0-65535)
unsignedByte	unsigned byte integer (0-255)
boolean	Boolean

dateTime	YYYY-MM-DDTHH:MM:SS date/time
time	HH:MM:SS time
NMTOKEN	String with no spaces
void	Internal use
IPAddr	IP address
IPNameAddr	IP address or name
IP4Addr	IPv4 address
IP6Addr	IPv6 address
IPPrefix	IP address / bitlen
IPRange	IP address / bitlen or range
IPNameRange	IP address / bitlen or range or name
IP4Range	IPv4 address / bitlen or range
IP4Prefix	IPv4 address / bitlen
IPSubnet	IP address / bitlen
IPFilter	Route filter
Password	Password
Community	xxx:xxx community
PortRange	xxx-xxx port range
Colour	#rgb #rrggbb #rgba #rrggbbaa colour
Secret	Secret/passphrase
duration	Period [[HH:]MM:]SS
username	Login name *(NMTOKEN)*
ipnamerangelist	List of IPranges or ip groups *(IPNameRange)*
routetable	Route table number (0-99) *(unsignedByte)*
ipnamelist	List of IP addresses or domain names *(IPNameAddr)*
datenum	Day number in month (1-31) *(unsignedByte)*
stringlist	List of strings *(string)*
iplist	List of IP addresses *(IPAddr)*
subnetlist	List of subnets *(IPSubnet)*
ra-max	Route announcement max interval (seconds) (4-1800) *(unsignedShort)*
ra-min	Route announcement min interval (seconds) (3-1350) *(unsignedShort)*
ip6list	List of IPv6 addresses *(IP6Addr)*
mtu	Max transmission unit (576-2000) *(unsignedShort)*
vlan	VLAN ID (0=untagged) (0-4095) *(unsignedShort)*
ip4rangelist	List of IP4ranges *(IP4Range)*
macprefixlist	List of strings *(macprefix)*
macprefix	MAC prefix *(hexBinary)*
ip4list	List of IPv4 addresses *(IP4Addr)*
graphname	Graph name *(token)*
prefixlist	List of IP Prefixes *(IPPrefix)*
ping-size	Data payload size to be sent in ping packet (0-1472) *(unsignedInt)*

iprangelist	List of IPranges *(IPRange)*
nmtokenlist	List of NMTOKEN *(NMTOKEN)*
userlist	List of user names *(username)*
prefix4list	List of IPv4 Prefixes *(IP4Prefix)*
filterlist	List of IP Prefix filters *(IPFilter)*
communitylist	List of BGP communities *(Community)*
portlist	List of protocol port ranges *(PortRange)*
protolist	List of IP protocols *(unsignedByte)*
unsignedIntList	List of integers *(unsignedInt)*
routetableset	Set of routetables *(routetable)*
aslist	List of AS numbers *(unsignedIntList)*
vlan-nz	VLAN ID (1-4095) *(unsignedShort)*
dates	Set of dates *(datenum)*
cug	CUG ID (1-32767) *(unsignedShort)*
tun-id	Local tunnel ID (1-20) *(unsignedShort)*
ses-id	Local session ID (1-250) *(unsignedShort)*

Index