

FireBrick FB6202

User Manual

 *FB6000 Versatile Network Appliance*

FireBrick FB6202 User Manual

This User Manual documents Software version V2.00.010
Copyright © 2012-2023 FireBrick Ltd.

Table of Contents

| | |
|--|----|
| Preface | xx |
| 1. Introduction | 1 |
| 1.1. The FB6000 | 1 |
| 1.1.1. Where do I start? | 1 |
| 1.1.2. What can it do? | 1 |
| 1.1.2.1. FB6202 Gigabit L2TP LNS with detailed monitoring of all lines | 2 |
| 1.1.3. Ethernet port capabilities | 2 |
| 1.1.4. Product variants in the FB6000 series | 2 |
| 1.2. About this Manual | 2 |
| 1.2.1. Version | 2 |
| 1.2.2. Intended audience | 3 |
| 1.2.3. Technical details | 3 |
| 1.2.4. Document style | 3 |
| 1.2.5. Document conventions | 3 |
| 1.2.6. Comments and feedback | 4 |
| 1.3. Additional Resources | 4 |
| 1.3.1. Technical Support | 4 |
| 1.3.2. IRC Channel | 4 |
| 1.3.3. Application Notes | 4 |
| 1.3.4. Training Courses | 5 |
| 2. Getting Started | 6 |
| 2.1. IP addressing | 6 |
| 2.2. Accessing the web-based user interface | 6 |
| 2.2.1. Initial configuration | 7 |
| 3. Configuration | 8 |
| 3.1. The Object Hierarchy | 8 |
| 3.2. The Object Model | 8 |
| 3.2.1. Formal definition of the object model | 9 |
| 3.2.2. Common attributes | 9 |
| 3.3. Configuration Methods | 9 |
| 3.4. Configuration upgrades and versioning | 9 |
| 3.5. Data types | 10 |
| 3.5.1. Sending and receiving values | 10 |
| 3.5.2. Lists of values | 10 |
| 3.5.3. Set of possible values | 10 |
| 3.5.4. Dates, times, and durations | 11 |
| 3.5.5. Colours | 11 |
| 3.5.6. Passwords and secrets | 11 |
| 3.5.7. IP addresses | 11 |
| 3.5.7.1. Simple IP addresses | 11 |
| 3.5.7.2. Subnets and prefixes | 12 |
| 3.5.7.3. Ranges | 12 |
| 3.5.7.4. Prefix filters | 12 |
| 3.6. Web User Interface Overview | 12 |
| 3.6.1. User Interface layout | 13 |
| 3.6.2. Config pages and the object hierarchy | 13 |
| 3.6.2.1. Configuration categories | 13 |
| 3.6.2.2. Object settings | 15 |
| 3.6.3. Navigating around the User Interface | 16 |
| 3.6.4. Backing up / restoring the configuration | 17 |
| 3.6.5. Customising the layout | 17 |
| 3.7. Configuration using XML | 17 |
| 3.7.1. Introduction to XML | 18 |
| 3.7.2. The root element - <config> | 19 |

| | |
|---|----|
| 3.7.3. Viewing or editing XML | 19 |
| 3.7.4. Example XML configuration | 19 |
| 3.8. Downloading/Uploading the configuration | 20 |
| 3.8.1. Download | 20 |
| 3.8.2. Upload | 21 |
| 4. System Administration | 22 |
| 4.1. User Management | 22 |
| 4.1.1. Login level | 22 |
| 4.1.2. Configuration access level | 23 |
| 4.1.3. Login idle timeout | 23 |
| 4.1.4. Restricting user logins | 24 |
| 4.1.4.1. Restrict by IP address | 24 |
| 4.1.4.2. Logged in IP address | 24 |
| 4.1.4.3. Restrict by profile | 24 |
| 4.1.5. Password change | 24 |
| 4.1.6. One Time Password (OTP) | 24 |
| 4.2. General System settings | 25 |
| 4.2.1. System name (hostname) | 25 |
| 4.2.2. Administrative details | 25 |
| 4.2.3. System-level event logging control | 25 |
| 4.2.4. Home page web links | 26 |
| 4.3. Software Upgrades | 26 |
| 4.3.1. Software release types | 26 |
| 4.3.1.1. Breakpoint releases | 27 |
| 4.3.2. Identifying current software version | 27 |
| 4.3.3. Internet-based upgrade process | 27 |
| 4.3.3.1. Manually initiating upgrades | 28 |
| 4.3.3.2. Controlling automatic software updates | 28 |
| 4.3.4. Manual upgrade | 29 |
| 4.4. Boot Process | 29 |
| 4.4.1. LED indications | 29 |
| 4.4.1.1. Port LEDs | 29 |
| 5. Event Logging | 30 |
| 5.1. Overview | 30 |
| 5.1.1. Log targets | 30 |
| 5.1.1.1. Logging to Flash memory | 30 |
| 5.1.1.2. Logging to the Console | 31 |
| 5.2. Enabling logging | 31 |
| 5.3. Logging to external destinations | 31 |
| 5.3.1. Syslog | 31 |
| 5.3.2. Email | 32 |
| 5.3.2.1. E-mail process logging | 33 |
| 5.4. Factory reset configuration log targets | 33 |
| 5.5. Performance | 33 |
| 5.6. Viewing logs | 33 |
| 5.6.1. Viewing logs in the User Interface | 33 |
| 5.6.2. Viewing logs in the CLI environment | 34 |
| 5.7. System-event logging | 34 |
| 5.8. Using Profiles | 34 |
| 6. Interfaces and Subnets | 35 |
| 6.1. Relationship between Interfaces and Physical Ports | 35 |
| 6.1.1. Port groups | 35 |
| 6.1.2. Interfaces | 35 |
| 6.2. Defining an interface | 36 |
| 6.2.1. Defining subnets | 36 |
| 6.2.1.1. Source filtering | 37 |

| | |
|---|----|
| 6.2.1.2. Using DHCP to configure a subnet | 37 |
| 6.2.1.3. Using SLAAC (IPv6 router announcements) to configure a subnet | 38 |
| 6.2.1.4. Providing IPv6 addresses to devices on a network (IPv6 router announcements) | 38 |
| 6.2.1.5. IPv6 prefix delegated subnets | 38 |
| 6.2.2. Setting up DHCP server parameters | 38 |
| 6.2.2.1. Fixed/Static DHCP allocations | 39 |
| 6.2.2.2. Restricted allocations | 40 |
| 6.2.2.3. Special DHCP options | 41 |
| 6.2.2.4. Logging | 41 |
| 6.2.3. DHCP Relay Agent | 41 |
| 6.3. Physical port settings | 41 |
| 6.3.1. Setting duplex mode | 42 |
| 6.3.2. Defining port LED functions | 42 |
| 7. Routing | 43 |
| 7.1. Routing logic | 43 |
| 7.2. Routing targets | 44 |
| 7.2.1. Subnet routes | 44 |
| 7.2.2. Routing to an IP address (gateway route) | 44 |
| 7.2.3. Special targets | 45 |
| 7.3. Dynamic route creation / deletion | 45 |
| 7.4. Routing tables | 45 |
| 7.5. Bonding | 45 |
| 8. Profiles | 47 |
| 8.1. Overview | 47 |
| 8.2. Creating/editing profiles | 47 |
| 8.2.1. Timing control | 47 |
| 8.2.2. Tests | 48 |
| 8.2.2.1. General tests | 48 |
| 8.2.2.2. Time/date tests | 48 |
| 8.2.2.3. Ping tests | 48 |
| 8.2.3. Inverting overall test result | 49 |
| 8.2.4. Manual override | 49 |
| 8.2.5. Scripting | 49 |
| 9. Traffic Shaping | 51 |
| 9.1. Graphs and Shapers | 51 |
| 9.1.1. Graphs | 51 |
| 9.1.2. Shapers | 52 |
| 9.1.3. Ad hoc shapers | 52 |
| 9.1.4. Long term shapers | 52 |
| 9.1.5. Shared shapers | 53 |
| 9.2. Multiple shapers | 53 |
| 9.3. Basic principles | 53 |
| 10. PPPoE | 54 |
| 10.1. PPPoE client | 54 |
| 10.2. Types of DSL line and router in the United Kingdom | 54 |
| 10.3. Defining PPPoE client links | 55 |
| 10.3.1. IPv6 | 55 |
| 10.3.2. Additional options | 56 |
| 10.3.2.1. MTU and TCP fix | 56 |
| 10.3.2.2. Service and ac-name | 56 |
| 10.3.2.3. Logging | 56 |
| 10.3.2.4. Speed and graphs | 56 |
| 10.4. PPPoE BRAS | 57 |
| 10.4.1. Additional options | 57 |
| 11. Tunnels | 59 |

| | |
|--|----|
| 11.1. L2TP tunnelling | 59 |
| 11.1.1. Incoming tunnel | 59 |
| 11.1.2. Incoming session | 60 |
| 12. System Services | 61 |
| 12.1. Protecting the FB6000 | 61 |
| 12.2. Common settings | 61 |
| 12.3. HTTP Server configuration | 62 |
| 12.3.1. Access control | 62 |
| 12.3.1.1. Trusted addresses | 63 |
| 12.3.2. HTTPS access | 63 |
| 12.4. Telnet Server configuration | 64 |
| 12.4.1. Access control | 64 |
| 12.5. DNS configuration | 64 |
| 12.5.1. Auto DHCP DNS | 65 |
| 12.5.2. Local DNS responses | 65 |
| 12.5.3. Blocking DNS names | 65 |
| 12.6. NTP configuration | 66 |
| 12.7. SNMP configuration | 66 |
| 12.8. RADIUS configuration | 66 |
| 12.8.1. RADIUS server (platform RADIUS) | 66 |
| 12.8.2. RADIUS client | 66 |
| 12.8.2.1. RADIUS client settings | 66 |
| 12.8.2.2. Server blacklisting | 67 |
| 13. Network Diagnostic Tools | 68 |
| 13.1. Access check | 68 |
| 13.2. Packet Dumping | 69 |
| 13.2.1. Dump parameters | 69 |
| 13.2.2. Security settings required | 70 |
| 13.2.3. IP address matching | 70 |
| 13.2.4. Packet types | 70 |
| 13.2.5. Snaplen specification | 71 |
| 13.2.6. Using the web interface | 71 |
| 13.2.7. Using an HTTP client | 71 |
| 13.2.7.1. Example using curl and tcpdump | 71 |
| 14. VRRP | 73 |
| 14.1. Virtual Routers | 73 |
| 14.2. Configuring VRRP | 74 |
| 14.2.1. Advertisement Interval | 74 |
| 14.2.2. Priority | 74 |
| 14.3. Using a virtual router | 74 |
| 14.4. VRRP versions | 74 |
| 14.4.1. VRRP version 2 | 74 |
| 14.4.2. VRRP version 3 | 75 |
| 14.5. Compatibility | 75 |
| 15. BGP | 76 |
| 15.1. What is BGP? | 76 |
| 15.2. BGP Setup | 76 |
| 15.2.1. Overview | 76 |
| 15.2.2. Standards | 76 |
| 15.2.3. Simple example setup | 77 |
| 15.2.4. Peer type | 77 |
| 15.2.5. Route filtering | 78 |
| 15.2.5.1. Matching attributes | 78 |
| 15.2.5.2. Action attributes | 78 |
| 15.2.6. Well known community tags | 79 |
| 15.2.7. Announcing black hole routes | 79 |

| | |
|--|----|
| 15.2.8. Grey holes | 80 |
| 15.2.9. Announcing dead end routes | 80 |
| 15.2.10. Bad optional path attributes | 80 |
| 15.2.11. <network> element | 80 |
| 15.2.12. <route>, <subnet> and other elements | 81 |
| 15.2.13. Route feasibility testing | 81 |
| 15.2.14. Status | 81 |
| 15.2.15. Diagnostics | 81 |
| 15.2.16. Router startup and shutdown | 82 |
| 15.2.17. TTL security | 82 |
| 16. Internet Service Providers | 83 |
| 16.1. Background | 83 |
| 16.1.1. How it all began | 83 |
| 16.1.2. Point to Point Protocol | 83 |
| 16.1.3. L2TP | 83 |
| 16.1.4. Broadband | 84 |
| 16.1.5. RADIUS | 84 |
| 16.1.6. BGP | 84 |
| 16.2. Incoming L2TP connections | 84 |
| 16.3. The importance of CQM graphs | 85 |
| 16.4. Authentication | 85 |
| 16.5. Accounting | 86 |
| 16.6. RADIUS Control messages | 86 |
| 16.7. PPPoE | 86 |
| 16.8. Typical configuration | 86 |
| 16.8.1. Interlink subnet | 86 |
| 16.8.2. BGP with carrier | 86 |
| 16.8.3. RADIUS session steering | 87 |
| 16.8.4. L2TP endpoints | 87 |
| 16.8.5. ISP RADIUS | 88 |
| 17. Command Line Interface | 89 |
| A. CIDR and CIDR Notation | 90 |
| B. MAC Addresses usage | 92 |
| B.1. Multiple MAC addresses? | 92 |
| B.2. How the FireBrick allocates MAC addresses | 93 |
| B.2.1. Interface | 93 |
| B.2.2. Subnet | 93 |
| B.2.3. PPPoE | 93 |
| B.2.4. Running out of MACs | 93 |
| B.3. Forcing particular MAC addresses | 94 |
| B.4. MAC address on label | 94 |
| B.5. Using with a DHCP server | 94 |
| C. Scripted access | 96 |
| C.1. Tools | 96 |
| C.2. Access control | 96 |
| C.2.1. Username and password | 96 |
| C.2.2. OTP | 96 |
| C.2.3. Allow list | 96 |
| C.2.4. Allowed access | 96 |
| C.3. XML data for common functions | 96 |
| C.4. XML data from diagnostics and tests | 97 |
| C.4.1. Cross site scripting security | 97 |
| C.4.2. Arguments to scripts | 97 |
| C.5. Special URLs | 98 |
| C.6. Web sockets | 98 |
| D. VLANs : A primer | 99 |

| | |
|--|-----|
| E. Supported L2TP Attribute/Value Pairs | 100 |
| E.1. Start-Control-Connection-Request | 100 |
| E.2. Start-Control-Connection-Reply | 100 |
| E.3. Start-Control-Connection-Connected | 101 |
| E.4. Stop-Control-Connection-Notification | 101 |
| E.5. Hello | 101 |
| E.6. Incoming-Call-Request | 101 |
| E.7. Incoming-Call-Reply | 102 |
| E.8. Incoming-Call-Connected | 102 |
| E.9. Outgoing-Call-Request | 103 |
| E.10. Outgoing-Call-Reply | 103 |
| E.11. Outgoing-Call-Connected | 103 |
| E.12. Call-Disconnect-Notify | 103 |
| E.13. WAN-Error-Notify | 103 |
| E.14. Set-Link-Info | 104 |
| E.15. Notes | 104 |
| E.15.1. BT specific notes | 104 |
| E.15.2. IP over LCP | 104 |
| F. Supported RADIUS Attribute/Value Pairs for L2TP operation | 105 |
| F.1. Authentication request | 105 |
| F.2. Authentication response | 106 |
| F.2.1. Accepted authentication | 106 |
| F.2.1.1. Prefix Delegation | 107 |
| F.2.2. Rejected authentication | 108 |
| F.3. Accounting Start | 108 |
| F.4. Accounting Interim | 109 |
| F.5. Accounting Stop | 110 |
| F.6. Disconnect | 110 |
| F.7. Change of Authorisation | 111 |
| F.8. Filter ID | 111 |
| F.9. Notes | 113 |
| F.9.1. L2TP relay | 113 |
| F.9.2. LCP echo and CQM graphs | 113 |
| F.9.3. IP over LCP | 114 |
| F.9.4. Closed User Group | 114 |
| F.9.5. Routing table | 114 |
| G. FireBrick specific SNMP objects | 115 |
| G.1. Conventions | 115 |
| G.1.1. IP addresses as indices | 115 |
| G.2. Firebrick-specific structures for BGP | 115 |
| G.2.1. Structure definitions | 115 |
| G.2.1.1. The list of BGP peers for this Firebrick | 115 |
| G.2.2. Enum Definitions | 116 |
| G.3. Firebrick-specific structures for L2TP | 117 |
| G.3.1. Structure definitions | 117 |
| G.3.1.1. fbL2tpGeneralTunnels | 117 |
| G.3.1.2. fbL2tpGeneralSessions | 117 |
| G.3.1.3. The list of L2TP peers for this Firebrick | 117 |
| G.4. Firebrick CPU usage | 118 |
| G.4.1. Structure definitions | 118 |
| G.4.1.1. CPU usage for this Firebrick | 118 |
| G.5. Firebrick system stats | 119 |
| G.5.1. Structure definitions | 119 |
| G.5.1.1. The table of runtime stats for this Firebrick | 119 |
| G.6. Monitoring for general system features | 119 |
| G.6.1. Structure definitions | 119 |

| | |
|--|-----|
| G.6.1.1. The list of readings for this Firebrick | 119 |
| G.7. System wide status | 120 |
| G.7.1. Structure definitions | 120 |
| G.7.1.1. fbGlobalMemory | 120 |
| G.7.1.2. fbGlobalBuffers | 120 |
| G.8. Monitoring information (deprecated) | 120 |
| H. Command line reference | 122 |
| H.1. General commands | 122 |
| H.1.1. Trace off | 122 |
| H.1.2. Trace on | 122 |
| H.1.3. Uptime | 122 |
| H.1.4. General status | 122 |
| H.1.5. Memory usage | 122 |
| H.1.6. Process/task usage | 122 |
| H.1.7. Login | 122 |
| H.1.8. Logout | 123 |
| H.1.9. See XML configuration | 123 |
| H.1.10. Load XML configuration | 123 |
| H.1.11. Show profile status | 123 |
| H.1.12. Enable profile control switch | 123 |
| H.1.13. Disable profile control switch | 123 |
| H.1.14. Show RADIUS servers | 123 |
| H.1.15. Show DNS resolvers | 123 |
| H.2. Networking commands | 124 |
| H.2.1. Subnets | 124 |
| H.2.2. Renegotiate DHCP for a subnet | 124 |
| H.2.3. Ping and trace | 124 |
| H.2.4. Show a route from the routing table | 124 |
| H.2.5. List routes | 125 |
| H.2.6. List routing next hops | 125 |
| H.2.7. See DHCP allocations | 125 |
| H.2.8. Clear DHCP allocations | 125 |
| H.2.9. Lock DHCP allocations | 125 |
| H.2.10. Unlock DHCP allocations | 125 |
| H.2.11. Name DHCP allocations | 125 |
| H.2.12. Show ARP/ND status | 125 |
| H.2.13. Show VRRP status | 125 |
| H.2.14. Send Wake-on-LAN packet | 126 |
| H.2.15. Check access to services | 126 |
| H.3. BGP commands | 126 |
| H.3.1. Show BGP | 126 |
| H.3.2. Show BGP Peer | 126 |
| H.3.3. Show BGP Summary | 126 |
| H.3.4. Show BGP Routes | 126 |
| H.3.5. Clear BGP | 126 |
| H.3.6. Refresh BGP | 127 |
| H.3.7. Refresh BGP | 127 |
| H.4. L2TP commands | 127 |
| H.4.1. Show L2TP | 127 |
| H.4.2. Show L2TP Tunnels | 127 |
| H.4.3. Clear L2TP All | 127 |
| H.4.4. Show L2TP Tunnel | 127 |
| H.4.5. Show L2TP Tunnel | 128 |
| H.4.6. Show L2TP Sessions | 128 |
| H.4.7. Show L2TP Session | 128 |
| H.4.8. Clear L2TP Tunnel | 128 |

| | |
|--|-----|
| H.4.9. Clear L2TP Tunnel | 128 |
| H.4.10. Clear L2TP Session | 128 |
| H.5. Advanced commands | 128 |
| H.5.1. Panic | 129 |
| H.5.2. Reboot | 129 |
| H.5.3. Screen width | 129 |
| H.5.4. Make outbound command session | 129 |
| H.5.5. Show command sessions | 129 |
| H.5.6. Kill command session | 129 |
| H.5.7. Flash memory list | 129 |
| H.5.8. Delete block from flash | 130 |
| H.5.9. Boot log | 130 |
| H.5.10. Flash log | 130 |
| I. Constant Quality Monitoring - technical details | 131 |
| I.1. Broadband back-haul providers | 131 |
| I.2. Access to graphs and csvs | 131 |
| I.2.1. Trusted access | 131 |
| I.2.2. Dated information | 132 |
| I.2.3. Authenticated access | 132 |
| I.3. Graph display options | 132 |
| I.3.1. Scaleable Vector Graphics | 132 |
| I.3.2. Data points | 133 |
| I.3.3. Additional text | 133 |
| I.3.4. Other colours and spacing | 133 |
| I.4. Overnight archiving | 134 |
| I.4.1. Full URL format | 134 |
| I.4.2. load handling | 135 |
| I.5. Graph scores | 135 |
| I.6. Creating graphs, and graph names | 135 |
| J. Hashed passwords | 137 |
| J.1. Password hashing | 137 |
| J.1.1. Salt | 137 |
| J.2. One Time Password seed hashing | 138 |
| K. Configuration Objects | 140 |
| K.1. Top level | 140 |
| K.1.1. config: Top level config | 140 |
| K.2. Objects | 141 |
| K.2.1. system: System settings | 141 |
| K.2.2. link: Web links | 142 |
| K.2.3. routing-table: Default source IP for services using a given table | 143 |
| K.2.4. user: Admin users | 143 |
| K.2.5. eap: User access controlled by EAP | 143 |
| K.2.6. log: Log target controls | 144 |
| K.2.7. log-syslog: Syslog logger settings | 144 |
| K.2.8. log-email: Email logger settings | 145 |
| K.2.9. services: System services | 145 |
| K.2.10. http-service: Web service settings | 146 |
| K.2.11. dns-service: DNS service settings | 147 |
| K.2.12. dns-host: Fixed local DNS host settings | 148 |
| K.2.13. dns-block: Fixed local DNS blocks | 148 |
| K.2.14. radius-service: RADIUS service definition | 148 |
| K.2.15. radius-service-match: Matching rules for RADIUS service | 150 |
| K.2.16. radius-server: RADIUS server settings | 152 |
| K.2.17. telnet-service: Telnet service settings | 152 |
| K.2.18. snmp-service: SNMP service settings | 153 |
| K.2.19. time-service: System time server settings | 153 |

| | |
|--|-----|
| K.2.20. ethernet: Physical port controls | 154 |
| K.2.21. sampling: Packet sampling configuration | 155 |
| K.2.22. portdef: Port grouping and naming | 155 |
| K.2.23. interface: Port-group/VLAN interface settings | 156 |
| K.2.24. subnet: Subnet settings | 157 |
| K.2.25. subnet-template: Subnet option templates for RA | 158 |
| K.2.26. vrrp: VRRP settings | 158 |
| K.2.27. dhcps: DHCP server settings | 159 |
| K.2.28. dhcp-attr-hex: DHCP server attributes (hex) | 160 |
| K.2.29. dhcp-attr-string: DHCP server attributes (string) | 160 |
| K.2.30. dhcp-attr-number: DHCP server attributes (numeric) | 161 |
| K.2.31. dhcp-attr-ip: DHCP server attributes (IP) | 161 |
| K.2.32. pppoe: PPPoE settings | 161 |
| K.2.33. ppp-route: PPP routes | 163 |
| K.2.34. route: Static routes | 163 |
| K.2.35. network: Locally originated networks | 164 |
| K.2.36. blackhole: Dead end networks | 164 |
| K.2.37. loopback: Locally originated networks | 165 |
| K.2.38. namedbgpmap: Mapping and filtering rules of BGP prefixes | 165 |
| K.2.39. bgprule: Individual mapping/filtering rule | 165 |
| K.2.40. bgp: Overall BGP settings | 166 |
| K.2.41. bgppeer: BGP peer definitions | 166 |
| K.2.42. bgpmap: Mapping and filtering rules of BGP prefixes | 168 |
| K.2.43. cqm: Constant Quality Monitoring settings | 169 |
| K.2.44. l2tp: L2TP settings | 171 |
| K.2.45. l2tp-incoming: L2TP settings for incoming L2TP connections | 171 |
| K.2.46. l2tp-relay: Relay and local authentication rules for L2TP | 173 |
| K.2.47. profile: Control profile | 174 |
| K.2.48. profile-date: Test passes if within any of the time ranges specified | 175 |
| K.2.49. profile-time: Test passes if within any of the date/time ranges specified | 175 |
| K.2.50. profile-ping: Test passes if any addresses are pingable | 175 |
| K.2.51. shaper: Traffic shaper | 176 |
| K.2.52. shaper-override: Traffic shaper override based on profile | 176 |
| K.2.53. ip-group: IP Group | 177 |
| K.2.54. dhcp-relay: DHCP server settings for remote / relayed requests | 177 |
| K.3. Data types | 178 |
| K.3.1. user-level: User login level | 178 |
| K.3.2. ppp-dump: PPP dump format | 178 |
| K.3.3. autoloadtype: Type of s/w auto load | 178 |
| K.3.4. config-access: Type of access user has to config | 179 |
| K.3.5. eap-subsystem: Subsystem with EAP access control | 179 |
| K.3.6. eap-method: EAP access method | 179 |
| K.3.7. syslog-severity: Syslog severity | 179 |
| K.3.8. syslog-facility: Syslog facility | 180 |
| K.3.9. http-mode: HTTP/HTTPS security mode | 180 |
| K.3.10. radiuspriority: Options for controlling platform RADIUS response priority tagging | 181 |
| K.3.11. radiustype: Type of RADIUS server | 181 |
| K.3.12. month: Month name (3 letter) | 181 |
| K.3.13. day: Day name (3 letter) | 182 |
| K.3.14. port: Physical port | 182 |
| K.3.15. Crossover: Crossover configuration | 182 |
| K.3.16. LinkFlow: Physical port flow control setting | 182 |
| K.3.17. LinkClock: Physical port Gigabit clock master/slave setting | 182 |
| K.3.18. LinkLED-y: Yellow LED setting | 183 |
| K.3.19. LinkLED-g: Green LED setting | 183 |

| | |
|--|-----|
| K.3.20. LinkPower: PHY power saving options | 183 |
| K.3.21. LinkFault: Link fault type to send | 183 |
| K.3.22. sampling-protocol: Sampling protocol | 184 |
| K.3.23. trunk-mode: Trunk port mode | 184 |
| K.3.24. ramode: IPv6 route announce level | 184 |
| K.3.25. bgpmode: BGP announcement mode | 184 |
| K.3.26. sampling-mode: Sampling mode | 185 |
| K.3.27. sfoption: Source filter option | 185 |
| K.3.28. pppoe-mode: Type of PPPoE connection | 185 |
| K.3.29. pppoe-calling: Additional prefix on PPPoE calling ID | 185 |
| K.3.30. peertype: BGP peer type | 185 |
| K.3.31. radius-nas: NAS IP to report | 186 |
| K.3.32. switch: Profile manual setting | 186 |
| K.4. Basic types | 186 |
| Index | 189 |

List of Figures

| | |
|--|----|
| 2.1. Initial web page in factory reset state | 7 |
| 3.1. Icons for configuration categories | 14 |
| 3.2. The "Setup" category | 14 |
| 3.3. Editing an "Interface" object | 15 |
| 3.4. Show hidden attributes | 15 |
| 3.5. Attribute definitions | 16 |
| 3.6. Navigation controls | 16 |
| 4.1. Setting up a new user | 22 |
| 4.2. Software upgrade available notification | 28 |
| 4.3. Manual Software upload | 29 |
| B.1. Product label showing MAC address range | 94 |

List of Tables

| | |
|--|-----|
| 2.1. IP addresses for computer | 6 |
| 2.2. IP addresses to access the FireBrick | 6 |
| 2.3. IP addresses to access the FireBrick | 6 |
| 3.1. Special character sequences | 18 |
| 4.1. User login levels | 23 |
| 4.2. Configuration access levels | 23 |
| 4.3. General administrative details attributes | 25 |
| 4.4. Attributes controlling auto-upgrades | 28 |
| 5.1. Logging attributes | 31 |
| 5.2. System-Event Logging attributes | 34 |
| 7.1. Example route targets | 44 |
| 12.1. List of system services | 61 |
| 12.2. List of system services | 62 |
| 13.1. Packet dump parameters | 69 |
| 13.2. Packet types that can be captured | 70 |
| 15.1. Peer types | 77 |
| 15.2. Communities | 79 |
| 15.3. Network attributes | 81 |
| B.1. DHCP client names used | 95 |
| C.1. Special URLs | 98 |
| E.1. SCCRQ | 100 |
| E.2. SCCRP | 100 |
| E.3. SCCCN | 101 |
| E.4. StopCCN | 101 |
| E.5. HELLO | 101 |
| E.6. ICRQ | 101 |
| E.7. ICRP | 102 |
| E.8. ICCN | 102 |
| E.9. OCRQ | 103 |
| E.10. OCRP | 103 |
| E.11. OCCN | 103 |
| E.12. CDN | 103 |
| E.13. WEN | 103 |
| E.14. SLI | 104 |
| F.1. Access-request | 105 |
| F.2. Access-Accept | 106 |
| F.3. Access-Reject | 108 |
| F.4. Accounting-Start | 108 |
| F.5. Accounting-Interim | 109 |
| F.6. Accounting-Stop | 110 |
| F.7. Disconnect | 110 |
| F.8. Change-of-Authorisation | 111 |
| F.9. Filter-ID | 112 |
| G.1. Indices | 115 |
| G.2. Fields | 115 |
| G.3. FbBgpPeerState - The state of a BGP peer | 116 |
| G.4. Fields | 117 |
| G.5. Fields | 117 |
| G.6. Indices | 117 |
| G.7. Fields | 118 |
| G.8. Indices | 118 |
| G.9. Fields | 118 |
| G.10. Indices | 119 |
| G.11. Fields | 119 |

| | |
|--|-----|
| G.12. Indices | 119 |
| G.13. Fields | 120 |
| G.14. Fields | 120 |
| G.15. Fields | 120 |
| G.16. iso.3.6.1.4.1.24693.1.X.Y | 120 |
| I.1. File types | 131 |
| I.2. Colours | 133 |
| I.3. Text | 133 |
| I.4. Text | 133 |
| I.5. URL formats | 134 |
| K.1. config: Attributes | 140 |
| K.2. config: Elements | 140 |
| K.3. system: Attributes | 141 |
| K.4. system: Elements | 142 |
| K.5. link: Attributes | 142 |
| K.6. routing-table: Attributes | 143 |
| K.7. user: Attributes | 143 |
| K.8. eap: Attributes | 143 |
| K.9. log: Attributes | 144 |
| K.10. log: Elements | 144 |
| K.11. log-syslog: Attributes | 144 |
| K.12. log-email: Attributes | 145 |
| K.13. services: Elements | 145 |
| K.14. http-service: Attributes | 146 |
| K.15. dns-service: Attributes | 147 |
| K.16. dns-service: Elements | 148 |
| K.17. dns-host: Attributes | 148 |
| K.18. dns-block: Attributes | 148 |
| K.19. radius-service: Attributes | 149 |
| K.20. radius-service: Elements | 150 |
| K.21. radius-service-match: Attributes | 150 |
| K.22. radius-server: Attributes | 152 |
| K.23. telnet-service: Attributes | 152 |
| K.24. snmp-service: Attributes | 153 |
| K.25. time-service: Attributes | 153 |
| K.26. ethernet: Attributes | 154 |
| K.27. sampling: Attributes | 155 |
| K.28. portdef: Attributes | 155 |
| K.29. interface: Attributes | 156 |
| K.30. interface: Elements | 157 |
| K.31. subnet: Attributes | 157 |
| K.32. subnet-template: Attributes | 158 |
| K.33. vrrp: Attributes | 158 |
| K.34. dhcps: Attributes | 159 |
| K.35. dhcps: Elements | 160 |
| K.36. dhcp-attr-hex: Attributes | 160 |
| K.37. dhcp-attr-string: Attributes | 160 |
| K.38. dhcp-attr-number: Attributes | 161 |
| K.39. dhcp-attr-ip: Attributes | 161 |
| K.40. pppoe: Attributes | 161 |
| K.41. pppoe: Elements | 163 |
| K.42. ppp-route: Attributes | 163 |
| K.43. route: Attributes | 163 |
| K.44. network: Attributes | 164 |
| K.45. blackhole: Attributes | 164 |
| K.46. loopback: Attributes | 165 |

| | |
|---|-----|
| K.47. namedbgpmap: Attributes | 165 |
| K.48. namedbgpmap: Elements | 165 |
| K.49. bgprule: Attributes | 165 |
| K.50. bgp: Attributes | 166 |
| K.51. bgp: Elements | 166 |
| K.52. bgppeer: Attributes | 167 |
| K.53. bgppeer: Elements | 168 |
| K.54. bgpmap: Attributes | 168 |
| K.55. bgpmap: Elements | 169 |
| K.56. cqm: Attributes | 169 |
| K.57. l2tp: Attributes | 171 |
| K.58. l2tp: Elements | 171 |
| K.59. l2tp-incoming: Attributes | 171 |
| K.60. l2tp-incoming: Elements | 173 |
| K.61. l2tp-relay: Attributes | 173 |
| K.62. profile: Attributes | 174 |
| K.63. profile: Elements | 175 |
| K.64. profile-date: Attributes | 175 |
| K.65. profile-time: Attributes | 175 |
| K.66. profile-ping: Attributes | 175 |
| K.67. shaper: Attributes | 176 |
| K.68. shaper: Elements | 176 |
| K.69. shaper-override: Attributes | 177 |
| K.70. ip-group: Attributes | 177 |
| K.71. dhcp-relay: Attributes | 177 |
| K.72. dhcp-relay: Elements | 178 |
| K.73. user-level: User login level | 178 |
| K.74. ppp-dump: PPP dump format | 178 |
| K.75. autoloadtype: Type of s/w auto load | 178 |
| K.76. config-access: Type of access user has to config | 179 |
| K.77. eap-subsystem: Subsystem with EAP access control | 179 |
| K.78. eap-method: EAP access method | 179 |
| K.79. syslog-severity: Syslog severity | 179 |
| K.80. syslog-facility: Syslog facility | 180 |
| K.81. http-mode: HTTP/HTTPS security mode | 180 |
| K.82. radiuspriority: Options for controlling platform RADIUS response priority tagging | 181 |
| K.83. radiustype: Type of RADIUS server | 181 |
| K.84. month: Month name (3 letter) | 181 |
| K.85. day: Day name (3 letter) | 182 |
| K.86. port: Physical port | 182 |
| K.87. Crossover: Crossover configuration | 182 |
| K.88. LinkFlow: Physical port flow control setting | 182 |
| K.89. LinkClock: Physical port Gigabit clock master/slave setting | 182 |
| K.90. LinkLED-y: Yellow LED setting | 183 |
| K.91. LinkLED-g: Green LED setting | 183 |
| K.92. LinkPower: PHY power saving options | 183 |
| K.93. LinkFault: Link fault type to send | 183 |
| K.94. sampling-protocol: Sampling protocol | 184 |
| K.95. trunk-mode: Trunk port mode | 184 |
| K.96. ramode: IPv6 route announce level | 184 |
| K.97. bgpmode: BGP announcement mode | 184 |
| K.98. sampling-mode: Sampling mode | 185 |
| K.99. sfoption: Source filter option | 185 |
| K.100. pppoe-mode: Type of PPPoE connection | 185 |
| K.101. pppoe-calling: Additional prefix on PPPoE calling ID | 185 |
| K.102. peertype: BGP peer type | 185 |

| | |
|---|-----|
| K.103. radius-nas: NAS IP to report | 186 |
| K.104. switch: Profile manual setting | 186 |
| K.105. Basic data types | 186 |

List of Examples

| | |
|-----------|-----|
| G.1. | 115 |
| G.2. | 115 |

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 bespoke 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 V2.00.010.

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 setup 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 : <https://www.firebrick.co.uk/support/manuals/>

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

The FB600 series are provided in a number of variants. This manual is for the FB6202. This variant includes:

- Layer 2 Tunnelling Protocol (L2TP) to terminate PPP connections (e.g. broadband lines)
- Border Gateway Protocol, to allow routes to be announced and accepted from peering BGP routers.

1.1.2.1. FB6202 Gigabit L2TP LNS with detailed monitoring of all lines

The FB6202 provides the key component for an Internet Service Provider (ISP) to terminate data connections from broadband or dial-up or similar service carriers that use L2TP. RADIUS is used to authenticate and account for data, and BGP is used to announce connected routes.

The FB6202 has unique features that provide invaluable levels of monitoring of connected circuits. There is also support for per-packet line bonding, IPv6 (including DHCPv6 and RA over PPP), and L2TP relay.

With the FB6202 you can handle thousands of broadband connections from a variety of carriers up to a gigabit throughput. You can pool multiple FB6102's to provide to handle much higher capacity even where links to carriers need aggregate traffic shaping.

1.1.3. Ethernet port capabilities

The FB6000 has two copper (RJ45) 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.

The two ports on the FB6000 can be combined as a single 2Gb/2 LACP bundle, with a choice of hashing logic for traffic distribution.

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
- **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 V2.00.010 - 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 [<https://www.firebrick.co.uk/support/software/>]. 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 appendices. These are intended to be a reference guide 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 significance) :-

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 details are :-

- IRC server: `irc.aachat.net`
- Port: 6697
- TLS: Required
- Channel: `#FireBrick`

1.3.3. Application Notes

Some applications notes have been created by the FireBrick team and included on the web site. There are also useful Wiki web sites provided by main dealers which cover specific configuration examples. Ask your dealer for more details. These are usually public Wiki web sites even if not buying from that specific dealer.

1.3.4. Training Courses

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

Training course attendance is mandatory for all FireBrick dealers to be accredited.

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 assigned 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

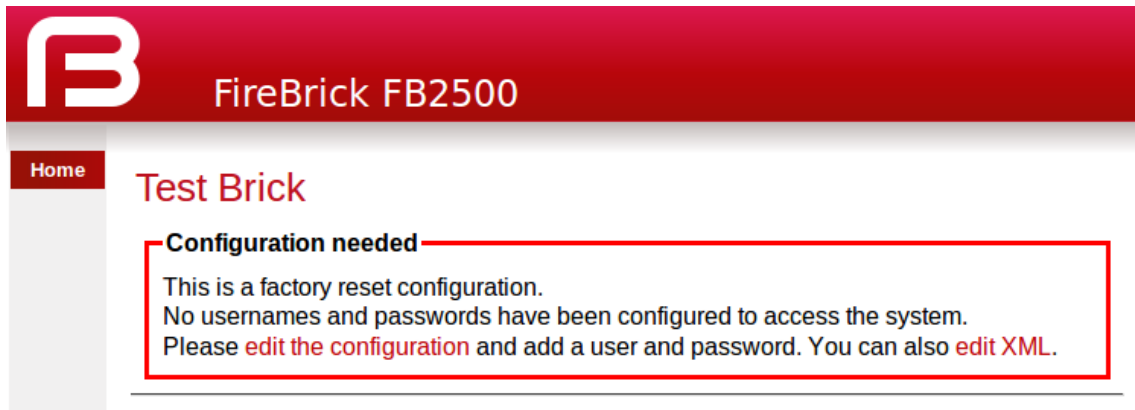
| IPv6 | IPv4 |
|---|---|
| http://[2001:DB8::1] | http://10.0.0.1 |

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



2.2.1. Initial configuration

Click on the "edit the configuration" link (red text) to perform the initial configuration.

You will then find yourself in the normal configuration editor where you can make changes to the default configuration to suit your needs.

You will need to create at least one user as once saved you are prompted to login using the username/password details you provided.

Note

If you have changed the LAN IP address settings and are using DHCP on your connected computer you may find you need to get a new address and re-connect to the FireBrick control pages at this point.

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 hierarchy - 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 K.

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 mandatory and often needs to be unique within the list of objects. 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 avoid 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 standard web-browser (uses javascript).
- 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. Which method you use is entirely up to you, and some users prefer one or the other, or may make some changes in one of the other. Some operations, such as changing the order of a list of objects, is easier in the XML editor, for example. The web based editor means you do not have to find/remember the attribute names as they are all presented to you. For more information on using XML, refer to Section 3.7.

3.4. Configuration upgrades and versioning

Whilst we aim to minimise the need for changes to existing configurations, occasionally a change to the object model will need to be made (usually in the interests of flexibility and clarity) that results in changes to the way options are expressed.

During a software upgrade that requires changes to be made the current config will be automatically transformed so that things keep working as they were before wherever possible.

We need to keep track of transformations that are not idempotent (that is applicable repeatedly without side effects). The `patch` attribute notes the config version to avoid those transformations being applied repeatedly.

Configs downloaded from the FireBrick will have the `patch` attribute, so that if they are stored and uploaded to a later version they will be correctly transformed. However the transformations aren't kept around forever, and are periodically cleared out in breakpoint releases (see Section 4.3.1.1 for more info).

If you wish to upload a config targetted for the current version without transformation simply omit the `patch` attribute.

The `FB{PRODUCT}` will warn during the upload of configurations with a `patch` version higher than that supported by the software.

3.5. Data types

All of the actual values in the configuration are provided by means of XML attributes, and so they are represented as a string of characters. The value is escaped as per XML rules, e.g. `&` for `&`, `<` for `<`, `>` for `>` and `"` for `"` within the string between the quote marks for the attribute value.

Obviously, even though all data is just a string, there are actually different data types, as defined in Section K.4, some of which have value restrictions (range of numbers, or specific string lengths, etc). Some of these are described in more details here.

3.5.1. Sending and receiving values

When you send the FireBrick a configuration the value is parsed and stored internally in a binary format. This means that when you access the config later it is possible the value you see is a *normalised* version of the value. For example storing a number as `000123` will return as `123`.

In some cases you can enter a value in a format you will never see come back when you view the config. For example, simple integers can have SI magnitude suffixes, e.g. `G` (giga), `M` (mega), or `k` (kilo), so you can enter `1.5k` but will see `1500`. You can also use `Gi` (gibi), `Mi` (mebi), or `Ki` (kibi) for IEC units based on powers of two. You can also suffix `B` to mean bytes and the resulting value stored will be 8 times larger (as integers are used for bits/second speed settings). Other examples include using colour names like `red` which you see as `#ff0000`.

3.5.2. Lists of values

Where the type is in fact a *list* of a type, the actual value in the XML attribute is actually a space separated list. This is consistent with the XSD (XML Schema Definition). In the web based editor such lists are automatically split on to separate lines to make it easier to read and edit, but in the raw XML the list simply uses a single space between each item.

For example `allow="192.168.0.0/16 10.0.0.0/8 172.16.0.0/12"` lists three IP prefixes to allow.

3.5.3. Set of possible values

Some types are simply a set of acceptable values, the simplest of which is *Boolean* allowing `false` and `true`.

The simple lists of values are detailed in Section K.2, with the acceptable values.

However, in some cases a data type is a reference to some other object, in which case the acceptable values are the *name* of those referenced objects. In most cases the web based config can ensure it provides a pull down menu of the acceptable values.

There is one special case for *IP groups* where the value can be an IP address, or range, or the name of an IP group. In this case the web config editor expects you to correctly type the name of an IP group.

There is one other special case of *graph names* where you can typically enter any name, including one of the defined *shaper* names or make up a new name as you wish with no error when you save the config. Graphs are created *on the fly* and so you have to be careful to correctly type graph names in the config to get the desired effect.

3.5.4. Dates, times, and durations

As per normal XML Schema rules, dates and times are in ISO8601 format, e.g. 2023-04-28 or 2023-04-28T16:37:48.

However, unlike XML Schema rules, durations are in the format for HH:MM:SS, or MM:SS, or just seconds, e.g. 1:00:00 for one hour. However, to allow XML Schema compliant input a duration can also be entered in the normal format such as PT1H for one hour, which appears as 1:00:00. Note that P1M and PT1M specify 1 month and one minute durations, respectively.

3.5.5. Colours

Colours can be entered in normal css style format such as #FF0000 or #F00, or use a small set of colour names such as red. A fourth byte for transparency can be provided if applicable.

3.5.6. Passwords and secrets

There are two cases where the information entered in the config may be sensitive. One case is a password, for example one used for a user log in to the FireBrick. This this case you can provide a simple text password in the config you send, but what you get back will be a hash, e.g. SHA256#92E12F9A333C68690078AC041CD840996EB366A190F7D8966C48AB84A5729F66DCBC7C1A019FC277583684E81015EA. The exact format used may change over time, and if an older hash, such as MD5 exists in the config, it will actually change to a newer hash format automatically next time someone logs in using that password. The hashes include *salt* to make them harder to crack, but none the less it is best to keep config files secure and not reveal the hashes used.

Another special case is the use of a OTP (One Time Passcode) system. You can put a plain text password and a BASE32 OTP seed in the config for a user, and they will come back as a hashed password (as above), and a # followed by base64 coded data for the OTP code.

In addition, there are also *secrets* which are passwords which cannot be stored in a hashed format (because of the way they are used). Both passwords and secrets are only displayed if you have full access to the config. If you have limited access, or select to view the config with secrets hidden then they all appear as "secret". However such secrets are stored in the config file in plain text.

It is possible (though complex) for you to make hashes and OTP seeds off-line and simply store in the config. For more information on how passwords are hashed and OTP seeds are stored, see Appendix J.

3.5.7. IP addresses

Being an IP based router the FireBrick config has a lot of places where an IP address can be entered. In some cases a simple single IP address, but in other cases as an IP and subnet size (CIDR notation) or even a range of IP addresses. Often a list of IP addresses and/or ranges can be specified in a space separated list.

3.5.7.1. Simple IP addresses

Where a simple IP address is expected you can enter in any valid format for IP addresses. In some cases the field may specifically be IPv4 or IPv6 but in most cases either type of IP address can be entered. In the case of IPv6, the :: shortened format is accepted, and used on output. The legacy format, e.g. 2001:db8::192.168.0.1 format is also accepted as per IPv6 RFCs.

There is also a case where up to one IPv4 and up to one IPv6 address can be specified (separated by a space), e.g. when setting the source IP address for some protocol.

3.5.7.2. Subnets and prefixes

In some cases an IP address and subnet length is expected (in CIDR format), e.g. `192.168.0.1/24`. There are two variations of this, one is a *subnet* which includes an IP address and length, such as `192.168.0.1/24`. The other case is where the *prefix* is what matters, and so the IP address can be any within the prefix, so `192.168.0.1/24` would actually read back as `192.168.0.0/24`.

Where the subnet is one IP, e.g. `192.168.0.1/32` it can be provided as, and reads back as a simple IP address, i.e. `192.168.0.1`.

3.5.7.3. Ranges

In some cases a range of IP addresses is needed, for example when making a filter for firewall rules or allow lists.

For IPv4 addresses this can use a format using a hyphen, e.g. `192.168.0.100-199`. Where the range is wider there may be more after the hyphen, e.g. `192.168.0.100-1.99` is all from `192.168.0.100` to `192.168.1.99`. Ultimately it can be a complete range such as `10.1.2.3-11.100.2.5`, though that is rarely needed.

Where a range covers all parts after the hyphen then an `x` can be used, e.g. `10.2-4.x.x` for `10.2.0.0` to `10.4.255.255`.

A range can be provided as a simple IP address (for one IP in range) or as a CIDR format. IPv6 address ranges can only be CIDR format prefixes. If a range happens to be a CIDR range it shows in that format. E.g. `192.168.0.0-192.168.255.255` will read back as `192.168.0.0/16`.

In some cases a list of IP ranges can also include named IP groups. In which case the IP groups are checked by name, firstly in the case of a `rule-set` where `ip-group` entries are specified, then system `ip-group` entries, and then named `subnet` entries (including DHCP client subnets).

3.5.7.4. Prefix filters

In some cases a filter needs to be specified to allow a set of prefixes to be defined, such as BGP filters.

In this case the format has a range of prefix lengths using a hyphen, e.g. `10.0.0.0/8-24`. The prefix must match at least the lower size bits (`/8` in that example), but can be any size in the range. Where the format uses either end of the range it can be omitted either side of the hyphen, e.g. `10.0.0.0/8-` is the same as `10.0.0.0/8-32`.

Some cases need to be more complex and a colon is used to add a third bit length, e.g. `10.0.0.0/8:16-24` means all prefixes within `10.0.0.0/8` which are a prefix length of 16 to 24 bits, so that would include `10.1.0.0/17` in that case.

3.6. 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

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 locally connected IP address. If you don't require such open access, you may wish to restrict access using the settings described in Section 12.3.

3.6.1. 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 is normally displayed on the left side, although on smaller displays (e.g. smartphones) it may be accessed via the 3 horizontal lines in the top right corner.
- a 'footer' area at the bottom of the page, showing the current software version
- the remaining page area contains the content for the selected part of the user-interface

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.

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.

3.6.2. Config pages and the object hierarchy

The structure of the config pages mirrors the object hierarchy, and therefore they are themselves naturally hierarchical. Your position in the hierarchy 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)
```

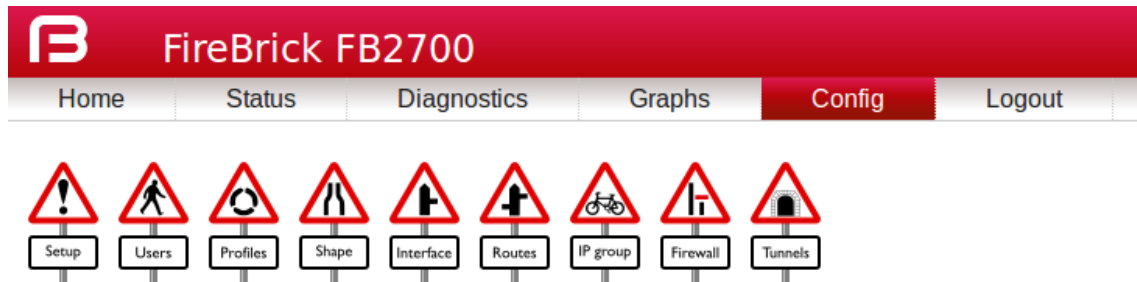
This shows that the current page is showing a rule, which exists within a rule-set, which in turn is in the "Firewall/mapping rules" *category* (see below).

3.6.2.1. Configuration categories

¹If 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).

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 3.1 :-

Figure 3.1. 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 3.2 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 3.2. 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 3.6.3.

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.

3.6.2.2. 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 3.3 shows an example for an interface object (covered in Chapter 6) :-

Figure 3.3. Editing an "Interface" object

| | | | |
|---|--|--|--|
| <input checked="" type="checkbox"/> name <small>Name</small> <input type="text" value="WAN"/> | <input type="checkbox"/> comment <small>Comment</small> | | <input type="checkbox"/> profile <small>Profile name</small> |
| <input checked="" type="checkbox"/> port <small>Port group name</small> <input type="text" value="WAN"/> | <input type="checkbox"/> vlan <small>VLAN ID (0=untagged)</small> <input type="text" value="0"/> | <input type="checkbox"/> graph <small>Graph name</small> | <input type="checkbox"/> mtu <small>MTU for this interface</small> <input type="text" value="1500"/> |
| <input type="checkbox"/> ra-client <small>Accept IPv6 RA and create auto config subnets and routes</small> <input type="checkbox"/> true | <input type="checkbox"/> ping <small>Ping address to add loss/latency to graph for interface</small> | | <input type="checkbox"/> log <small>Log events including DHCP and related events</small> <input type="checkbox"/> Not logging |
| <input type="checkbox"/> log-error <small>Log errors</small> <input type="checkbox"/> Log as event | <input type="checkbox"/> log-debug <small>Log debug</small> <input type="checkbox"/> 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 3.4. The hidden attributes can be displayed by clicking on the link "Show all".

Figure 3.4. Show hidden attributes

There are additional attributes which have not been shown. [Show all](#)

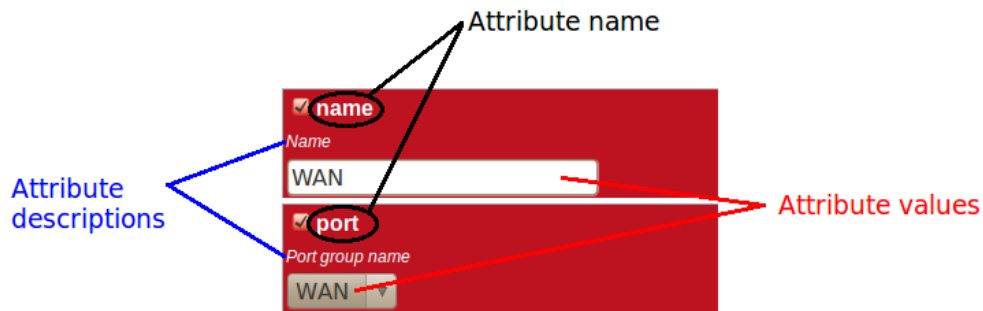
Each brick in the wall contains the following :-

- 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 3.5 :-

Figure 3.5. 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).

3.6.3. 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 3.6: "Up" - move one level up in the object hierarchy, "Prev" - Previous object in a list, and "Next" - Next object in a list.

Figure 3.6. Navigation controls

Interface :: interface 2 of 3 (LAN)



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 3.6, 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 described 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 set up 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 3.6) is used to delete the object you are viewing.

3.6.4. 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.

It is also possible to set `auto-backup-url` to a URL (starting `https://`) to automatically post a copy of the config to a server of your choice shortly after any changes. There is a delay of a few minutes after the last change before posting the config. The config post also includes a header `x-Signed` with a digital signature of the config itself using the private key stored in the FireBrick Certificates store against the FireBrick serial number. This should be used to check for authenticity of the posted config.

3.6.5. Customising the layout

It is possible to change the colour of the interface's header and footer banners from the config editor. Select the "Setup" category icon and choose to edit "General system services" and then "Web server settings". Click "Show all" at the bottom of the page, and then select "banner-background". By default, this will also change the colour of the config editor and the highlight text colour (used for hyperlinks and headings). If needed, you can also set those colours separately.

Note

Once you have saved (or test saved) the configuration, you will need to navigate to a different page or reload to see the new colours.

Alternatively you can edit the XML configuration file (see Section 3.7) and set the `banner-background` attribute for the `http` service.

It is also possible to configure an external CSS to use with the FireBrick web control pages, via the `css-url` attribute. This allows a great deal of control of the overall layout and appearance. This can be useful for dealers or IT support companies to set up FireBricks in a style and branding of their choice.

3.7. Configuration using XML

3.7.1. 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 hierarchy.

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 |
|----------|-----------------------|
| < | < |
| > | > |
| " | " |
| & | & |

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.

3.7.2. 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.

3.7.3. 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.

3.7.4. 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/fb9000/" ❶
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://firebrick.ltd.uk/xml/fb9000/ ...
  timestamp="2023-03-14T12:24:07Z"
  patch="8882">

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

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

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

  <services> ❺
    <time/>
    <telnet log="default"/>
    <http/>
```

```

<dns domain="watchfront.co.uk"
  resolvers="81.187.42.42 81.187.96.96"/>
</services>

<port name="WAN"
  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"
  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 Top level attributes are effectively read only as they are overwritten when a config is uploaded. These are for your information, and include things like the date/time the config was uploaded, and the username and IP address that uploaded the config, etc.
- ❷ sets some general system parameters (see Section 4.2)
- ❸ defines a single user with the highest level of access (DEBUG) (see Section 4.1)
- ❹ defines a log target (see Chapter 5)
- ❺ configures key system services (see Chapter 12)
- ❻ defines physical-port group (see Section 6.1)
- ❼ defines an interface, with one subnet and a DHCP allocation pool (see Chapter 6)

3.8. 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

3.8.1. 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 3.6.4).

Note

When fetching the config in this way, the initial config attributes will include formal namespace and `xsi:schemaLocation` attributes. These are not normally shown on the config editor via the web interface, and are ignored when uploading a config.

3.8.2. 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 `multipart/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"
```

Note

You can also include `--form override=true` to force the config to be loaded even if it has minor (recoverable) errors, e.g. if it is config for older version of FireBrick.

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 3.6. 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

Admin users :: user 1 of 1

The screenshot shows a web interface for configuring a user. At the top, there are four buttons: "Up", "New", "Erase", and "Help". Below these is a section titled "User names, passwords and abilities for admin users". This section contains several red-bordered boxes with white text, each representing a configuration field:

- name**: User name, value: wallace
- comment**: Comment
- profile**: Profile name, value: None
- password**: User password, value: masked with dots
- full-name**: Full name
- otp**: OTP serial number
- timeout**: Login idle timeout (zero to stay logged in), value: 5:00
- config**: Config access level, value: full
- level**: Login level, value: 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, but can access control switches for which the user has access. |
| GUEST | Guest user, access to some menu items |
| USER | Normal unprivileged user |
| ADMIN | System administrator |
| DEBUG | System debugging user |

Tip

In general you only want to use NOBODY, ADMIN or DEBUG levels.

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.

Config access also requires at least `admin` level for their *login level* to access config via the web interface.

It is possible to *test* a new config, causing it to be applied but not saved to permanent storage. This test config automatically reverts after a few minutes if not *committed*, or if the brick restarts, e.g. power cycle. It is recommended that you test a config first to ensure you have not locked yourself out, and there is a user level to force you to have to test configs first.

Table 4.2. Configuration access levels

| Level | Description |
|-------------------|---|
| <code>none</code> | No access unless explicitly listed |
| <code>view</code> | View only access (no passwords or hashes) |
| <code>read</code> | Read only access (with passwords and hashes) |
| <code>demo</code> | Full view and edit access, but can only test new config, not save them. |
| <code>test</code> | Full view and edit access, but must test new config before committing it. |
| <code>full</code> | Full view and edit access. |

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 17), 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 should 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 12.3 and Section 12.4), 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 of 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 implicitly 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.

4.1.5. Password change

Normally, all config data is updated via the config edit process, and this allows a new password to be set for any user.

However, there is also a menu to allow a logged in user to change their own password. This does not require the user to have any config access permission. Simply enter the old password, and the new password twice and the password is updated.

If you have set up an OTP configuration for a user, then you cannot change the password simply using the configuration editor (unless also setting a new OTP configuration from scratch or removing it). In such cases the password should be set using the password change web page. This is also good practice as it avoids the administrator knowing people's passwords.

4.1.6. One Time Password (OTP)

A login to the FireBrick normally requires only a username and password. However you can configure an additional security measure using a One Time Password (OTP) device. These are available as key fobs that show a code, but are more commonly done by use of a mobile phone application.

In order for the device to work you need a *key* which is known to the FireBrick and the device. However, this is very simple to set up. A user can access the Password / OTP menu where a random *key* is allocated and displayed within a QR 2D bar code. Most authenticator applications simply scan the QR code and start showing the 6 digit number on the display (which changes every 30 seconds). You then enter your password and a code from your device to complete the process.

It is possible for anyone with configuration access to edit your user settings and remove the OTP settings if you wish. This can be useful if you lose or break the phone, for example. You may want to keep a local configuration user as a backup as well, as OTP cannot be used if the clock is not set for any reason.

When you login, after you submit your username and password you are asked for a code from the authenticator to complete the login process.

It is also possible to enter the password as the authenticator code followed by the configured password. This is useful if using HTTP authentication to access a web page where there is no separate option for the authenticator input to be provided.

If OTP is configured you can leave the password blank (which is not normally allowed) and hence use the authenticator code as the entire password, though this is not recommended for security reasons as it also means the TOTP seed is recoverable from the config.

Note

Technical details to allow you to create configs with password and OTP seed hashes are described in Appendix J.

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 |
|-----------------------|--|
| <code>comment</code> | General comment field |
| <code>contact</code> | Contact name |
| <code>intro</code> | 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 log in to a command-line session. |
| <code>location</code> | 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 overnight. The reboot will close all L2TP connections first. The reboot will close all BGP sessions first. 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

Sometimes the configuration will need to be transformed during an upgrade, these transformations will be automatically performed when needed as part of the upgrade (see Section 3.4 for details). However we do not keep the update procedures around forever, instead we flag certain releases as breakpoints that upgrades must go through sequentially to maintain a valid configuration.

Note

Between breakpoints changes to config could be made, for instance in alpha releases. These transformations will be kept around until at least the next breakpoint release to maintain compatibility.

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, we recommend saving a new version after upgrading to a breakpoint release.

If you use automated methods to configure your FB6000, you will need to check the documentation to see whether those methods need updating before the next breakpoint release.

Downgrading the software past a release that makes configuration changes will (provided the running version is recent enough to have this feature) show a warning that the current configuration patch version is too recent. In this case check that the configuration looks correct for the running version, and correct or remove the `patch` attribute (this can be done in the XML editor, or by making any change in the UI based editor).

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      FB9001 TEST Abbotscliffe (V2.00.010 2023-09-18T08:29:31)
```

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 log in 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



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 |
|--------------------------------|--|
| <code>sw-update</code> | 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 <code>false</code> to achieve this. <ul style="list-style-type: none"> <code>false</code> : Disables auto upgrades <code>factory</code> : Only download/install factory releases - this is the default if the attribute is not specified <code>beta</code> : Download/install factory or beta releases <code>alpha</code> : Download/install factory, beta or alpha releases |
| <code>sw-update-profile</code> | Specifies the name of a profile to use to control when software upgrades are attempted (see Chapter 8 for details on profiles). |
| <code>sw-update-delay</code> | Specifies a minimum number of days after release to attempt the upgrade (intended for automating staggered upgrades). |

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

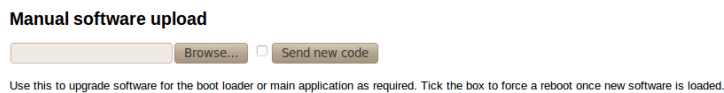
This method is entirely manual, in the sense that the brick itself does not download new software from the FireBrick servers, and responsibility for loading breakpoint releases as required lies with the user.

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



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 upon power-up, initialises the 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 H.

4.4.1. LED indications

4.4.1.1. Port LEDs

Whilst the bootloader is waiting for an active Ethernet connection, the green and yellow LEDs by the ports flash in a continual left-to-right then right-to-left sequence.

Note

The same port LED flashing sequences (but this time in red) 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/status 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 into 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 set up multiple log targets 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 H for details on using this command). Chapter 17 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 17. 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 <code>log-error</code> 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 <code>log-error</code> is not set but <code>log</code> is set then errors are logged to the <i>log</i> 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 set up 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` attribute, 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-support` 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. There may be other cases where log entries are made to this log target to advise the FireBrick support team of unusual events.

Caution

Please do not use the `fb-support` log target for any other logging. This is normally only used for crash/error reporting back to FireBrick.

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 determine what you wish to 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-support | System support events (e.g. panic stack traces). |
| 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 how to set up *Ethernet* interfaces and the definition of subnets that are present on those interfaces.

For information about other types of 'interfaces', refer to the following chapters :-

- Point-to-Point Protocol over Ethernet (PPPoE) - Chapter 10
- Tunnels, including FB105 tunnels - Chapter 11

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

The FB6000 has two physical ports and no internal switch. This means that the port group configuration can either be the default where each port is in one group, or where both ports are in a *trunked* group.

The port group has a *trunk* setting which defaults to being true. When there are two ports in the port group this is the only option. When only one port, it makes no difference. It is included for compatibility with other models.

The FireBrick supports *LACP* (Link Aggregation Control Portocol) which is used to coordinate and control trunked port groups by exchanging LACP packets over the links. There is a *lACP* setting in the individual ethernet port settings which can be used to control LACP's behaviour, as follows:

- *lACP="false"*: It is assumed that the link is not connected to a device supporting LACP. LACP packets are not sent, and any received are ignored. The ports in a trunked port group will be used for aggregation when the physical link is up, after a short delay to ensure the partner is ready.
- *lACP="true"*: The link must be connected to an LACP-enabled device in order to function. LACP packets are sent, and the link will only be enabled for traffic when LACP negotiation is successful.
- *lACP not set*: This is the default (Auto) setting. LACP packets will be sent if the port is part of a trunked port group, or if LACP packets are detected from the linked device. If LACP is not detected, a non-trunked port will always be enabled, while a port which is part of a trunked port group will only be enabled if it is the lowest-numbered (leftmost) port in the group. There will be a short delay after the port is physically up to allow for detection of LACP. When LACP is detected, the LACP negotiation controls the availability of the port.

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.

Appendix D contains a brief overview of VLANs and the concept of broadcast domains.

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 definition. 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 14)

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 definition. 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. Source filtering

The interface has an option to `source-filter` traffic received from the interface. This means checking the source IP of all traffic that arrives.

Setting source filtering to `true` will only allow IPs that would be routed back down that interface. That is the most restrictive setting and only really makes sense for a LAN such as a customer connection, rather than a connection to another network, as it would impact any triangular routing. Using `self` is usually safer.

Setting source filtering to `self` will allow IPs where replies are routed somewhere off the FireBrick, but block any attempt to spoof a source address that is one of the FireBricks own IPs.

Setting source filtering to `nowhere` will block IPs where a reply would not route anywhere (i.e. blackhole, or nowhere/deadend). It allows packets where replies would be back to the FireBrick itself which is an unusual requirement.

Setting source filtering to `blackhole` is the least restrictive, blocking IPs where replies would go to an explicit blackholed route. Blackhole routes can be configured manually or via BGP using a specific community tag to define routes which are specifically invalid (without even an ICMP error).

Tip

The routing look up to check the source IP is normally done in the routing table of the interface. However, it is possible to set a `source-filter-table` which allows the check to be done in a different routing table. This usually only makes sense when used with the `blackhole` option. It allows a separate routing table to be used to define source filtering explicitly if needed.

Note

Link local IPv6 addresses starting FE80 are always allowed, as is the 0.0.0.0 null IP for DHCP usage. IPv6 addresses within `2002::/16` are treated as the embedded IPv4 address for filtering checks.

6.2.1.2. 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 acquire multiple independent IP addresses by DHCP on the same interface if required.

When a subnet is configured to be a dhcp client, you may sometimes want to renegotiate (e.g. if you have disconnected and connected to a different DHCP server). If you are an ADMIN (or DEBUG) user, this can be achieved from the web user interface. Select the "Subnets" item under the "Status" menu, and then select a particular subnet by clicking on its ID number in the left hand column. For subnets that are DHCP clients, you will see a "Renegotiate DHCP" button.

Caution

Renegotiating DHCP on an active subnet will interrupt your connection to the external network, and it is very possible that you may be allocated a different IP address.

6.2.1.3. Using SLAAC (IPv6 router announcements) to configure a subnet

For automatic IPv6 addressing (client), unlike for IPv4, you do not create a separate subnet, instead you create `ra-subnet-template` entries. These have matching criteria and give options for the subnet that will be generated on the Interface when a suitable router announcement is received (see `subnet-template` for details). If no entries match then a subnet with the default options will be created, however if no entries are present then the Firebrick will not act as a route announcement client, and so no subnet will be created. The router announcement can set up an IPv6 address, gateway and DNS servers automatically.

Note that the gateway *localpref* is not something that can be configured, and defaults to 100X, 200X, 300X, or 400X depending on the priority setting in the router announcement. The X is based on the scope of the IPv6 address so a global scope has higher localpref.

Note

DHCPv6 client is supported for interfaces, and is active if router announcements seen indicate managed IPs ('M' flag).

6.2.1.4. Providing IPv6 addresses to devices on a network (IPv6 router announcements)

An IPv6 subnet can have `ra` enabled - this provides router announcements that allow devices to automatically obtain an IPv6 address. When set you will normally want to disable `ra-client` in the interface settings.

There are a number of RA settings (see `subnet`) which can be configured to allow DNS and other details to be provided.

A simple DHCPv6 server can also be enabled - this is of limited use at present and simply provides a fixed address to each client (using a hash).

6.2.1.5. IPv6 prefix delegated subnets

The interface configuration allows an interface to be set for prefix delegation (default if wan not set), i.e. an IPv6 subnet configured automatically based on settings provided by your Internet Service Provider. Prefix delegation is then requested for such interfaces when DHCPv6 client is used (e.g. on PPPoE interfaces).

Once prefix delegation is set up, the assigned subnet is used to provide IPv6 addresses and gateway to devices on the interface using SLAAC.

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 (eg addresses already in use, perhaps through a device configured with a fixed static address).

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) 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 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 reused 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, or the client name using the `client-name` attribute.

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 existing allocation and then force the client to issue

a new DHCP request (e.g. unplug the ethernet cable, do a software 'repair connection' procedure or similar). See the `show dhcp` and `clear dhcp` CLI commands in Appendix H for details on how to clear the allocation. Chapter 17 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.

6.2.2.2. Restricted allocations

You can restrict a pool to apply only to devices with specific MAC addresses, client names or vendor class names using the `mac`, `client-name` and `class` attributes on the `dhcp` object. The client and class names can be specified using wildcard strings, where the characters '*' and '?' stand for any run of characters, and any single character, respectively. The value specified for the `mac` attribute can be a list of partial MAC addresses, where each item can be less than a full 6-byte address. Any device whose MAC's leading bytes match one of the items in the `mac` list is acceptable. [The fixed-allocation example above is actually a special case of this general method for restricting allocation pools, where a single MAC address was specified in full.]

For example, as discussed in Appendix B, the first three octets (bytes) of a MAC address identify the organization (often the end product manufacturer) which is registered to 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 can 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 of your VoIP phones are from one manufacturer, as you can have appropriate firewall rule(s) that apply to addresses in that pool.

The following example illustrates this technique. It will match any devices which have MAC addresses beginning 00:04:13 or 00:0E:08, and which also have a vendor class name containing the string *phone*:

```
<interface ...>
...
  <dhcp name="VoIP"
    ip="10.20.30.40-50"
    mac="000413 000E08"
  class="*phone*"
    dns="8.8.8.8"
    log="DHCP-Phones"
    comment="VoIP phones" />
...
</interface>
```

The algorithm used to determine which pools apply to a particular device is as follows:

- If no restricted pools (ie those with one or more of the `mac`, `client-name` or `class` attributes present) match the device's properties, then all non-restricted pools apply.
- If *any* restricted pools match the device's properties, then *only* restricted pools which match the device apply. Furthermore, a scoring system is used to select which of those pools to use based on how well the device's properties match. An exact match scores higher than any partial or wildcarded match and a MAC match scores higher than a client-name match, with a class-name match scoring the least. The score for a partial or wildcarded match is based on the number of bytes or characters which were explicitly matched.
- Only the pool or pools with the highest score are considered.

Note

While this may seem rather complex, it achieves the intuitively-expected result in most cases - for example it allows a pool to be set up for a general class of device or a range of MAC addresses, and for more specific pool entries to be included which will take precedence for individual devices, eg with a full MAC address or a specific client-name.

6.2.2.3. Special DHCP options

For each pool, in addition to the common DHCP options to be supplied to the client device which you can configure using recognized attributes (eg `gateway`, `dns`, `domain`), you can also supply other DHCP options, specified as a string, IPv4 address or number, or even as raw data in hexadecimal. You can force sending of an option even if not requested.

For vendor-specific options (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 option type 43 with the code and length for the option which can be string, IPv4 address, number, or hexadecimal.

6.2.2.4. Logging

The logging options allow a general `log` but also specific logging for various types of allocation such as *new* allocations or *renewed* allocations, etc. These are also logged to the `log` target if it is different. The format of these logs is a JSON object. .

The logging on the DHCP only relates to allocations that are tied to that pool. For cases that cannot be defined as a specific pool, the logging is done at the `interface` level with `log-dhcp` setting, and these are not generally in JSON format. Note that logging of expiry of DHCP allocations applies at the interface level as the pool is not known at that point.

6.2.3. DHCP Relay Agent

You can configure the FireBrick to operate as a DHCP/BOOTP Relay agent simply by setting the `dhcp-relay` in the `interface` object to the IPv4 address of the remote DHCP server.

If you also configure a DHCP allocation on the same interface, this is checked first, and if there are no suitable allocation pools or IP addresses available then the request is relayed. Normally you would configure either a relay or a pool and not both.

The top level `dhcp-relay` configuration allows you to configure the FireBrick to be the remote server for a DHCP/BOOTP Relay Agent. The `relay` attribute allows specific pools to be set up for specific relays. The `table` and `allow` allow you to limit the use of the DHCP Remote server to requests from specific sources - note that renewal requests come from the allocated IP, or NAT IP if behind NAT and not necessarily from the relay IP.

The `allocation-table` attribute allows for this pool of IPs to be placed in a separate table, thus allowing it to be independant from other DHCP allocations on the FireBrick and to allow different overlapping pools for different relay endpoints, which is not uncommon if the endpoints are behind separate NAT routers.

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.

Note

Whether the settings can be applied to a given SFP port will depend on the capabilities of that SFP. Obviously crossover doesn't make sense for a fibre connection, and many copper SFPs don't support this either.

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-negotiation 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-attached 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 <i>black hole</i> (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 *suppressed* 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.

When using BGP you can also define a *network* which is announced by default, along with any *dead-end-community*, and treated otherwise the same as `nowhere`.

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. Refer to Chapter 11 for details on how to achieve this via the `routes` attribute on the tunnel definition objects.

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 decision is made on a per packet basis as to which interface to use. 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 ISP.

For L2TP use as an LNS, the graph created for each L2TP session has an egress speed automatically set based on the speed details sent on the L2TP connection. These can also be overridden by a RADIUS response. The effect of this is that multiple lines that are connected to the same LNS and have the same IPs routed to the lines will automatically per packet bond traffic down those lines.

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. Similarly, 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. When considering the state of another profile, it is the previous second's state that is considered - i.e. profile states are all updated in one go after considering all profiles.

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 profile logic tests are also done based on the defined *interval*.

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 permanently 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 :-

- PPPoE connection state : the `ppp` attribute lists one or more PPPoE connection names (see Chapter 10) - if *any* of the specified connections are up, this `pppoe-state` test will pass
- Routable addresses : the `route` attributes lists one or more IP addresses (full addresses, not CIDR prefix ranges) - only if *all* the addresses are 'routeable' - 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
- DHCP addresses : the `dhcp` attributes lists one or more IPv4 addresses (full addresses, not CIDR prefix ranges) or names - if *any* of the listed names or IPs has a current DHCP lease issued, then this test will pass
- VRRP state : the `vrrp` attribute lists one or more Virtual Router group membership definitions (see Chapter 14) by name - if the FB6000 is not the master device in any of these Virtual Routers, this test will fail
- Port state : the `ports` attribute lists one or more physical Ethernet ports. If any of these ports is up then the test passes.

Tip

You can also control port state with a profile, so you could have a port come up if another port is down to create a fallback arrangement.

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 change of state due to 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.

Note

A ping of IPv4 address 0.0.0.0 is a special case, which causes a ping of the gateway address.

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 persistent data* (along with DHCP settings, etc), so survives a power cycle / restart. The control switch uses `initial` as the initial state when first added to the config, but at start up it picks up the state of the stored state.

These profiles can be used as simple on/off controls for configuration objects. The following shows an example of two fixed-state profiles and one control switch, 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"/>
```

Note

The control switches ignore other tests, just like other manual settings, but can be combined with `and`, `or` and `not` settings - these have the effect of forcing the control switch one way. E.g. if an `and` profile is *off* then the control switch is forced off. If it is *on* then the control switch can be manually set on or off as needed.

Note that the value of the `invert` attribute is ignored when manual override is requested. If we add special case switched profiles that are used internally, these will be named starting with `fb-`, so it is best not to name your own profiles starting `fb-`. An example of the ACME renewal control profile is given below.

Note

There is a special case where profiles can be used to interact with the ACME certificate management. A `control-switch` profile which has `fb-` followed by the name of the CA certificate authority, e.g. `fb-letsencrypt.org` will be activated at the start of the validation phase, and deactivated at the end. This profile can therefore be used to change firewall rules, etc, to allow ACME access to validate the certificate. You can set `invert` to reverse the operation of the switch. Use with care.

8.2.5. Scripting

As with many things in the FireBrick it is possible to script access to the control pages. The control profiles are a particularly useful example of this as it allows external scripts to control if a profile is active or inactive.

Obviously security is a concern, and it is possible to create a user allowed access only from some IP addresses, and even allowed NOBODY level access, but listed as a user allowed to set a specific control profile. This then allows a script using a username and password to control the profile from specific machines.

For example, setting a named profile:-

```
curl --silent --user name:pass 'http://1.2.3.4/profile?set=profilename'
```

And unsetting a named profile:-

```
curl --silent --user name:pass 'http://1.2.3.4/profile?unset=profilename'
```

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 or SVG 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 from the FB6000 via the web User Interface - to view a graph, click the "PNG" or "SVG" 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 I 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.

Graphing relies on the FireBrick's internal clock and so graphs will not work reliably if the clock is not set for any reason. However, you can still set an object's `graph` attribute and use it for traffic shaping.

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.

Graphs can sometimes be created automatically and may have speeds applied. For L2TP sessions the circuit ID (which may be overridden by RADIUS auth responses) is used to make a graph for the session.

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 dynamically 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 the maximum rate that should apply over a long term, but actual speed is set dynamically between min and max rates.

The rate is initially set to the maximum. Actual usage below the target builds up *credit*. This *credit* is limited to allow a defined minimum burst time at the maximum rate. So if not used for a while the maximum rate can be used for the minimum burst time. However continued use at more than the target will accumulate *over usage*.

Once this *credit* is used up the rate starts to drop. It will drop all the way to the minimum set if necessary.

Once the *over usage* is cleared, but less usage or usage capped to below the target rate, the rate increases to the target rate.

Once a further *credit* is accumulated by lower usage than the target, and this is at a level to allow the minimum burst time at maximum rate, the rate increases to the maximum rate again.

9.1.5. Shared shapers

Shapers can be shared between FireBricks with a common broadcast domain.

The interface to broadcast and listen on for sharing is set via the `share-interface` setting in the global `com` options. Once that is configured then setting the `share` option on a `shaper` will combine its state with that of other FireBricks when calculating any damping required.

Note

Shared shapers are identified in the broadcast domain by their name, so this must be the same on all FireBricks participating in the share.

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
- If the packet is carried via an L2TP tunnel of any sort, there can be an aggregate shaper for the tunnel (e.g. the broadband carrier). There can also be a *class* applied to the session which is an aggregate shaper for an arbitrary group of sessions (often used when reselling broadband).
- When passing through an L2TP tunnel, the session typically has a graph and shaper based on the circuit ID which is specific to that session. This is important for bonding multiple sessions as it controls the levels of traffic sent via each session.
- Obviously traffic could come in via one L2TP tunnel and go back out via another, incurring yet another set of shapers as above.
- PPPoE links can also have a defined shaper, which is important when bonding multiple links as it is used to decide how much traffic goes via each link.
- 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. PPPoE

The FB6000 can either operate as a PPPoE client or as a PPPoE endpoint Broadband Remote Access Server (BRAS). A client connection is typically used to connect to an Internet Service Provider, either via a suitable PPPoE modem, bridging router, or direct connection. The PPPoE BRAS functionality might be used by service providers to provide connectivity, acting as a gateway between a carrier network (e.g. Broadband or mobile carrier) and the Internet.

10.1. PPPoE client

The typical usage is to use one or more ports on the FB6000 each connected directly to a suitable PPPoE device such as a bridging router.

The PPPoE device is usually very dumb and may not need any configuration at all. The FireBrick is responsible for logging in to the ISP using the PPPoE link, and the configuration for this is part of the FB6000's configuration and not the router. This makes it very easy to make use of spare routers, etc, without the complication of configuring additional devices.

It is possible to connect more than one PPP device to a single FB6000 port using an Ethernet switch. If you do this then you ideally need a switch that handles VLANs (see Appendix D if you are not familiar with VLANs) so that each router can be logically connected to a different interface on the FireBrick. It is also a good idea to have a switch that supports *jumbo frames* where the endpoint supports them (FTTC, FTTP, and via suitable modems BT 21CN and TalkTalk).

Note

This section contains information relating to access network services (such as DSL and Fibre-To-The-Cabinet) available in the United Kingdom. Although this information will not be directly applicable to services available in other countries, the concepts are the same - with appropriate knowledge of your ISP service, and suitable equipment, the FB6000 should work equally well with services that are available in other countries.

10.2. Types of DSL line and router in the United Kingdom

In the UK there are various types of DSL line and router than can be used. Any device that supports PPPoE can work with the FireBrick, but some options are only available with some devices, as listed below :-

- BT 20CN or 21CN lines can support PPPoE and PPPoA *on the wire*. This means you can use them with a PPPoE/A modem (such as a *Vigor V-130 out of the box*, or with a bridging router such as the *Zyxel P660* configured in bridge mode. BT support baby jumbo frames too.
- Other carrier PPPoA lines only support PPPoA, in theory. In practice they also support PPPoE. This means you can use a PPPoE/A modem, or a bridging modem.
- TalkTalk lines support both PPPoA and PPPoE, and so can work with a bridging modem. They support baby jumbo frames too.
- BT FTTC lines come with a VDSL modem which supports PPPoE directly so no extra equipment is needed to connect to the FireBrick.
- BT FTTP lines terminate on an active NTE which supports PPPoE directly so no extra equipment is needed to connect to the FireBrick.

For other types of lines in the UK, or those in other countries, you need to know what they can do *on the wire* (PPPoA or PPPoE) and have a suitable modem/router to talk that protocol and convert to PPPoE on the LAN link to the FB6000. It seems most DSL routers will bridge PPPoE on the wire to PPPoE on the LAN, but few will act as a PPPoE access concentrator. The Vigor V-130 is one of the few that handle PPPoA on the wire and PPPoE link to the FB6000.

A significant benefit of the Vigor V-130 is that it works with *no configuration* on BT 20CN and 21CN lines as well as other carrier PPPoA lines and TalkTalk lines - you just plug it into the line and the FB6000 and it just works. There are also modems that work in bridged mode, and support baby jumbo frames allowing PPPoE through to the carrier BRAS with full size MTU.

For fibre to the cabinet (FTTC) and fibre to the premises (FTTP) service you connect the FB6000 directly to the service (BT supplied modem) with no extra equipment.

10.3. Defining PPPoE client links

A PPPoE link is defined by a `ppp` top-level object. To create or edit PPPoE links in the web user interface, select the "Interface" category icon - under the section headed "PPPoE settings" you will see the list of existing `ppp` objects (if any), and an "Add" link.

For most situations, configuring a PPPoE link only requires that you specify the port that the router/modem is connected to and the login credentials i.e. username and password. The port group is specified via the `port` attribute on the `ppp` object, and credentials are specified via the `username` and `password` attributes.

If you are connecting multiple routers/modems via a VLAN capable switch to a single FB6000 port, you will also need to specify the VLAN used for the FB6000 to router/modem layer 2 connection - this is done by setting the value of the `vlan` attribute too.

As an example, if you were to connect a single modem/router directly to port 1 on your FB6000 (i.e. not using VLANs), and you had decided to name the port group `PPP_PORT` then the configuration needed, shown as an XML fragment, would be :-

```
<port name="PPP_PORT" ports="1"/>
...
<ppp port="PPP_PORT" eth="1" username="..." password="..." />
```

Note

It is common to connect a port directly to a PPPoE modem - in this case it can be beneficial to configure the `eth` setting to that port as well. This causes the Ethernet port to be powered down briefly when the PPPoE link closes for any reason, and for the re-try (PADI) sending to start quickly when the link comes up. The port reset can work around a known bug in many bridging broadband modems when used with fixed IP services.

You may also want to give the PPPoE link a name, by setting the `name` attribute - you can then reference the link in, for example, a profile (see Section 8.2.2.1).

There are a number of additional options (see below), but for most configurations this is all you need. It causes the FB6000 to connect and set a default route for internet access via the PPP link.

10.3.1. IPv6

If your ISP negotiates IPv6 on the link, then a default route is set for IPv6 traffic down the line. If the ISP handles ICMPv6 prefix delegation then an IPv6 block will automatically be assigned to your LAN. If not, then you could manually configure the IPv6 prefix the ISP is providing. There are options to control which interfaces get automatic prefix delegations in this way.

10.3.2. Additional options

10.3.2.1. MTU and TCP fix

Normally PPPoE operates with a maximum packet size of 1492 bytes - this is due to the 8 byte PPPoE header that is used, and the normal 1500 byte payload limit of an Ethernet packet. The FB6000 includes an option to set the PPPoE MTU, so that when used with equipment capable of jumbo frames (such as BT FTTC and FTTP services, and with appropriate ADSL bridging modems) this allows use of slightly larger frames to provide a 1500 byte MTU. To achieve this, simply set the `mtu` attribute to a value of 1500. By default the `tcp-mss-fix` attribute is also set, which means when working with a smaller MTU such as 1492, any connections that try and establish 1500 byte links are adjusted on the fly to be the lower MTU. This avoids problems with a lot of corporate and bank web sites that do not handle MTU and ICMP correctly. Typically your ISP will be doing this TCP fix for you as well.

Testing has been done which confirms setting `mtu="1500"` works correctly on BT FTTC and FTTP lines, as well as BT 21CN and TalkTalk lines via a suitable bridging modem (Dlink 320B).

Note

Testing using a Zyxel P660R in bridge mode confirms that BT 21CN ADSL lines will negotiate 1500 byte MTU, but it seems the Zyxel will not bridge more than 1496 bytes of PPP payload. If you select more than 1492 MTU and have problems it could be that some device connecting you to the access concentrator cannot handle the larger packets (such as a bridge or a switch). For this reason the default MTU is 1492.

10.3.2.2. Service and ac-name

The PPPoE protocol allows multiple services to be offered, and the `service` setting can be used to select which is available. This is rarely needed and should be ignored unless you know what you are doing. If specified, even as an empty string, then only matching services will be selected.

The name specified via the `ac-name` attribute is the name of the PPPoE endpoint (access controller). In some cases there may be a choice of endpoints and setting this causes one to be selected by name. Again, this is rarely needed, and if specified will only match the name you specify. On some other carrier PPPoE lines, for example, you could select a specific LAC by name if you wanted to.

10.3.2.3. Logging

The PPP connection status, and PPP negotiation can be logged by setting the `log` attribute to a valid log target.

The `log-debug` will log the whole PPP negotiation which is particularly useful when debugging connection problems.

10.3.2.4. Speed and graphs

As discussed in Chapter 9, graphs allow you to visualise connections, in terms of their state, traffic rates and patterns etc. By setting the `graph` attribute, you can cause the state of the line, data transferred each way, and current packet loss and latency to be recorded on a graph.

Once you are graphing the PPPoE connection, you can set traffic shaping to control speed (see Section 9.1.2). Alternatively, a PPPoE connection is something you can set a speed limit on directly - setting the `speed` attribute will control the speed of traffic *sent* to the Internet - this is mainly used when bonding PPP links.

As uplink/egress speed can be very important to manage bonded lines, a further setting of `auto-percent` can be set to a percentage, e.g. 95. If set then the Firebrick looks for a connection info string in the final CHAPS connect message for a string in the format of `digits/digits` and assumes the second sequence of digits is an uplink speed in bits/second. The percentage is then applied and the tx speed set. If the `speed` is also set, this acts

as a cap not allowing speeds to be set higher by `auto-percent`. A silly low value for speed in the message will be ignored.

10.4. PPPoE BRAS

To configure the FireBrick as a PPPoE BRAS, you need to configure a PPPoE link (similar to configuring a client) - either by defining a `ppp` top-level object in the XML config or by using the web user interface. PPPoE links can be created from the "Interface" category icon under the section "PPPoE settings", where you can click to "Add" a new link. The mode should be set to `bras-l2tp`.

When the FireBrick is acting as a server, you should also configure the `l2tp` object to contain a suitable `incoming` configuration. This is because the PPPoE connections appear as if they've arrived via L2TP, so they have the same options of local IP termination or relay via L2TP onwards to another LNS. See Section 11.1.1 for more information about the handling of incoming L2TP tunnels.

Note

Since the BRAS uses these "virtual" L2TP connections, many of the options in the `ppp` object are ignored or not used to configure the server. Instead you should configure the incoming PPPoE connection using L2TP settings.

The BRAS can be associated with an L2TP incoming section by matching the `ac-name` of the BRAS `ppp` entry with the `remote-hostname` of the incoming L2TP entry. Note that when `ac-name` is specified on the BRAS, the client must also be configured with a matching `ac-name`. To avoid this, it's also possible to use the `name` element of the BRAS `ppp` entry as this will be used for matching with the `remote-hostname` if no `ac-name` is specified. This makes it possible to associate BRAS `ppp` and L2TP incoming sections without needing to specify an `ac-name` on the client.

Note

If a `remote-hostname` is not specified for an incoming section, then that configuration will match with all remaining BRAS entries, so make sure that specific L2TP incoming sections occur first in your config file.

L2TP settings can be created from the web user interface by selecting the "Tunnels" category icon and selecting "Edit L2TP settings". Select "Add an incoming connection". You can specify various options such as `pppdns` and the `local-hostname` (i.e. the hostname reported by the BRAS).

A simple configuration using local authentication might contain:

```
<ppp port="PPP_PORT" mode="bras-l2tp" name="bras-example" />
...
<l2tp>
  <incoming remote-hostname="bras-example" local-ppp-ip="..." pppdns1="...">
    <match username="..." password="..." remote-ppp-ip="..." />
    <match username="..." password="..." remote-ppp-ip="..." />
  </incoming>
</l2tp>
```

More complex configurations might typically use RADIUS to decide whether the session is accepted and what settings should be applied, or might relay sessions down an L2TP tunnel.

10.4.1. Additional options

Just like for the PPPoE client, the BRAS mode supports baby jumbo frame negotiation to allow full 1500 byte MTU operation (as described earlier).

If an interface is configured to work in PPPoE BRAS mode, then it can accept packets with an additional VLAN tag. This is passed as the NAS_PORT on RADIUS requests relating to the connection. The reply packets have the same VLAN tag added. Where the interface is set up on VLAN 0 (untagged) then the additional VLAN tag is only processed where there is not an interface or ppp setting for that specific VLAN configured.

Note

The FireBrick identifies TR-101 *Agent Remote ID* and *Agent Circuit Id* as called and calling identities. It also picks up *Downstream line rate*. These are standard in BT GEA FTTC/FTTP services.

Chapter 11. Tunnels

The FB6000 supports the following tunnelling protocols :-

- L2TP

L2TP client functionality enables tunnelled connections to be made to an L2TP server

11.1. L2TP tunnelling

Layer 2 tunnelling protocol is a standard way to pass IP packets via a tunnel. It was originally used for dialup connections, connecting the RAS (Remote Access Server) to the LNS (L2TP Network Server), and is still used today within Internet Service Providers for broadband connections. L2TP has the advantage that it is very standard and interacts with a variety of equipment.

The FireBrick provides a means to accept static L2TP connections with configured username, password, and IP routing.

Note

Some systems may refer to L2TP when they actually mean L2TP over IPsec, which was used more commonly before IKEv2 was available in IPsec, and is not supported.

The FireBrick can act as full LNS, designed to allow thousands of connections to be handled by an ISP. See Chapter 16 for more details.

L2TP is usually used by Internet Service Providers, but it can also be used to create point to point tunnels connecting, say, two FireBricks together to pass IP traffic. The configuration for this can be a little confusing because L2TP is designed for multiple connections over separate tunnels between Access Controllers and L2TP Network Controllers, so the protocol involves more than one layer and authentication, and a lot of detailed settings.

Once upon a time the Remote Access Server (RAS) would have had a bank of modems or ISDN links, but these days it would be broadband connections over PPPoE. A RAS would create a *tunnel* to an L2TP Network Server (LNS). Once a call comes in, the RAS would then create a *session* over that tunnel connecting the LNS to the end user. Each tunnel can hold multiple sessions all to the same LNS. The LNS then talks PPP over the tunnel/session to the end user and negotiates login details and IP addressing and so on. It then uses PPP to pass IP packets allowing a connection to the Internet.

If you make a point to point link, one end acts as the RAS, and makes a *tunnel* to the other end acting as an LNS, and then it creates a *session* over that tunnel. The settings needed for this are often a lot simpler than an ISP would use, so there is a lot of the FireBrick configuration you can ignore. It is, however, useful to understand some of the terms and settings, especially the *authentication*.

11.1.1. Incoming tunnel

The FireBrick uses a static configuration to decide if it will accept an incoming L2TP tunnel. This is the *Incoming L2TP* configuration. An incoming tunnel has very few credentials - the main one being the *hostname* it sends. This is matched against the `remote-hostname` (which allows a wildcard, even). You can also match to allow specific RAS IP addresses, as well as the routing table used. The `secret` needs to be the same on the RAS and LNS.

Once the first match is found, the L2TP tunnel can be accepted. Most of the remaining configuration settings are not related to the tunnel as such, but define the default settings for any *session* that comes in over the tunnel. These settings can normally be omitted as they will have sensible defaults anyway.

A single tunnel config allows multiple instances of that tunnel to be created. If you are making multiple connections you may want to create different tunnel hostnames to allow fine control of specific features.

Alternatively you may want to have one incoming tunnel configuration, even matching any hostname, and set specific session details using RADIUS. As an ISP, you will typically have an incoming tunnel config for each carrier, or possibly service type.

11.1.2. Incoming session

Once the RAS has a tunnel to the LNS it can create a session. The FireBrick acting as the LNS has to decide to accept the session or not, and what settings to use for the session.

The FireBrick acting as an LNS can use a static configuration to accept a session, or use RADIUS. An ISP would typically use RADIUS which allows a separate authentication service to decide if to accept the session and the settings that apply.

The static configuration consists of *match* settings. These are part of the relevant Incoming L2TP configuration for the tunnel being used. The main criteria is `username` (which can be wildcard, even), but can also match `calling-station-id` and `called-station-id` which would have once been phone numbers, but are typically some sort of circuit ID and service ID in broadband. The `password` has to be correct, if specified. Making a point to point L2TP link allows all of these to be set on the outgoing connection.

Having found the first match, there are a few settings which can be controlled. This is not the full set of *session* settings, but most can be set as a default in the Incoming L2TP tunnel configuration if required on a point to point static configuration. The main setting is `remote-ppp-ip` which defines the IPv4 endpoint for the far end. If this is set then the session is accepted and routing for this IP is negotiated and routed. Additional routes can also be set.

If the `remote-ppp-ip` is not defined, then *relay* settings can be defined to allow the session to be relayed down another L2TP tunnel. These include the hostname, IP, and secret to use.

If not static match is made then RADIUS is used. As an ISP it will be unusual to have any static incoming session matching apart from special test cases.

If a profile associated with an incoming connection is disabled then existing connections will be allowed to persist, however no new ones can be established. This is useful if you need to gracefully drain a particular set of connections.

Note

The config makes use of prefixes like `local-...` and `remote-...`. These are relative to the device you are configuring. So a config to link two FireBricks will have `local-hostname` one end matching the `remote-hostname` the other. Similarly settings like `tx-...` and `rx-...` are relative, tx being transmit/send from this device, and rx being receive to this device.

Note

As with other tunnels, it is often useful to set the payload table to a different routing table to the tunnel traffic, to avoid sending tunnel traffic down the tunnel. This is particularly important for an outgoing tunnel as the default is to route all traffic (i.e. `0.0.0.0/0` and `::/0`) down the tunnel when connected. You can configure specific routes to be passed down the tunnel if you wish. Routing in to this tunnel depends entirely on the configuration at the other end.

Tip

The debug level logging is ideal for showing the details of the low level connection and PPP negotiation. This is recommended when first setting up such tunnels, and should be checked at both ends to identify any errors or problems.

Chapter 12. System Services

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

Table 12.1 lists the services that the FB6000 can provide :-

Table 12.1. 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 <i>resolvers</i> |
| RADIUS | Configuration of RADIUS service for <i>platform RADIUS</i> for L2TP. Configuration of RADIUS client accessing external RADIUS servers. |

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 list 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.

12.1. 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 if there is a denial of service attack against the web interface. When using the FB6000 as an LNS you may be allowing access to CQM graphs linked from control systems as an ISP and so have to have the web interface open to the world. You should make use of the trusted IP settings to ensure you still have access even if there is a denial of service attack against the web interface. You should also set up access restrictions for users (see Section 4.1.4 for details).

12.2. Common settings

Most system services have the following common access control attributes.

Tip

You can verify whether the access control performs as intended using the diagnostic facility described in Section 13.1

Table 12.2. 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). |
| local-only | This normally defaults to <code>true</code> , but not in all cases. If <code>true</code> then access is only allowed from machines on IPs on the local subnet ^a . This restriction even applies if the address happens to be in the <code>allow</code> list. |
| 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 <code>local-only</code> is specified, the <code>allow</code> list allows access from addresses that are not local, if they are in the <code>allow</code> list. |
| log | The standard <code>log</code> , <code>log-error</code> , and <code>log-debug</code> settings can be used to specified levels of logging for the service. |

^aA *locally-attached* subnet is one which can be directly reached via one of the defined interfaces, i.e. is not accessed via a gateway, and not via an interface which has been marked `wan="true"`.

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.

12.3. 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 apart from access control. It is also possible to customise the colours of the user-interface (see Section 3.6.5).

12.3.1. Access control

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 password is correct.

By default, the FB6000 will only allow access from *local* interfaces. This is locked down by the `local-only` setting defaulting to `true`. If you change this, it will allow access from anywhere and you may want to set up IP ranges or groups in the `allow` setting to control access.

Note that a subnet can be marked as `wan` to indicate that even though directly connected, it is not considered *local*. This is mainly for cases where the external interface is a wide DHCP subnet spanning other users of the same ISP, and so should not be considered local.

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.

There are a number of security related headers with sensible defaults. These can be changed in the config. If you wish to remove a header simply make it an empty string to override the default.

12.3.1.1. 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. The *trusted* access list also has priority over `local-only` and `allow`, i.e. if the source IP is in the *trusted* access list, it is always allowed.

12.3.2. HTTPS access

The FireBrick provides a means to access the control pages using HTTPS rather than HTTP. When you first use a FireBrick, if you access using HTTPS to its IP address or `my.firebrick.uk` you will get a warning about the certificate being self signed. You can bypass the warning or use HTTP as you prefer, though HTTPS (even with a warning) prevents *passive* snooping, so is preferable. Ideally you want to set up HTTPS properly for your normal access to your FireBrick in the long term.

In most cases, all that is needed to manage HTTPS access it to ensure the FireBrick can be accessed via port 80 on a public IP address using a proper public hostname. Once you have done this, simply add the hostname to the `acme-hostname` field, and your email address in the `acme-terms-agreed-email` field, in the system config. This will cause an automatic ACME certificate issue using Let's Encrypt, adding private key pairs for the letsencrypt.org account and the domain and then adding the certificate and any intermediate certificates for HTTPS to work.

Note

By putting your email address in the `acme-terms-agreed-email` field you are indicating that you agree to the terms and conditions of the Certificate Authority being used. This may include publishing a list of certified domain names, and sending emails for changes of terms, etc.

ACME is an automated system for the issuing of X.509 certificates for HTTPS (and other) use. The FireBrick can work as an ACME client to obtain certificates and is preset to use *Let's Encrypt's* free certificate issuing service, and automatically renew certificates. You can change settings to use an alternate ACME server if you prefer though you may have to upload root certificates for the alternate CA. Contact support if you have any issues. Renewal happens automatically, but you can lock down when this happens (e.g. time of day) by setting an `acme-profile`.

Note

You do not need to allow port 80 (HTTP) access in your network, but you can lock down HTTP and HTTPS access using the `allow` and `trusted` settings as normal. By default, during the ACME update process, port 80 is allowed generally at a TCP level (bypassing the `allow` settings) but only for the special URL for ACME validation and only for the few seconds needed to validate your device. This means access to the config pages always respects the `allow` and `trusted` settings at all times, even if accepting TCP port 80 briefly for one specific ACME URL. This can be disabled, but may stop the automatic certificate renewal process. In addition, if a profile is set up as a control switch, and named as `fb-` followed by the certificate authority, e.g. `fb-letsencrypt.org` then this profile is activated at the start of the validation phase, and deactivated at the end. This can be used (with care) to allow specific firewall rules during certificate renewal.

Note

If you wish to load your own private keys, and allow ACME renewal of certificates, you can do so. The keys need to have a *name* in the same way as the automatically generated keys, i.e. one for the account with the CA, e.g. `letsencrypt.org` and one matching the hostname for the certificate name. If such keys exist then they are used instead of making new keys when requesting a certificate. At present these must be RSA keys. If you do not load your own keys, the FireBrick makes keys internally. You can disable automatic key generation by setting `acme-keygen` false.

Note

The HTTPS server uses SNI to find the matching certificate against its common or subject alt name. This must therefore exactly match the name used to access the FireBrick or be a wildcard for the first level and match the rest, as per normal HTTPS certificates. If no match is found then HTTPS will not work. By default, if no certificate can be found for HTTPS, one is created, self signed (which will give a browser warning). This can be turned off in the web config settings for `self-sign`. Self signed certificates have limited life, are removed on reboot or expiry, and only a small number are retained in the certificate store at one time. If no SNI is provided, a self signed certificate matching the IP address literal is used on the assumption that this is what was used with the `https://` protocol. Obviously the automated ACME process creates, and renews, a verified CA signed certificate with the correct name, so is recommended. Once set up, you may wish to turn off self signed certificates, and possibly even HTTP access altogether.

By default access is permitted using HTTP and HTTPS (directing to HTTPS if an ACME certificate has been set up), but you can lock down to HTTP only, HTTPS only, or redirection from HTTP to HTTPS. It is recommended that HTTPS is used for security reasons. FireBrick HTTPS works with all modern browsers (e.g. IE 10 and above, chrome, firefox, safari). It uses TLS1.2 only with TLS1.3 planned when appropriate.

Tip

There are many things that could stop ACME working. The status page (`config/certificates`) shows the progress of the process. There is also a `log-acme-debug` setting to allow more detailed logging of the process. It is recommended you set `log-acme` to email you so that you are made aware of any problems automatically renewing certificates.

12.4. 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 17 and in the Appendix H.

12.4.1. 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"/>
```

12.5. 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 and PPPoE. In most cases you do not need to set the resolvers.

12.5.1. 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.

In addition, the DHCP server records the IP of the latest new automatically allocated address on any interface, and you can set `auto-dns-new` as a name (e.g. `new`) to be used for this IP address. This is ideal for finding the IP of a newly added device, and is particularly useful when adding new IoT devices one at a time so they can be configured via a web interface, for example. This is only set for a new dynamic IP allocation, not a renewed, or reused IP for same device previously using the IP.

Note

It is assumed that the `domain` used is either within your control or has been designated for local use and therefore that any additional record types (e.g. SVR records) for those hosts are within your control. If they do exist then these will be assumed to be intentional and thus the recursive lookup and caching process will occur as normal. If this is not desirable then Blocking DNS names can be used to hide the remainder of the zone.

12.5.2. 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.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.

This mechanism will respond to A, AAAA, PTR or ANY type requests. Those of other types will be blocked rather than sent upstream to avoid conflicting results from the override and the wider zone.

Note

You can leave the IP unset, and this will return the FireBrick's own IP (as used for `my.firebrick.uk`), but this can also return an IP based on the request if the match is for a `*.name` in the right format. For example `192-168-0-1.my.firebrick.uk` returns A record `192.168.0.1`.

12.5.3. 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`.

Note

The blocking rules are applied *after* local and automatic DNS have been taken into account. As a result they will not block those explicitly overridden domains, but can be used to hide other record types for an auto DHCP DNS domain if needed.

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.

12.6. 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 servers instead.

The NTP service allows the FireBrick to be set accurately and act as an NTP server to machines on your network. By default it will not serve time to non local IP addresses, but will, of course, handle the expected replies from the servers being use to set the FireBrick clock. The status pages show the NTP servers being used and NTP system status.

Generally no config is needed to set the clock and allow the FireBrick to be an NTP server. The config can be used to change to other local time zones and "daylight saving" settings if required.

12.7. 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 12.3.1 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).

12.8. RADIUS configuration

12.8.1. RADIUS server (platform RADIUS)

Chapter 16 provides details of how the platform RADIUS service can be used to steer incoming sessions from a carrier for L2TP.

12.8.2. RADIUS client

RADIUS is used for authentication and accounting for incoming L2TP connections. Chapter 16 provides details of how RADIUS is used for L2TP. Appendix F provides details of the specific AVPs used with RADIUS for L2TP.

12.8.2.1. RADIUS client settings

The system settings for a RADIUS client allow multiple different client settings to be created by name. L2TP uses RADIUS by default, and if not set then the first settings found are used. However, you can set a named RADIUS client setting to be used for each L2TP server setting. This then looks for the named client setting for accounting and/or authentication.

The corresponding RADIUS servers are queried for the authentication or account messages. Each client setting can list multiple servers. The servers listed in matching named entries (i.e. with same name) are considered. You can see the status of each RADIUS server in the Status/RADIUS menu. This includes the average response time, and the last 64 responses (good/bad).

The set of servers being considered are sorted primarily by the order of the config items they were derived from and secondarily by their previous responses. The least recently failed to respond for a given config item are listed first and then the fastest responding servers listed first. Only the last 64 responses of each server are taken into account. The first 5 servers are then considered for answering the RADIUS query. If fewer than 5 are available, then the list is repeated. This means there are always 5 servers attempts made, even if that is one server 5 times.

Each server is then given a timeout. The timeout is normally based on the *scale-timeout* multiplied by the average response time of that server. If there are no responses from this server a default of half a second will be used. This timeout is then clamped to limits calculated relative to the time of the first attempt given by the following equation: $n^2 m / 5^2$ where n is the attempt number (1-5) and m is the configured max or min timeout. This creates a sequence of requests to be sent to one or more RADIUS servers.

If, within the overall timeout, any of the servers respond then this is accepted. If none respond then all record a timeout.

To allow servers to recognise duplicate requests, each request in the sequence that is to the same server has the same content and ID. This allows the server to simply resend the previous reply if it was dropped.

In addition to these timeouts, it is also possible to set a maximum *queue* for the set of servers. This limits how many concurrent requests can be waiting.

Tip

If your RADIUS servers are struggling, then set the queue lower, e.g. 8. If the response times have a lot of jitter then consider setting the *scale-timeout* higher (the default is only 2, so try 3, 4, etc). For VoIP, you will want a very fast server to respond to authentication used for call routing. For accounting you may want to allow a longer scale and max timeout to ensure accounting requests are not lost.

12.8.2.2. Server blacklisting

For each request to a server, a log is made of whether there was a response or a timeout, and this is recorded and shown on the server status page. This logs the last 64 requests.

If all of the last 64 requests have failed then the server is blacklisted. This stops it being considered when there are other servers to consider. If all appropriate configured servers are blacklisted then the FireBrick will attempt to use the blacklisted servers in a round-robin manner.

It is quite possible for a server to *go away* when there are no current RADIUS requests, or even *come back* when not being used for current requests. To allow for this the FireBrick sends *status-server* requests to the server periodically, and records the most recent 64 responses to these requests. This means a blacklisted server will be recorded as usable again once it starts answering such requests. It also means a server can become blacklisted if a server stops responding to such requests without actually losing any real RADIUS requests.

If a server has never answered a *status-server* request, it is assumed not to be enabled. We strongly recommend enabling this feature on your RADIUS servers. If not enabled then servers are provided with a dummy *good response* periodically to take them out of *blacklisted* status and allow them to be tried occasionally in case they are now working again.

Chapter 13. 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 12

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.

13.1. Access check

For each network service implemented by the FB6000 (see Chapter 12), 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.
```

13.2. 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 set up 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`.

13.2.1. Dump parameters

Table 13.1 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 13.1. 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 |
| l2tp | L2TP session | Where L2TP is available, one or more sessions, using the full hex accounting ID, can be specified, e.g. l2tp=002132D94AE297DFF51E01 or you can use l2tp=* followed by a calling line ID - this sets up logging for a session based on calling line id when it next connects. |
| 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. |
| party | IPs (2-off) | IPs, blocks and ranges can be specified to filter packets. Filling in |

| | | |
|------|--------------------------|--|
| | | one field will capture any traffic to/from the given host(s). Filling in a second field will only capture traffic between the specified hosts. |
| self | Include this TCP session | By default any TCP traffic to or from the IP and ports of the TCP session used for the dump is excluded. You can opt to include it, but that is usually a bad idea for obvious reasons as you may be dumping the packet dump itself. Note that if dumping tunnel wrappers and dumping via that tunnel then the dump traffic is not normally detected/excluded. |

13.2.2. 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 12.3).
- 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.

13.2.3. IP address matching

You may optionally specify up to 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.

13.2.4. 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 13.2. 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.

13.2.5. 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.

13.2.6. Using the web interface

The web form is accessed by selecting the "Packet dump" item under the "Diagnostics" main-menu item. Set up the dump parameters with reference to Table 13.1 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.

13.2.7. 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 13.1. Then you retrieve 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[¶meter_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 capture is currently running.

13.2.7.1. 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&timeout=300&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 14. 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.

14.1. 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 operating 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.

14.2. 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.

14.2.1. 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.

14.2.2. 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.

A VRRP setting can have a profile. Unlike most profiles, if the profile is off then VRRP continues to work, but works with a priority forced to a low priority (usually 1, but can be configured). This means the VRRP will be backup but this depends on the setting for low-priority and what priority other devices are set to.

14.3. 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.

14.4. VRRP versions

14.4.1. 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).

14.4.2. 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.

14.5. 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 15. BGP

15.1. What is BGP?

BGP (Border Gateway Protocol) is the protocol used between ISPs to advise *peers* of routes that are available. Each ISP tells its peers the routes it can *see*, being the routes it knows itself and those that it has been advised by other peers.

In an ideal world everyone would tell everyone else the routes they can see; there would be almost no configuration needed; all packets would find the best route across the Internet automatically. To some extent this is what happens between major transit providers in the Internet backbone.

In practice things are not that simple and you will have some specific relationships with peers when using BGP. For most people there will be *transit providers* with which you peer. You can receive a *full table* from each transit provider, containing routes to everywhere in the Internet via that provider. The FB6000 can then decide which provider has the best route to any destination. You can advise the transit provider of your own routes for your own network so that they can route to you, and they tell their peers that they can route to you via that provider. This only works if you have IP address space of your own that you can announce to the world - unless you are an ISP then this is not commonly the case.

Even though IPv4 address space has already run out, it is possible to obtain IPv6 PI address space and an AS number to announce your own IPv6 addresses to multiple providers for extra resilience.

You can use BGP purely as an internal routing protocol to ensure parts of your network know how to route to other parts of your network, and can dynamically reroute via other links when necessary.

In most cases, unless you are an ISP of some sort, you are not likely to need BGP.

15.2. BGP Setup

15.2.1. Overview

The FB6000 series router provides BGP routing capabilities. The aim of the design is to make configuration simple for a small ISP or corporate BGP user - defining key types of BGP peer with pre-set rules to minimise mistakes.

Caution

Misconfiguring BGP can have a serious impact on the Internet as a whole. In most cases your transit providers will have necessary filtering in place to protect from mistakes, but that is not always the case. If you are an ISP and connect to peering points you can cause havoc locally, or even internationally, by misconfiguring your BGP. Take care and get professional advice if you are unsure.

Note

The FireBrick has some defaults and specific *types* of peer that set some of these defaults. These are designed so that beginners do not make the more obvious mistakes with BGP and can be a good way of working. However, if you are already familiar with BGP you may wish to set all peers to type *peer* and define your own community tags and filters rather than using the defaults provided by FireBrick peer types.

15.2.2. Standards

The key features supported are:-

- Simple pre-set configurations for typical ISP/corporate setup

- RFC4271 Standard BGP capable of handling multiple full internet routing tables
- RFC4893 32 bit AS number handling
- RFC2858 Multi protocol handling of IPv6
- RFC1997 Community tagging, with in-built support for well-known communities
- RFC2385 TCP MD5 protection
- RFC2796 Route reflector peers
- RFC3392 Capabilities negotiation
- RFC3065 Confederation peers
- RFC5082 TTL Security
- Multiple independent routing tables allowing independent BGP operations
- Multiple AS operation

15.2.3. Simple example setup

A typical installation may have *transit* connections from which a complete internet routing table is received, *peers* which provide their own routes only, *internal* peers making an IBGP mesh, *customers* to which transit is provided and customer routes may be accepted. To make this setup simple the <peer> definition contains a *type* attribute. This allows simple BGP configuration such as:-

```
<bgp as="12345">
  <peer as="666" name="transit1" type="transit" ip="1.2.3.4"/>
  <peer as="777" name="transit2" type="transit" ip="2.3.4.5 2.3.4.6"/>
  <peer type="internal" ip="5.6.7.8"/>
</bgp>
```

This example has two transit providers, the second of which is actually two peer IP addresses, and one internal connection. Note that the peer AS is optional and unnecessary on internal type as it has to match ours.

The exact elements that apply are defined in the XML/XSD documentation for your software release.

15.2.4. Peer type

The *type* attribute controls some of the behaviour of the session and some of the default settings as follows.

Table 15.1. Peer types

| Type | Meaning |
|----------|--|
| normal | Normal mode, no special treatment. Follows normal BGP rules. |
| transit | Used when talking to a transit provider, or a peer that provides more than just their own routes. Peers only with different AS. The community no-export is added to imported routes unless explicitly de-tagged |
| peer | Used when talking to a peer providing only their own routes. Peers only with different AS. The community no-export is added to imported routes unless explicitly de-tagged <i>allow-only-their-as</i> defaults to true |
| customer | Used when talking to a customer's routers, expecting transit feed and providing their own routes Peers only with different AS <i>allow-only-their-as</i> defaults to true <i>allow-export</i> |

| | |
|-------------|--|
| | defaults to true The community no-export is added to exported routes unless explicitly de-tagged |
| internal | For IBGP links. Peers only with same AS <i>allow-own-as</i> defaults to true |
| reflector | For IBGP links that are a route-reflector. Route reflector rules apply Peers only with same AS <i>allow-own-as</i> defaults to true |
| confederate | For EBGP that is part of a confederation. Confederation rules apply Peers only with different AS |
| ixp | Must be EBGP, and sets default of no-fib and not add-own-as. Routes from this peer are marked as IXP routes which affects filtering on route announcements. Only announced on EBGP not IBGP. |

15.2.5. Route filtering

Each peer has a set of import and export rules which are applied to routes that are imported or exported from the peer. There are also named *bgp-filter* which can be used as *import-filters* or *export-filters*.

The objects *import* and *export* work in exactly the same way, checking the routes imported or exported against a set of rules and then possibly making changes to the attributes of the routes or even choosing to discard the route.

Each of these objects contain:-

- Cosmetic attributes such as *name*, *comment*, and *source*.
- Route matching attributes allowing specific routes to be selected
- Action attributes defining changes to the route
- A continuation attribute *stop* defining if the matching stops at this rule (default) or continues to check further rules

The rules are considered in order. The first rule to match all of the matching attributes is used. If no rules match then the default actions from the import/export object are used.

In addition, the top level import/export has a prefix list. If present then this will limit the prefixes processed at a top level, dropping any that do not match the list without even considering the rules.

15.2.5.1. Matching attributes

The actual attributes are listed in the XML/XSD documentation for the software version. The main ones are:-

- A list of prefixes filters defining which prefixes to match
- It is also possible to match by checking presence or absence of specific community tags.
- It is also possible to match by checking presence of specific AS in the AS-Path, or check the last AS in the path (originator).

You can have a rule with no matching attribute which will always be applied, but this is generally pointless as no later rules will be considered. If you want to define defaults then set them in the top level import/export object.

15.2.5.2. Action attributes

The actual attributes are listed in the XML/XSD documentation for the software version. The main ones are:-

- Adding specific community tags
- Removing specific community tags, including defaults added by the peer type.

- Dropping the route completely
- Changing the MED
- Changing the localpref
- Padding the AS list

The logic works by creating a set of actions that are applied, and these are based on top level settings in the peer (such as *set-med*) followed by the list of import or export named filters from which one matching action is picked, and then followed by the peers individual import and export rules from which one matching action is picked. The matching action causes each of the settings that are present to replace what is currently picked. E.g. if a MED is set in the top level and a named rule set the named rules set replaces the top level setting.

Important note - adding or removing community tags does not compound. For each setting (e.g. *tag*, *untag*, *med* and *localpref* and any added in future) the latest that was found after checking top level peer settings, the ordered list of filters, and then the local filters, is what applies. Multiple *tag* do not cause all the tags to be added, just the latest listed tags in the action. There are plans to improve this in the future to work step by step and even allow MED and localpref adjustments to compound.

You can have a rule with no action attributes. If matched then this means none of the actions are taken and communities, localpref, MED, etc., are all unchanged.

15.2.6. Well known community tags

Specific well known communities are supported natively. Some of these are set automatically based on peer type and can be explicitly removed using the *detag* action. These rules are automatically checked for exporting routes unless overridden on the peer attributes.

Table 15.2. Communities

| Community | Name | Meaning |
|-----------|--------------|---|
| FFFFFF01 | no-export | The route is not announced on any EBGp session (other than confederation or where <i>allow-export</i> is set). |
| FFFFFF02 | no-advertise | The route is not considered part of BGP. Whilst it is applied and used for routing internally it is not announced at all or considered to have been received for the purposes of BGP. |
| FFFFFF03 | local-as | The route is only advertised on IBGP (same AS) sessions. |
| FFFFFF04 | no-peer | This tag is passed on to peers but does not have any special meaning internally |

15.2.7. Announcing black hole routes

The FireBrick allows black hole routes to be defined using the the *blackhole* object. Routing for such addresses is simply dropped with no ICMP error. Such routes can be marked for BGP announcement just like any other routes.

It is also possible for L2TP announced routes to be marked as black hole routes using the *D* filter. If L2TP is marked to BGP announce such routes they are set to be *bgp="true"* rather than the *bgp* setting defined.

In order to ensure that your internal BGP network sees such routes as a black hole, and not simply as a route to the router than has the black hole defined (where the packets will be dropped), you can ensure all black hole routes are announced using a suitable community tag. In many cases an EBGp peer will even allow you to announce black hole routes to them with a suitable community tag.

The top level *bgp* object includes a *blackhole-community* attribute which can be set to a tag that is used to mark routes as a black hole within your network. Any route received on a BGP peer within that config object

which includes the specified community is treated as a black hole route. It is installed in the BGP routes and propagated as normal but it is internally set as forwarding to nowhere and packets dropped as a black hole.

Each *peer* object also has a *blackhole-community* tag. If set then this is added to any black hole routes announced. If not set, then black hole routes are not sent on EBGP links. On IBGP links, if not set, the *blackhole-community* from the parent *bgp* is added if present. Black hole routes are always announced on IBGP (subject to normal rules for announcement).

To use this, define a suitable *blackhole-community* for your network, such as *YourAS:666* and set in all *bgp* objects. For all EBGP peers, set the *peer* object *blackhole-community* attribute with the tag they expect for black hole routes.

It is unlikely you would want to announce a black hole route to an EBGP peer without an agreed tag as you will be drawing traffic from them only to be discarded. If you want to do this, you have to specify a *blackhole-community* to add, but this could simply be your own community tag for black holes.

15.2.8. Grey holes

Sometimes you may want to inject a blackhole route into your network, but not actually blackhole within your network, simply ensuring that the routes propagate via BGP until edge routers where they are announced to transit/peering as blackhole community tagged routes (as above).

For this reason, blackhole route objects can be tagged *no-fib* which creates the routing in the routing table but does not impact packet forwarding. By defining a *greyhole-community* on your *bgp* settings, this will be used for IBGP to pass the route around as a blackhole route but with *no-fib* set.

This is useful where injecting a black hole is needed, but you want to ensure internal routing to externally blackholed routes.

15.2.9. Announcing dead end routes

The top level *bgp* object includes a *dead-end-community* attribute which can be set to a tag that is used to mark routes as a dead end within your network. Any route received on a BGP peer within that config object which includes the specified community is treated as a dead end route. It is installed in the BGP routes and propagated as normal but it is internally set as forwarding to nowhere and icmp errors generated (rate limited as usual). Any route installed as *network* are announced with this community. Note, this is not set automatically on a *nowhere* route, allowing a route to be announced to get to this FireBrick to be propagated via IBGP.

The effect of this is that your network can include one (or more) source of top level *network* routes which, within your network, are installed as dead ends at each point. Without this these would be announced to your internal network so traffic is sent to the originating router and it has to handle all dead end traffic. Using this system you can ensure dead end traffic is handled at your borders instead.

15.2.10. Bad optional path attributes

The BGP specification is clear that receipt of a path attribute that we understand but is in some way wrong should cause the BGP session to be shut down. This has a problem if the attribute is one that is not known to intermediate routers in the internet which means a bad content is propagated to multiple routers on the internet and they will drop their session. This can cause a major problem in the internet.

To work around this *ignore-bad-optional-partial* is set to *true*, by default. The effect is that if a path attribute we understand is wrong, and it is optional, and the router that sent it to us did not understand or check it (*partial* bit is set), we ignore the specific route rather than dropping the whole BGP session.

15.2.11. <network> element

The network element defines a prefix that is to be announced by BGP by default, and tagged with any *dead-end-community*, but otherwise treated the same as a *nowhere* route.

Table 15.3. Network attributes

| Attribute | Meaning |
|-----------|--|
| ip | One or more prefixes to be announced |
| as-path | Optional AS path to be used as if we had received this prefix from another AS with this path |
| localpref | Applicable localpref to announce |
| bgp | The bgp mode, one of the well known community tags or <i>true</i> (the default) which is announced by BGP with no extra tags |

15.2.12. <route>, <subnet> and other elements

Subnet and route elements used for normal set-up of internal routing can be announced by BGP using the *bgp* attribute with the same values as the well known community tags, or with *true* meaning simply announce with no tags, and *false* meaning the same as *no-advertise*.

Many other objects in the configuration which can cause routes to be inserted have a *bgp* attribute which can be set to control whether the routes are announced, or not.

15.2.13. Route feasibility testing

The FB6000 has an aggressive route feasibility test that confirms not only routability of each next-hop but also that it is answering ARP/ND requests. Whenever a next-hop is infeasible then all routes using that next-hop are removed. When it becomes feasible the routes are re-applied. This goes beyond the normal BGP specification and minimises any risk of announcing a black hole route.

Note

There is an option relating to imported routes *reduce-recursion* which, when set, changes any received next hop to the peer address unless the next hop relates to a locally connected Ethernet subnet. This helps reduce the recursion involved, and is important in some cases for route reflectors if they pass recursive routes on to routers that do not handle BGP recursive routes properly (such as BIRD).

15.2.14. Status

The rows in the BGP status page are coloured grey when idle (or incoming), yellow when connecting, teal until the EORIB marker is received and green thereafter. Red connections are waiting for a backoff timer before retrying an outgoing connection.

Under some circumstances (for instance during graceful restart), the FB6000 will hold routes received from the peer for a time while renegotiation occurs, these routes are included in the inbound route count.

Note

Since held routes are included in the count, it is sometimes possible to prematurely hit the prefix limit when refreshing (for instance when changing filters in the config). This means that if a prefix limit is configured, fewer routes than expected may be picked up. However, the held routes will time out shortly and the count will drop, after which a refresh will pick up as many routes as the prefix limit allows. A refresh can be manually triggered from the BGP status page for an individual peer.

15.2.15. Diagnostics

The web control pages have diagnostics allowing routing to be shown, either for a specific target IP (finding the most specific route which applies), or for a specified prefix. This lists the routes that exist in order, and indicates if they are suppressed (e.g. route feasibility has removed the route). There are command line operations to show routing as well.

It is also possible to confirm what routes are imported from or exported to any peer.

The diagnostics also allow ping and traceroute which can be useful to confirm correct routing.

Routes that are rejected during import (either due to hitting the max prefix limit or due to configured filtering) are logged to the debug target on the corresponding peer.

15.2.16. Router startup and shutdown

On router shutdown/reboot (e.g. for software load) all established BGP sessions are closed cleanly. Before the sessions are closed all outgoing routes are announced with a lower priority (high MED, low localpref, prefix stuffed) and then a delay allows these to propagate. This is a configurable option per peer and the maximum delay of all active peers is used as the delay. Setting to zero will not do the low priority announcement. A special case of setting this delay to a negative value on a peer causes routes to be specifically withdrawn before the delay rather than announced low priority.

On startup, each peer can be configured with a startup delay which will stop BGP announcements being sent within a period of a reboot. This allows a FireBrick to connect BGP sessions and receive routes before announcing routes to peers. This allows a cleaner startup when used as a pair of BGP routers.

15.2.17. TTL security

The FireBrick supports RFC5082 standard TTL security. Simply setting `ttl-security="1"` on the peer settings causes all of the BGP control packets to have a TTL of 255 and expects all received packets to be TTL 255 as well.

You can configure multiple hops as well, setting `ttl-security="2"` for example still sends TTL 255 but accepts 254 or 255. This works up to 127.

You can also configure a non standard forced TTL mode by setting a negative TTL security (-1 to -128) which forces a specific TTL on sending packets but does not check received packets. For example, setting `ttl-security="-1"` causes a TTL of 1 on outgoing packets. This simulates the behaviour of some other routers in IBGP mode. Using -2, -3, etc, will simulate the behaviour of such routers in EBGP multi-hop mode. This is non standard as RFCs recommend a much higher TTL and BGP does not require TTLs to be set differently.

Without `ttl-security` set (or set to 0) the RFC recommended default TTL is used on all sent packets and not checked on received packets.

Chapter 16. Internet Service Providers

The FireBrick can be used by Internet Service Providers (ISPs) to provide Internet connectivity by acting as a gateway between a carrier network (e.g. Broadband or mobile carrier) and the Internet. This chapter covers the ISP use of a FireBrick including L2TP, and PPPoE.

L2TP can also be used on a smaller scale to create point to point tunnels.

16.1. Background

16.1.1. How it all began

Once upon a time end users would use a computer and a modem to dial a provider. The provider would have a modem connected to a server and this would allow simple text access to a computer system. This was then used to provide bulletin boards.

This moved on, and providers started to allow direct Internet Protocol (IP) access to end users. The modem would connect and the computer would authenticate and pass IP packets using protocols such as SLIP and PPP. This allowed the computer to authenticate and be allocated an IP address.

16.1.2. Point to Point Protocol

Point to Point Protocol (PPP) worked well and is still in use today in broadband access networks. The modem at the provider would connect to the provider's network and the Internet. Typically there would be one device, an Access Concentrator which connected IP on one side and modems on the other. The IPs would be fixed for each modem (so dynamic for the end user as depends which port they hit) and routing could be static to each Access Concentrator.

PPP is quite a simple protocol that allows packets to be marked with their type, but it also provides negotiation protocols for Link Control (LCP), authentication (CHAP and PAP), and IP level negotiations (IPCP and IPV6CP). Once negotiation is complete then IP packets can be passed using PPP.

As networks became more complex a separation of the Access Concentrator into a L2TP Access Concentrator (LAC) which has the modems, and the L2TP Network Server (LNS) was sensible. The LAC accepted the call on the modem and established a Layer 2 Tunnelling Protocol (L2TP) connection to the LNS. This allowed PPP to be passed from the end user computer to the LNS. The LNS is responsible for the PPP negotiations and passing IP packets to and from the Internet.

16.1.3. L2TP

L2TP provides a simple means for PPP packets to be passed over an IP network. It uses a small header and UDP to pass packets between the LAC and LNS.

Sometimes it became sensible for the LAC to decide to which LNS it should connect by some means. A good example is where a carrier with LACs will route connections to wholesale customers' LNSs. This would allow ISPs to make use of providers that have modems. This is actually the way it works on broadband access networks. For example, BT, O2, and TalkTalk have LACs in their network which pass L2TP to their ISP customers.

To achieve this, the LAC does some of the initial PPP negotiations. It handles the LCP and starts the authentication. It then establishes the L2TP connection passing these proxy details on to the LNS. The choice of LNS is done using the username, which is why it has to start the authentication. Typically a realm is included in the user name, using an @ and a string at the end of the username to steer the connection to the right LNS.

16.1.4. Broadband

In a typical broadband network we don't have dialup modems in the same sense. The modems are jumpered to the phone line at the exchange and are part of an Access Node, usually called a DSLAM or MSAN. This then passes PPP packets on to a Remote Access Server, usually called a BRAS. The link from DSLAM to BRAS is typically PPPoE. The BRAS acts as the LAC and connects to an ISP's LNSs.

PPPoE is PPP over Ethernet. Some access networks use DSL to carry PPP packets directly (PPPoA), and some use the ADSL as an Ethernet Bridge (PPPoE). There are access networks which provide Ethernet by some means to the end user equipment which then communicates via PPPoE to the BRAS. All of these work in much the same way at the BRAS as it sees PPPoE connections.

Typically the BRAS provides the initial proxy negotiation and then establishes an L2TP connection, after which it is no longer involved in any negotiation, but just passes on PPP packets each way.

16.1.5. RADIUS

Remote Access Dial Up Server is a system that allows the authentication decisions and allocation of IP addresses to be passed on to separate servers rather than being configured into the various equipment. RADIUS uses UDP to send a request to a server and send a reply back.

RADIUS is used within carrier networks so that the BRAS can check to where it is to send an L2TP connection. The RADIUS response can contain the tunnel details it needs, including the authentication within L2TP.

RADIUS is also used between a carrier and an ISP. The carrier will send a RADIUS request to the ISP asking the ISP for details of the LNS to which the connection is to be sent. This allows the ISP to steer sessions as they need.

Once the LNS gets the L2TP connection, RADIUS is used to obtain the IP address details to be assigned to the specific connection.

RADIUS is also used for accounting, to provide details of connections in progress and volumes of data transferred.

Appendix F provides details of the specific AVPs used with RADIUS for L2TP.

16.1.6. BGP

Once a connection is made to an LNS, the end user is assigned IP addresses. Obviously there is a need to ensure that the IP addresses are routed within the ISPs network to the correct LNS. OSPF and BGP are the main routing protocols used for this (though, back in dialup days, RIP and RIP2 were often used, and were a bit slow). OSPF is not ideal for this as it means the whole OSPF network tracking every connection of every user. The FireBrick supports use of BGP to announce connected IP addresses into an ISPs internal network as connections are made via L2TP.

16.2. Incoming L2TP connections

To allow a connection to the FireBrick you have to decide on a `hostname`. This is not a DNS hostname and is more like a *login* or *username*. It can be anything you like. You can pre-agree with your carrier the hostname they will use and the IP address of your LNS. When the connection arrives the protocol includes the hostname and a *secret* (i.e. a *password*). The hostname allows the FireBrick to check which connection details apply). The *secret* is used so that the FireBrick can confirm its identity, at present it is not checked on incoming connections by the FireBrick. You should use `allow` or other means to ensure connections are valid if needed.

The FireBrick can be configured with many hostnames, which would typically be used for different carriers to connect. You can also use the hostname to separate different types of connection - for example, in the UK, BT have 20CN IPStream, and 21CN WBC connections which typically need separate monitoring and traffic

shaping. You could even use the hostname to separate different grades of service, or, if the ISP is providing wholesale connections, for different ISP customers.

The incoming connection configuration includes the password, the RADIUS servers to use to validate the users, and various defaults that apply to the PPP connections. Most of these defaults can be set by the RADIUS server as well, but it can be useful to make the RADIUS configuration simple by the use of defaults in the FireBrick config.

Taking one step back, the choice of LNS and hostname that the carrier uses when sending the connection to the ISP can either be pre-configured, or more usefully it can be based on a RADIUS request (sometimes called *platform RADIUS*). This allows the ISP to decide on a per-connection basis the tunnel endpoint details and steer sessions. The FireBrick can act as a platform RADIUS server, answering all queries to steer sessions to the correct LNS and hostname.

16.3. The importance of CQM graphs

The FireBrick has *Constant Quality Monitoring*. When used with the FireBrick acting as an LNS the CQM graphs play an important role.

Per connection monitoring. Each connection is assigned a CQM graph name. This is normally set based on the circuit ID passed by the carrier, or if not present, the username used. A long graph name (over 20 characters) is reduced to a hash. The name can also be set by RADIUS response (Chargeable User Identity attribute). Each graph shows the send and receive throughput for each 100 seconds. The graph also shows the loss and latency with minimum, average, and maximum per 100 seconds. This is based on an LCP echo sent every second on every connection. The interval can be configured to be lower if you wish (either in the config or by RADIUS).

The per connection graphs also have a downlink speed setting. This is set based on the connection speed from L2TP connection. This can also be set in the RADIUS response. This limits the speed of traffic to the line. This is usually done so that the LNS is in control of the speed of the line as the FireBrick will drop larger packets before smaller packets, which helps VoIP and many other protocols work well even on a full link. The speed control can also be used to provide slower services.

In addition to the per-connection graphs, there is also an aggregate graph based on the incoming L2TP connection settings - e.g. typically for a whole carrier. This tracks the overall throughput for all of the lines. This is useful simply for reporting and tracking, but the aggregate graph can also have a speed setting. This allows rate limiting to meet commit levels with carriers, which can be very important where, for example, there is 100th percentile billing.

This also allows a *damping* setting to be used. Where the aggregate is hitting the limit, all lines within that aggregate are reduced in their shaper settings by a percentage to damp the overall throughput. The continued hitting of the aggregate increases the percentage level. Individual lines can be tagged high or low priority by RADIUS which affects the level of damping applied, and so allows three grades of service when an aggregate link is full. At each stage, aggregate and per line, the shaping still drops larger packets first making for a very effective way to manage overall traffic levels.

It is also possible to set a third level of aggregation, where each connection can be placed in a *group* which is, itself, another CQM graph. This can be useful for tracking and shaping at a per wholesale customer or customer grouping in some way. This additional level of aggregation is not adjusted by the RADIUS priority settings, and the lesser of the 2 is used to determine the damping to apply to a given line.

Damping is displayed on the aggregate graphs, both as a "drop" line along the bottom and as an overlaid series for the strength of the damping applied. Damping is *not* shown on the per connection graphs (other than in the resulting speed).

16.4. Authentication

Normally an incoming connection uses RADIUS to obtain details of the IP addresses to use. It is also possible for RADIUS to provide relay tunnel endpoint details to pass the connection on to another LNS.

In addition to RADIUS based authentication it is also possible to pre-set local authentication details based on circuit ID and/or username and password. This bypasses RADIUS, and can be used to handle individual lines or patterns of login - e.g. use of a `@realm` to steer to another LNS for a wholesale customer.

In an ISP scenario this is typically used for special cases, test lines, etc. The main use of this feature is for a corporate LNS handling direct point to point tunnels, e.g. from other offices or roaming users.

16.5. Accounting

RADIUS can also be used for accounting. This involves a RADIUS message at start and end of each connection, and also interim accounting updates.

Interim accounting is done on a *snapshot* basis. Normally this is hourly. On the hour, within a second, the details are recorded and then sent by RADIUS. The RADIUS updates may take many minutes to be sent and confirmed by a RADIUS server, but the timestamp on the messages is the hourly snapshot. This is useful where an ISP is charging differently at different times of day.

Interim updates can also be sent based on reaching a pre-set time or data usage level defined in the authentication RADIUS response.

16.6. RADIUS Control messages

The FireBrick also supports RADIUS control messages. These can be used to disconnect a connection, or to update the details of the connection including line speed, time or data limits, and routing (apart from the PPP endpoint and DNS).

The update can also include change of routing table. This can be useful to move an active connection into a walled garden and back without ever dropping the PPP connection itself. This can be useful for credit control systems.

16.7. PPPoE

In addition to working as a conventional LNS, the FireBrick can also be configured to operate as a PPPoE endpoint as a BRAS. The PPPoE connections appear as if they have arrived via L2TP, so can have local IP termination or be relayed via L2TP to another LNS. See the section Section 10.4 for more details.

16.8. Typical configuration

The FB6000 is very flexible, but a typical configuration for L2TP as an ISP connected to a carrier generally follows a standard setup. Some carriers need specific extra settings, and the FireBrick support team can provide examples if you require.

16.8.1. Interlink subnet

A carrier will normally have an interlink - this could be a dedicated port on the FB6000 or a VLAN perhaps via a suitable switch. In any case this is an `interface` in the configuration. Some carriers use a /30 IPv4 interlink per LNS, and some use a larger subnet covering several LNSs.

16.8.2. BGP with carrier

This interlink is usually used solely for the purpose of a BGP link to the carrier, and all other IPs used by the ISP or carrier are announced via that BGP connection. You may want to configure filters on the BGP connection to limit the prefixes accepted from the carrier or announced to the carrier.

An alternative approach is to configure the interlink interface on a separate routing table. The FB6000 can have separate routing tables which act as completely separate internets. Using a separate table means you do not have to worry about what prefixes are announced on the BGP link as they will only apply to that routing table.

Whilst we would recommend using a BGP connection like this, if the carrier does not handle BGP then you will need static routes. Using a separate routing table can make this much simpler as you can set a default route to the carrier gateway on the interlink subnet.

If using a separate routing table, you have to take care to correctly configure the routing table on the interface, BGP, RADIUS, L2TP and loopback configuration elements.

16.8.3. RADIUS session steering

We recommend using RADIUS session steering with the carrier, if they support it. Session steering means that the carrier sends a RADIUS request to the ISP before connecting each session by L2TP. The reply steers the connection to a specific LNS. The connection details include the target (*IP address*) which will be one of the FireBrick's address, and a pre-agreed *hostname* which identifies the tunnel level connection, along with a *secret* to authentication the connection. Obviously these details have to match what the FireBrick is expecting in its L2TP configuration. Session steering gives a lot of control to the ISP, and is ideal if you operate bonding connections where multiple links need to use the same LNS.

The carrier will typically expect you to have two RADIUS endpoints to which they can send requests. One for master and one for backup. Whilst the FB6000 will answer RADIUS on any of its IP addresses, we know some carriers have issues using the interlink IP addresses. We recommend you create two additional loopback addresses for session steering RADIUS.

These addresses are configured as a BGP announced loopback address. You can use MEDs to steer which IP is on which LNSs. If you have more than two LNSs you can ensure that the same IPs are announced from more than one LNS, and let BGP decide which LNS gets the RADIUS requests. RADIUS is a simple question and answer protocol so it does not matter which LNS gets the request.

The session steering configuration (Platform RADIUS) is very flexible. We suggest you use the same configuration on each LNS so that the replies are consistent regardless of where the request goes. The reply says where to connect the session (as well as hostname and password to use). The reply can be a single LNS or can be more than one reply with a priority tagging if the carrier supports this. The FB6000 can pick an LNS randomly from a set, or pick one based on a hash of the username, part of the username, or circuit ID. This can be useful when multiple lines are in a bonded arrangement where using the same prefix on a username ensures all of the connections are steered to the same LNS for bonding.

If you have a lot of LNSs we recommend an N+1 arrangement. E.g. if you have 4 LNSs you may set your RADIUS session steering to reply with one of three *active* LNSs as the first choice (perhaps using a hash) and the 4th (backup) LNS as a second choice. This keeps one LNS as a hot standby for failure and also allows maintenance on it, such as s/w updates. You can then change which of the there LNSs are *active* and either wait for lines to move when they reconnect or clear lines on the new *backup* LNS. This makes it easy to do rolling upgrades on s/w or other maintenance.

Session steering also allows specific configurations to be based on username, and circuit and so on, so allowing different responses for different carriers and different end users to be customised if necessary. It is also possible to send a copy of the session steering RADIUS to your own RADIUS server for logging.

16.8.4. L2TP endpoints

The FB6000 will accept L2TP connections on any of its IP addresses, but again we recommend allocating a loopback address or using the address from the LAN rather than the interlink address as we know some carriers cannot handle that.

Unlike RADIUS where any request can go to any LNS, the L2TP connections have a state, and so you will want the address to stay with the particular LNS. Do not have a loopback that floats between LNSs via BGP

as this would mean existing connections trying to talk to another LNS. It is better for the a failure to cause a clean timeout on the failed LNS and reconnect (via RADIUS session steering) than to have an active session's traffic go to a different LNS where the session IDs could match an existing session (unlikely, but possible).

The L2TP connection is matched to an incoming L2TP configuration. The RADIUS session steering can be used to specify the hostname and password that is sent so that the correct incoming configuration can be matched. This can select the specific RADIUS servers to use at the ISP for authorising the connection (though typically a single set of RADIUS servers is used for all connections). It can also specify defaults for DNS, PPP endpoint addresses and so on.

16.8.5. ISP RADIUS

Once the L2TP connection arrives you can use RADIUS in your own network to control the connection, accepting it or rejecting it, and defining IP addressing, DNS, traffic speeds, routing table, and much more. Appendix F provides details of the specific AVPs used with RADIUS for L2TP.

You would normally have more than one RADIUS server. You can set these in a priority order, for example a set of main servers and a set of backup. The FB6000 will find a config line for RADIUS based on the named RADIUS server in the L2TP incoming configuration, or pick any if this is not set. It checks these in order. Each RADIUS configuration can have multiple servers.

Having picked a set of RADIUS servers to try they are sorted based on their previous response time and reliability. The requests are then sent to servers in order, allowing enough time for a response based on previous performance. There are settings to fine tune these timings. Once a response is received then the L2TP connection can proceed.

The same process is followed for RADIUS accounting. Each config can say if it is used for authentication or accounting or both.

Chapter 17. 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 12.4.

The CLI is also available via the serial interface.

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
tracertool <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 H 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 $M = 32 - \text{prefix_length}$

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 $M = 32 - \text{prefix_length}$

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. The *subnet* status page shows the MAC addresses currently in use on the Ethernet interfaces.

B.1. Multiple MAC addresses?

A MAC address does have to be unique on an Ethernet LAN segment, and typically a device will have one MAC address, or one for each physical interface, preset by the network card in use. However, the FireBrick makes use of multiple MAC addresses. There are two key reasons for this.

- The FireBrick can operate as a DHCP client device multiple times on the same LAN segment, obtaining several separate IP addresses. This is useful on some cable modem type installations where multiple IPs are only available if the FireBrick appears to be multiple devices at once. Whilst DHCP theoretically does not need separate MAC addresses, experience suggests this is by far the most practical approach. If you have more than one DHCP client subnet in your configuration they will automatically get separate MAC addresses.
- In theory the scope of a MAC address is a single LAN segment. The fact that they are globally unique is simply to avoid any clashes on a LAN segment. However, once again, practical experience shows that some network devices and some network switches do not handle the concept of the same MAC address appearing on different ports or VLANs within the network. This can lead to broken networks or traffic leaks between VLANs, neither of which is good. For this reason the FireBrick uses distinct MAC addresses on each interface.

B.1. Using the same MAC address

There are cases where it is sensible or required to use the same MAC address for more than one thing. For a start, the FireBrick does not have unlimited MAC addresses, but there are other reasons, for example:-

- Distinct subnets on the same LAN segment do not cause any switch/MAC issues as the FireBrick appears to simply be one device on the LAN segment with multiple IPs. This is quite a normal configuration for network devices. In these cases the FireBrick can use the same MAC address for multiple IPs on the same LAN segment.
- There can be MAC restrictions on some devices - this is mainly at the ISP level where peering points and network connections may be set up with limited MAC addresses. In such cases any packet with a different MAC address seen on a port can cause the port to shut down, or the additional MAC addresses to be blocked. For this reason there are cases where multiple subnets need to be restricted to exactly one MAC address.

Tip

The *interface* settings in the configuration have a `restrict-mac` setting which, when set to `true` causes the same MAC to be used for all subnets and operations on that specific interface (port group / VLAN combination).

B.2. Changing MAC address

There is no reason for any network device to maintain the same MAC address for ever. It is normal for the MAC address to change if the network card is changed on a PC, for example.

However, it is inconvenient if MAC addresses change simply because a device is power cycled or a new configuration is loaded. This can cause delays accessing the device if other devices have the MAC cached. It is also a serious problem for ISP links as above where ports are locked to only accept one MAC.

The way the FireBrick manages MAC addresses is designed to be a bit *sticky* so that a config change will not usually cause a MAC address assigned to a subnet or interface to change.

B.2. How the FireBrick allocates MAC addresses

To meet these requirements the FireBrick allocates MAC addresses to specific aspects of the configuration when it is loaded, and stores this separately in persistent data. If the config is then changed, such as changing the order of interface definitions, then the allocated MAC stays with the config object based on some key aspect (such as port group and VLAN tag for interfaces, or IP for subnets).

B.2.1. Interface

Each interface object is allocated a MAC, keyed by the port group and VLAN tag of the interface. This is used for dynamic IPv6 allocation on the interface using router announcements (RA) and any other interface specific uses that are not related to a subnet.

B.2.2. Subnet

Each subnet object is allocated a MAC, which is used for all of the IPs listed in that subnet object. This allows many IPs to have the same MAC by listing them in the same subnet object. The MAC allocation is keyed on the port group and VLAN tag and the first listed IP address in the subnet. If a later subnet has the same first IP listed then this is allocated a separate MAC (i.e. the key for the MAC is also based on which instance of this specific first IP it is, 1st, 2nd, 3rd, within the interface).

DCHP client subnets work in much the same way - they are based on the port group and VLAN tag and which instance of DHCP client they are (1st, 2nd, 3rd, etc) within the interface. The special case for DHCP clients is that the first of these within an interface is given the same MAC as the interface itself.

B.2.3. PPPoE

Each PPPoE object is given a MAC. This is keyed on the port group and VLAN and works in the same way as if it was a DHCP client subnet in a corresponding interface. i.e. where there is an interface with same port group and VLAN the PPPoE object gets the interface MAC.

B.2.4. Running out of MACs

The allocations are recorded in persistent data, so if an object is removed from the config and later put back it should get the same MAC address. If however there are not enough MAC addresses when loading a config,

then previous assignments are re-used. If there are too many interface, subnet and ppp objects within the config to allocate MAC addresses (even reusing old allocations) then an error is given and the config cannot be loaded.

Tip

Using `restrict-mac` on an interface restricts that interface (port group/VLAN) to only use one MAC and not one per subnet.

B.3. Forcing particular MAC addresses

On rare occasions, it may be necessary or desirable to use specific MAC addresses for particular connections. In these cases, you may configure `mac-suffix` on a subnet, interface or PPPoE object. This allows you to set the end byte(s) of the MAC to determine which MAC address is used from within the FB6000's block of allocated MACs.

It is also possible to change the base MAC address and hence change the entire block of allocated MACs. This can be done by setting `spoof-mac` under "System settings". This should not be necessary under normal circumstances but if you use this feature, make sure that the MAC addresses do not clash with other devices on any LAN segments to which the FB6000 is connected.

Caution

Be careful when changing MAC addresses, as this can cause disruption. As stated above, it is generally desirable for MAC addresses to be consistent. Connections will drop because it can take time for other network equipment to become aware of the changes. Some equipment (often at the ISP level) does not expect MAC addresses to change at all.

Because MAC suffixes are stored in persistent data (separately from the config), changes to MACs will persist even if the `mac-suffix` is later removed. This should minimise unnecessary changes to suffixes, but this means that even a 'Test' save of the config can cause permanent changes. Also note that if you are explicitly setting a MAC suffix that is already in use on a different subnet, then that subnet will be assigned a new MAC.

B.4. MAC address on label

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



The x in the MAC address shows that that point could be any value 0 to F.

B.5. Using with a DHCP server

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. Scripted access

The web interface for the FireBrick is mainly intended to be used from a web browser, however, there are a number of cases where the use from scripts or tools can be useful. This appendix covers some of the ways the FireBrick web interface can be used. Some of these are already mentioned in other sections, such as customised CQM archiving URLs, and so not repeated here. We also do not cover loading and saving a new config.

C.1. Tools

There are a great many tools for accessing web pages, in a variety of languages. Please feel free to use whichever tools you are happy with.

In this section we suggest the use of the linux command line tool `curl`, and all of the examples are based on this.

C.2. Access control

You will need to consider access control carefully if using scripts to access the FireBrick. In particular you need to ensure that someone that has access to the scripts, or simply a copy of the script, could not gain unfettered access to your FireBrick(s).

C.2.1. Username and password

You will probably want to create a separate user for the script access, with a separate password. You can then make use of this with a command like `curl` using the `--user` argument in the script.

Note

We do not recommend simply hard coding the password, and it may be better to have passwords stored in a database of some sort. This will depend on your own security procedures.

C.2.2. OTP

You could, if you have the tools to manage OTP codes, make use of an OTP seed for your script user. Remember that the authentication system allows the OTP code to be pre-pended to the password, e.g. `--user name:123456password`. This is recommended if using HTTP access which can be snooped upon.

C.2.3. Allow list

We strongly recommend you have additional allow list access controls on your script user to lock down to machines which will be accessing the FireBrick. You may even want different users for different scripts, machines running scripts, and FireBricks depending on your own security policies.

C.2.4. Allowed access

It is likely that scripts, as described in this appendix, will not need to access or change the configuration on your FireBricks, and so you can lock down access for the user to restrict access to the config.

Note

If you do have scripts that update configuration, you may want to use a different user for security reasons.

C.3. XML data for common functions

When using the web interface you will encounter a number of cases where there is an XML link shown. For example, in *Status/Subnets*. This has an XML link to `/status/subnets/xml`.

This is an example of one of the many places that an XML version of some user data can be accessed. We have not listed all of these in the manual as you can see them in the web pages that include the XML link.

C.4. XML data from diagnostics and tests

Many of the diagnostic tools also include an XML link or check box, for example *Diagnostics/Ping*.

C.4.1. Cross site scripting security

When using the web page, the page expects that you will post back a special code that was included as a hidden field. This is to prevent the use of cross site scripting where a form on a separate web page will attempt to post to your FireBrick with arguments that do something on your FireBrick, hoping you happen to be logged in to the FireBrick still on your browser.

However, where an Authentication header is used, such as the `--user` option on `curl` this requirement is dropped, and so you can make scripts work with a single `curl` command with authentication and arguments.

C.4.2. Arguments to scripts

Most tools (e.g. *ping*) require some arguments. The simplest way to identify these is to see what is in the form on the web site. In some cases the field names are actually taken from the command line, which means that in some cases the field name is just a number. For example, *ping* needs 1 as a field with the IP address to ping, and count as the ping count. Typically all arguments are in fact optional, so do not need to be included.

The arguments need to be supplied as if sent from a form. In most cases these can be passed as a query string style, or as posted form data.

For example:-

```
curl http://my.firebrick.uk/diag/ping/p --user adrian:237426 --data 1=8.8.8.8 --data xml
```

The result being:-

```
<?xml version="1.0" encoding="UTF-8"?>
<status xmlns="http://firebrick.ltd.uk/xml/statusv1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://firebrick.ltd.uk/xml/statusv1 http://
firebrick.ltd.uk/xml/statusv1.xsd"
  firmware-version="TEST Abbotscliffe (V2.00.010 2023-09-18T08:29:31)"
  generated="2023-03-10T13:50:10Z">
<note>Pinging 8.8.8.8 from 90.155.42.98</note><ping>
<ping number='1' time='3.848' ip='8.8.8.8' note='Reply' name='google-public-dns-
a.google.com' />
<ping number='2' time='3.985' ip='8.8.8.8' note='Reply' name='google-public-dns-
a.google.com' />
<ping number='3' time='3.724' ip='8.8.8.8' note='Reply' name='google-public-dns-
a.google.com' />
<ping number='4' time='3.848' ip='8.8.8.8' note='Reply' name='google-public-dns-
a.google.com' />
<ping number='5' time='3.911' ip='8.8.8.8' note='Reply' name='google-public-dns-
a.google.com' />
<ping number='6' time='4.298' ip='8.8.8.8' note='Reply' name='google-public-dns-
a.google.com' />
<ping number='7' time='3.839' ip='8.8.8.8' note='Reply' name='google-public-dns-
a.google.com' />
```

```
<ping number='8' time='3.748' ip='8.8.8.8' note='Reply' name='google-public-dns-
a.google.com' />
<ping number='9' time='3.796' ip='8.8.8.8' note='Reply' name='google-public-dns-
a.google.com' />
<ping number='10' time='3.722' ip='8.8.8.8' note='Reply' name='google-public-dns-
a.google.com' />
<summary sent='10' received='10' loss='0.00' minimum='3.722' average='3.871'
maximum='4.298' />
</ping>
</status>
```

Note

The form will typically include an XML check box, hence the use of `--data xml` in the example so as to include this as a field that is sent in the query string.

C.5. Special URLs

A number of special URLs exist purely for script use. These can be accessed under `/config/` as follows :-

Table C.1. Special URLs

| URL | Type | Function |
|------------------|------|--|
| reboot | HTML | Simple reboot |
| reboot-when-free | HTML | Reboot when free |
| reboot-cancel | HTML | Cancel the reboot when free if possible |
| reboot-hard | HTML | Forced hard reboot now |
| capability | XML | Return XML capability |
| schema | XML | Return XML that is the internal format config edit schema |
| timestamp | Text | Timestamp of config |
| uptime | Text | Uptime |
| ip | Text | Your IP address as seen by the FireBrick |
| version | Text | Current software version |
| et | Text | Current software version, or an HTTP level error with text explaining why, such as Upgrade required. |

C.6. Web sockets

There are some web socket feeds available, generally these are used internally for config edit, status page, and logging, etc. However, we may add additional specific URLs for script use in due course. The general principle is that the websocket feed provides a stream of JSON objects in real time.

The status page uses `/status/` but an additional port usage page `/status/stats` provides per second byte counts for all ports.

Note that you can define a `js-url` to be loaded at the end of pages to make use of these within the web user interface. This only works when logged in and from a trusted IP address, and not on pages where passwords can be entered.

Appendix D. 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 segregate 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 E. Supported L2TP Attribute/Value Pairs

This appendix details the L2TP protocol messages supported, and the attribute/value pairs (AVPs) which are sent and expected for each message.

E.1. Start-Control-Connection-Request

Table E.1. SCCRQ

| AVP | No. | Incoming | Outgoing |
|----------------------|-----|---|------------------------------|
| Message Type | 0 | Value 1 | Value 1 |
| Protocol Version | 2 | Mandatory, value 1 expected | Value 1 |
| Framing Capabilities | 3 | Ignored | Value 3 |
| Bearer Capabilities | 4 | Ignored | Not sent |
| Tie Breaker | 5 | Ignored as FireBrick only accepts connections for inbound calls | Not sent |
| Firmware Revision | 6 | Ignored | FireBrick s/w version string |
| Host Name | 7 | Used to select which incoming L2TP configuration applies. | As per config/RADIUS request |
| Vendor Name | 8 | Ignored | FireBrick Ltd |
| Assigned Tunnel ID | 9 | Mandatory | Mandatory, our tunnel ID |
| Receive Window Size | 10 | Accepted, assumed 4 if not present or less than 4 is specified | Value 4 |
| Challenge | 11 | Accepted if a configured secret is defined, a response is sent in the SCCRP | Not sent at present |

E.2. Start-Control-Connection-Reply

Table E.2. SCCRP

| AVP | No. | Incoming | Outgoing |
|----------------------|-----|---|---------------------------------|
| Message Type | 0 | Value 2 | Value 2 |
| Protocol Version | 2 | Value 1 expected | Value 1 |
| Framing Capabilities | 3 | Ignored | Value 3 |
| Bearer Capabilities | 4 | Ignored | Not sent |
| Firmware Revision | 6 | Ignored | FireBrick s/w ID number |
| Host Name | 7 | Logged as hostname for tunnel | Configured hostname, if defined |
| Vendor Name | 8 | Ignored | FireBrick Ltd |
| Assigned Tunnel ID | 9 | Expected as far end ID | Mandatory, our tunnel ID |
| Receive Window Size | 10 | Accepted, assumed 4 if not present or less than 4 | Not sent, assume 4 |

| | | | |
|--------------------|----|---|--|
| Challenge | 11 | Accepted if a configured secret is defined, a response is sent in the SCCCN | Not sent at present |
| Challenge Response | 13 | Not expected at present | Sent if SCCRQ contained a challenge and we have a secret defined |

E.3. Start-Control-Connection-Connected

Table E.3. SCCCN

| AVP | No. | Incoming | Outgoing |
|--------------------|-----|--------------|------------------------|
| Message Type | 0 | Value 3 | Value 3 |
| Challenge Response | 13 | Not expected | Sent if was challenged |

E.4. Stop-Control-Connection-Notification

Table E.4. StopCCN

| AVP | No. | Incoming | Outgoing |
|--------------------|-----|--------------------|--------------------------------------|
| Message Type | 0 | Value 4 | Value 4 |
| Result Code | 1 | Ignored (logged) | Sent as appropriate for tunnel close |
| Assigned Tunnel ID | 9 | Expected, see note | Sent if a tunnel has been allocated |

Note that a StopCCN may not have a zero tunnel ID in the header. If this is the case the source IP, port and assigned tunnel are used to identify the tunnel.

If an unknown tunnel ID is received on any any incoming packet a StopCCN is generated (once per 10 seconds) with header tunnel ID 0 and specified assigned tunnel ID.

E.5. Hello

Table E.5. HELLO

| AVP | No. | Incoming | Outgoing |
|--------------|-----|----------|----------|
| Message Type | 0 | Value 6 | Value 6 |

Always responded to. Sent periodically if no other messages sent.

E.6. Incoming-Call-Request

Table E.6. ICRQ

| AVP | No. | Incoming | Outgoing |
|---------------------|-----|--|---------------------------|
| Message Type | 0 | Value 10 | Value 10 |
| Assigned Session ID | 14 | Mandatory | Mandatory, our session ID |
| Call Serial Number | 15 | Accepted and passed on if relaying | Passed on incoming value |
| Bearer Type | 18 | Ignored | Not sent |
| Called Number | 21 | Accepted, used in RADIUS and passed on if relaying | Passed on incoming value |

| | | | |
|---------------------|----|--|--------------------------|
| Calling Number | 22 | Accepted, used in RADIUS and passed on if relaying | Passed on incoming value |
| Sub-Address | 23 | Ignored | Not sent |
| Physical Channel ID | 25 | Ignored | Not sent |

E.7. Incoming-Call-Reply

Table E.7. ICRP

| AVP | No. | Incoming | Outgoing |
|---------------------|-----|-----------|---------------------------------------|
| Message Type | 0 | Value 11 | Value 11 |
| Assigned Session ID | 14 | Mandatory | Mandatory |
| Rx Connect Speed | 38 | Accepted | Sent when <i>advise-speed</i> is set. |

Note

Sending and receiving *Rx Connect Speed* in an *ICRP* message is a FireBrick specific extension. It is useful when the tunnel partner acting as the LNS is in a better position to determine the speed of the connection than the LAC.

E.8. Incoming-Call-Connected

Table E.8. ICCN

| AVP | No. | Incoming | Outgoing |
|------------------------------|-----|--|--------------------------|
| Message Type | 0 | Value 12 | Value 12 |
| Framing Type | 19 | Ignored | 1 |
| Tx Connect Speed | 24 | Accepted, used in RADIUS and passed on if relaying | Passed on incoming value |
| Initial Received LCP CONFREQ | 26 | Ignored | Not sent |
| Last Sent LCP CONFREQ | 27 | Accepted, used in RADIUS and passed on if relaying | Passed on incoming value |
| Last Received LCP CONFREQ | 28 | Accepted, used in RADIUS and passed on if relaying | Passed on incoming value |
| Proxy Authen Type | 29 | Accepted, used in RADIUS and passed on if relaying | Passed on incoming value |
| Proxy Authen Name | 30 | Accepted, used in RADIUS and passed on if relaying | Passed on incoming value |
| Proxy Authen Challenge | 31 | Accepted, used in RADIUS and passed on if relaying | Passed on incoming value |
| Proxy Authen ID | 32 | Accepted, used in RADIUS and passed on if relaying | Passed on incoming value |
| Proxy Authen Response | 33 | Accepted, used in RADIUS and passed on if relaying | Passed on incoming value |
| Private Group ID | 37 | Ignored | Not sent |
| Rx Connect Speed | 38 | Accepted, used in RADIUS and passed on if relaying | Passed on incoming value |

| | | | |
|---------------------|----|----------------------|----------|
| Sequencing Required | 39 | Accepted on honoured | Not sent |
|---------------------|----|----------------------|----------|

E.9. Outgoing-Call-Request

Table E.9. OCRQ

| AVP | No. | Incoming | Outgoing |
|--------------|-----|----------|----------|
| Message Type | 0 | Value 7 | Value 7 |

Not supported, ignored.

E.10. Outgoing-Call-Reply

Table E.10. OCRP

| AVP | No. | Incoming | Outgoing |
|--------------|-----|----------|----------|
| Message Type | 0 | Value 8 | Value 8 |

Not supported, ignored.

E.11. Outgoing-Call-Connected

Table E.11. OCCN

| AVP | No. | Incoming | Outgoing |
|--------------|-----|----------|----------|
| Message Type | 0 | Value 9 | Value 9 |

Not supported, ignored.

E.12. Call-Disconnect-Notify

Table E.12. CDN

| AVP | No. | Incoming | Outgoing |
|---------------------|-----|--------------------|--------------------------------------|
| Message Type | 0 | Value 14 | Value 14 |
| Result Code | 1 | Ignored (logged) | Sent as appropriate for tunnel close |
| Q.931 Cause Code | 12 | Ignored | Not sent |
| Assigned Session ID | 14 | Expected, see note | Sent if assigned |

Note that a CDN may have a zero session ID in the header. If this is the case the tunnel ID and assigned session ID are used to identify the session.

If an unknown session ID on a known tunnel ID is received on any any incoming packet a CDN is generated with header session ID 0 and specified assigned session ID.

E.13. WAN-Error-Notify

Table E.13. WEN

| AVP | No. | Incoming | Outgoing |
|-----|-----|----------|----------|
|-----|-----|----------|----------|

| | | | |
|--------------|---|----------|----------|
| Message Type | 0 | Value 15 | Value 15 |
|--------------|---|----------|----------|

Not supported, ignored.

E.14. Set-Link-Info

Table E.14. SLI

| AVP | No. | Incoming | Outgoing |
|--------------|-----|----------|----------|
| Message Type | 0 | Value 16 | Value 16 |

Not supported, ignored.

E.15. Notes

E.15.1. BT specific notes

The L2TP and PPP specifications are clear that the HDLC framing bytes are not sent or received within the L2TP packet. However, BT send type bytes (FF03) on the start of all PPP frames. This is silently discarded. Also, BT will not process packets if these type bytes are not included in outgoing packets. Sending the HDLC framing can be controlled in the config and on a per session basis using a Filter-Id in RADIUS authentication response.

BT sometimes negotiate incorrect MRUs on behalf of the LNS. Where the L2TP proxy details indicate and incorrect MRU has been negotiated then LCP negotiation is restarted and the correct MRU negotiates. This helps avoid various issues with fragmentation on some services on the internet when the broadband fully supports 1500 byte MTU. This is also relevant where the FB6000 is deliberately configured to use a smaller MRU for example when the L2TP connection is remote via a 1500 MTU link.

There are options using Filter-Id from RADIUS to force LCP restart. However this does confuse some ppp implementations as it is after authentication is complete. This can be useful where BT have provided an incorrect MRU for the end user (another bug). There is also an option to forward 1500 byte packets rather than fragmenting them. When enabled ICMP is still generated for DF and IPv6.

E.15.2. IP over LCP

IP over LCP is a non standard coding of PPP packets for IPv4 and IPv6. The coding uses the LCP code (C021) instead of the IPv4 (0021) or IPv6 (0057) code. The first byte which would normally be the LCP type is 0x4X (IPv4) or 0x6X (IPv6). The FireBrick assumes any such LCP codes are IPv4/IPv6 when received, and using a RADIUS response can send IP packets using LCP. This is specifically to bypass any carrier IP specific shaping or DPI.

Appendix F. Supported RADIUS Attribute/Value Pairs for L2TP operation

RADIUS is used for authentication and accounting of L2TP connections. If no authentication servers are configured then authentication is not performed. If no accounting servers are configured then no accounting is generated. Multiple servers can be configured and they are processed in order. Each can have multiple IP addresses. The IP addresses are tried based on the previous performance (response time, etc). If a server does not respond a number of times as configured then it is blacklisted (see Section 12.8.2.2 for more detail).

It is possible to configure local configurations which are checked before any RADIUS authentication.

It is possible to configure L2TP so that RADIUS accounting must respond, and if not then the sessions are disconnected.

F.1. Authentication request

Table F.1. Access-request

| AVP | No. | Usage |
|------------------------|-----|---|
| Message-Authenticator | 80 | Message signature as per RFC2869 |
| User-Name | 1 | Username from authentication (PAP/CHAP) or proxy authentication received on L2TP |
| Called-Station-Id | 30 | Called number as received on L2TP |
| Calling-Station-Id | 31 | Calling number as received on L2TP |
| Acct-Session-Id | 44 | Unique ID for session as used on all following accounting records |
| NAS-Identifier | 32 | Configured hostname of FireBrick |
| NAS-IP-Address | 4 | NAS IPv4 address if using IPv4 |
| NAS-IPv6-Address | 95 | NAS IPv6 address if using IPv6 |
| NAS-Port | 5 | L2TP session ID |
| NAS-Port-Id | 87 | For PPPoE "port{:vlan}/MAC" |
| Service-Type | 6 | Framed |
| Framed-Protocol | 7 | PPP |
| CHAP-Password | 3 | CHAP ID and response |
| CHAP-Challenge | 60 | CHAP challenge (only present if not the same as RADIUS authenticator) |
| Framed-MTU | 12 | MTU requested by PPP, if one was requested (even if 1500) |
| Connect-Info | 77 | Text Tx speed/Rx speed from L2TP connection if known |
| Tunnel-Client-Endpoint | 66 | Indicates the L2TP tunnel configured name attribute, allowing connections via different L2TP incoming configurations to be identified |
| Proxy-State | 33 | Added to session steering RADIUS requests (i.e. previous RADIUS returned type S tunnel) |

Note that the NAS-IP-Address is normally the local end of the L2TP connection for the incoming connection. However, there is a configuration option to pass the remote end of the L2TP as the NAS-IP-Address as this is

often more useful. If the remote Ip is used the NAS-Port is set to the far end L2TP session ID rather than the local end session ID. The NAS-Identified remains the name of the FB6000. This option is separately available for accounting messages.

Note that the Calling-Station-Id is included even if not present in L2TP connection if a cache platform RADIUS request matched the L2TP connection and had a Calling-Station-Id.

F.2. Authentication response

F.2.1. Accepted authentication

Table F.2. Access-Accept

| AVP | No. | Usage |
|-------------------------|--------|--|
| Reply-Message | 18 | Reply message sent on PPP authentication response |
| MS-Primary-DNS-Server | 311/28 | Primary DNS address used in PPP IPCP (Vendor 311 specific) |
| MS-Secondary-DNS-Server | 311/29 | Secondary DNS address used in PPP IPCP (Vendor 311 specific) |
| DNS-Server-IPv6-Address | 169 | DNS address used in DHCPv6 (may be specified multiple times) |
| Framed-Interface-ID | 96 | Peer IPv6 Interface ID expected in PPP IPV6CP |
| Framed-IP-Address | 8 | Peer IPv4 address expected in PPP IPCP (does not support 255.255.255.255 or 255.255.255.254 yet). Maximum localpref used. |
| NAS-IP-Address | 4 | Our end IPv4 address to use in IPCP negotiation. Does not add loopback route. This is non standard. |
| Framed-Route | 22 | May appear more than once. Text format is <i>IPv4-Address/Bits 0.0.0.0 metric</i> . The target IP is ignored but must be valid IPv4 syntax. The metric is used as localpref in routing. |
| Delegated-IPv6-Prefix | 123 | IPv6 prefix to be routed to line. Maximum localpref used. |
| Framed-IPv6-Prefix | 97 | IPv6 prefix to be routed to line. Maximum localpref used. |
| Framed-IPv6-Route | 99 | May appear more than once. Text format is <i>IPv6-Address/Bits :: metric</i> . The target IP is ignored but must be valid IPv6 syntax. The metric is used as localpref in routing. Alternative format <i>IPv6/Bits IPv4-Address metric</i> defines that prefix is to be protocol 41 IPv4 tunneled to specified target via this link. |
| Framed-IPv6-Address | 168 | Adds a /128 IPv6 route. |
| User-Name | 1 | Username may be specified - this replaces the username already present and is then used on accounting start and relay L2TP. |
| CHAP-Challenge | 60 | Overwrite relay details: If no length then forces PAP, else forces CHAP and sets challenge. Send before CHAP-Password or User-Password in packet. |
| CHAP-Password | 3 | Overwrite relay details: This replaces the CHAP ID and CHAP response |
| User-Password | 2 | Overwrite relay details: (not, plain text, not encoded). If CHAP, sets a response based on this, else forces PAP using this password. |
| Called-Station-Id | 30 | Called number may be specified - this replaces the number already present |
| Calling-Station-Id | 31 | Calling number may be specified - this replaces the number already present |

Supported RADIUS Attribute/
Value Pairs for L2TP operation

| | | |
|--------------------------|----|--|
| Chargeable-User-Identity | 89 | This is used as the preferred CQM graph name. |
| Class | 25 | Secondary CQM graph name to group sessions allowing group logging or shaping. |
| Session-Timeout | 27 | Absolute limit on session, in seconds |
| Filter-Id | 11 | See filter ID section |
| Framed-MTU | 12 | Set MTU for session |
| Connect-Info | 77 | Text tx/rx speed limit to apply to session (see below) |
| Tunnel-Type | 64 | If specified must be 3 (L2TP), L2TP is assumed. Also allows special 'R' and 'S' types, see below. |
| Tunnel-Medium-Type | 65 | If specified must be 1 (IPv4) or 2 (IPv6), syntax of endpoint is used if this is not specified |
| Tunnel-Server-Endpoint | 67 | Text IPv4 or IPv6 address of endpoint (FQDN is not accepted) |
| Tunnel-Client-Auth-ID | 90 | Hostname to quote on outgoing tunnel, if omitted then configured FireBrick hostname is used |
| Tunnel-Password | 69 | Shared secret to use on outgoing tunnel (encrypted), if omitted then assumed no secret |
| Tunnel-Assignment-ID | 81 | Name of outgoing tunnel shaper/graph. Also groups sessions together in a tunnel as per RFC. Only use valid text graph names. |
| Tunnel-Preference | 83 | Specifies preference order when multiple tagged endpoints sent |

Note that whilst a RADIUS response is normally relatively small it can get larger when multiple tunnel endpoints are included. Fragmented responses are handled but there is an internal limit to the size of response that can be processed - as such we recommend keeping the response to a single un-fragmented packet of up to 1500 bytes. You can use tag 0 for common settings such as Tunnel-Client-Auth-ID or Tunnel-Password when using multiple endpoints in order to reduce the size of the response.

The Tunnel-Type can be special, non-RFC values: 'R' (0x52) or 'S' (0x53) to indicate that the Tunnel-Server-EndPoint is the name or IP of a further RADIUS server to be interrogated. For 'R' this may respond with any valid authentication response. For 'S' the response will have all IP assignments filtered, and also not allow a Tunnel-Type 'S' response. Only one, tag 0 (untagged), RADIUS referral response will be accepted. The endpoint specified is server name or IP, and can have a number of numeric values appended (#port, >mintimeout, <maxtimeout, *scale, ?maxqueue).

The *Connect-Info* response can be a simple number (bits/second) tx rate, or a number followed by a % where this sets a speed based on a percentage of connection line speed. This can be followed by a slash (/) and the same for rx rate if required (default is rx is not limited). It is also possible to set long term shapers, this involves a number of additional parameters in the string prefixed with characters: > min, < max, _ minburst, and \ step. You need to include the *Chargeable-User-Identity* in the request for speeds to be set.

Some of the response fields are somewhat unconventional, such as User-Name and User-Password, allowing existing entries to be overwritten. These impact any details passed in a RADIUS accounting start and also impact relayed L2TP connections. Note that these can also impact if a secondary RADIUS request is done (e.g. if far end sends a new CHAP response, for example).

F.2.1.1. Prefix Delegation

The RADIUS authentication response can include Delegated-IPv6-Prefix, Framed-IPv6-Prefix, and Framed-IPv6-Route in order to route native IPv6 prefixes to the line. If there are any native IPv6 routes, or the Framed-IPv6-Interface attribute was specified, then IPV6CP negotiation is started. Framed-IPv6-Route can also be used to added IPv4 tunneled routes to the line. The FE80::/10 link local address negotiated with IPV6CP is not added to the routing for the line.

The client can send a Router solicitation to which the FireBrick will reply advising to use DHCPv6 for addressing. Once a router solicitation is sent, periodic Router Advertisements will then be sent on the connection by the Firebrick.

The client can use DHCPv6 to request an IA_NA (/128 link address), IA_TA (/128 temp link address), IA_PD (Prefix Delegation) and DNS servers. Prefixes are delegated based on the order in the DHCPv6 request and the order of Delegated-IPv6-Prefix, Framed-IPv6-Prefix, and then Framed-IPv6-Route, with multiple such entries in the order that they appeared in the RADIUS response. Such prefixes are not split up if a smaller prefix is requested, but the first part of a prefix is delegated.

As you can see, it is possible to send back *CHAP-Challenge* and *CHAP-Password* or *User-Password* to override those received by proxy or negotiation. This is mainly used when relaying the L2TP onwards and changes the proxy authentication field sent. This also overrides the Last LCP Tx proxy setting to show PAP/CHAP or no authentication as having been negotiated.

F.2.2. Rejected authentication

Table F.3. Access-Reject

| AVP | No. | Usage |
|---------------|-----|---|
| Reply-Message | 18 | Reply message sent on PPP authentication response |

Note that an authentication reject will normally cause the reply message to be sent as an authentication reject message. The reply "Try another" causes the L2TP session to be closed with result/error 2/7 (Try another) without sending an authentication reply on PPP.

F.3. Accounting Start

Table F.4. Accounting-Start

| AVP | No. | Usage |
|--------------------------|-----|---|
| Acct-Status-Type | 40 | 1 Start |
| User-Name | 1 | Username from authentication (PAP/CHAP) or proxy authentication received on L2TP or received in authentication response |
| Class | 25 | From authentication response if present |
| Chargeable-User-Identity | 89 | Graph name that applies, sanitised to comply with CQM graph name rules.. |
| Called-Station-Id | 30 | Called number as received on L2TP |
| Calling-Station-Id | 31 | Calling number as received on L2TP |
| Service-Type | 6 | Framed |
| Framed-Protocol | 7 | PPP |
| Framed-MTU | 12 | Final MTU being used for session |
| Filter-Id | 11 | Filters in use |
| Session-Timeout | 27 | Absolute limit on session, in seconds, if specified in authentication reply |
| Framed-Interface-ID | 96 | Peer IPv6 Interface ID from PPP IPV6CP |
| Framed-IP-Address | 8 | Peer IPv4 address negotiated in PPP (normally from authentication response) |
| Connect-Info | 77 | Text Tx speed/Rx speed in use |
| Acct-Delay-Time | 41 | Seconds since session started |

Supported RADIUS Attribute/
Value Pairs for L2TP operation

| | | |
|----------------------------|----|---|
| Acct-Event- Timestamp | 55 | Session start time (unix timestamp) |
| Acct-Session-Id | 44 | Unique ID for session |
| NAS-Identifier | 32 | Configured hostname of FireBrick |
| NAS-IP-Address | 4 | NAS IPv4 address if using IPv4 |
| NAS-IPv6-Address | 95 | NAS IPv6 address if using IPv6 |
| NAS-Port | 5 | L2TP session ID |
| Tunnel-Type | 64 | Present for relayed L2TP sessions, L2TP |
| Tunnel-Medium-Type | 65 | Present for relayed L2TP, 1 (IPv4) or 2 (IPv6) |
| Tunnel-Client- Endpoint | 66 | Present for relayed L2TP, text IPv4 or IPv6 address of our address on the outbound tunnel |
| Tunnel-Server- Endpoint | 67 | Present for relayed L2TP, text IPv4 or IPv6 address of the far end address of the outbound tunnel |
| Tunnel-Assignment- ID | 82 | Present for relayed L2TP, text local L2TP tunnel ID |
| Tunnel-Client-Auth- ID | 90 | Present for relayed L2TP, local end hostname quoted by outgoing tunnel |
| Tunnel-Server-Auth- ID | 91 | Present for relayed L2TP, far end hostname quoted by outgoing tunnel |

Note that most parameters are not included in interim and stop accounting records. The acct-session-id should be used by accounting servers to correlate interim and stop records with the start record. The graph name could be used, but is only available where there is a graph. If too many different graphs then that is not present. Some exceptions apply as they can be changed by a change of authorisation RADIUS request, such as Connect-Info.

F.4. Accounting Interim

Table F.5. Accounting-Interim

| AVP | No. | Usage |
|------------------------------|-----|---|
| Acct-Status-Type | 40 | 3 Interim-Update |
| Acct-Delay-Time | 41 | Seconds since accounting data collected |
| Acct-Event- Timestamp | 55 | Data collected time (unix timestamp) |
| Acct-Session-Id | 44 | Unique ID for session |
| Chargeable-User- Identity | 89 | Graph name that applies, sanitised to comply with CQM graph name rules.. |
| Connect-Info | 77 | Text Tx speed/Rx speed in use |
| Framed-IP-Address | 8 | Peer IPv4 address negotiated in PPP (normally from authentication response) |
| NAS-Identifier | 32 | Configured hostname of FireBrick |
| NAS-IP-Address | 4 | NAS IPv4 address if using IPv4 |
| NAS-IPv6-Address | 95 | NAS IPv6 address if using IPv6 |
| NAS-Port | 5 | L2TP session ID |
| Acct-Input-Octets | 42 | Rx byte count |
| Acct-Input- Gigawords | 52 | Rx byte count (high 4 bytes) |

Supported RADIUS Attribute/
Value Pairs for L2TP operation

| | | |
|------------------------|----|---|
| Acct-Output-Octets | 43 | Tx byte count |
| Acct-Output-Gigawords | 53 | Tx byte count (high 4 bytes) |
| Acct-Input-Packets | 47 | Rx packet count |
| Acct-Output-Packets | 48 | Tx packet count |
| Tunnel-Type | 64 | Present for relayed L2TP sessions, L2TP |
| Tunnel-Medium-Type | 65 | Present for relayed L2TP, 1 (IPv4) or 2 (IPv6) |
| Tunnel-Client-Endpoint | 66 | Present for relayed L2TP, text IPv4 or IPv6 address of our address on the outbound tunnel |
| Tunnel-Server-Endpoint | 67 | Present for relayed L2TP, text IPv4 or IPv6 address of the far end address of the outbound tunnel |
| Tunnel-Assignment-ID | 82 | Present for relayed L2TP, text local L2TP tunnel ID |
| Tunnel-Client-Auth-ID | 90 | Present for relayed L2TP, local end hostname quoted by outgoing tunnel |
| Tunnel-Server-Auth-ID | 91 | Present for relayed L2TP, far end hostname quoted by outgoing tunnel |

F.5. Accounting Stop

As accounting interim update plus

Table F.6. Accounting-Stop

| AVP | No. | Usage |
|----------------------|-----|---------------------------|
| Acct-Terminate-Cause | 49 | Cause code as appropriate |

Cause codes of note are 2(Lost-Carrier) which is sent if LCP echos do not reply for several seconds, and 14(Port-Suspended) which is sent if the dos-limit is exceeded on a session. For DOS handling it is recommended that subsequent authentication requests are rejected for several minutes or a fake accept is and session-timeout is used as DOS attacks usually continue until the customer is off-line.

F.6. Disconnect

A disconnect message is accepted as per RFC5176, if the session can be disconnected, and ACK is sent, else a NAK

Table F.7. Disconnect

| AVP | No. | Usage |
|--------------------------|-----|---|
| Acct-Session-Id | 44 | Unique ID for session |
| Chargeable-User-Identity | 89 | This is used as CQM graph name. |
| Acct-Terminate-Cause | 49 | Cause code as appropriate to be used in accounting stop message |

The session is identified by Acct-Session-Id if present, else by Chargeable-User-Identity. No other identification parameters are supported. If sent then they are ignored.

F.7. Change of Authorisation

A change of authorisation message is accepted as per RFC5176

Table F.8. Change-of-Authorisation

| AVP | No. | Usage |
|--------------------------|-----|--|
| Acct-Session-Id | 44 | Unique ID for session |
| Chargeable-User-Identity | 89 | This is used as CQM graph name. |
| Framed-Route | 22 | May appear more than once. Text format is <i>IPv4-Address/Bits 0.0.0.0 metric</i> . The target IP is ignored but must be valid IPv4 syntax. The metric is used as localpref in routing. |
| Delegated-IPv6-Prefix | 123 | IPv6 prefix to be routed to line. Maximum localpref used. |
| Framed-IPv6-Prefix | 97 | IPv6 prefix to be routed to line. Maximum localpref used. |
| Framed-IPv6-Route | 99 | May appear more than once. Text format is <i>IPv6-Address/Bits :: metric</i> . The target IP is ignored but must be valid IPv6 syntax. The metric is used as localpref in routing. Alternative format <i>IPv6/Bits IPv4-Address metric</i> defines that prefix is to be protocol 41 IPv4 tunneled to specified target via this link. |
| Session-Timeout | 27 | Absolute limit on session, in seconds |
| Terminate-Action | 29 | If not specified, or 0, then terminate on Session-Timeout or Quota reached, else send RADIUS Interim accounting update (not an Access Request) |
| Connect-Info | 77 | Text tx speed limit to apply to session (see below) |
| Filter-Id | 11 | See filter ID section |

The session is identified by Acct-Session-Id if present, else by Chargeable-User-Identity. No other identification parameters are supported. If sent then they are ignored.

Parameters are left unchanged if not specified.

- If any route entries (IPv4 or IPv6) are present then the existing routing (apart from the PPP endpoint address) are all removed and replaced with what has been sent. To clear all routes send a framed route with a blank string. The Framed-IP-Address cannot be changed.
- To clear the session timeout send a value of 0.
- To clear remove shaping send a *Connect-Info* of "0".

No other parameters are supported, and if sent then they are ignored

The *Connect-Info* response can be a simple number (bits/second) tx rate, or a number followed by a % where this sets a speed based on a percentage of current line speed. It is also possible to set long term shapers where a *Chargeable-User-Identity* is also included, this involves a number of additional parameters in the string prefixed with characters: > min, < max, _ minburst, and \ step.

F.8. Filter ID

The Filter-ID can be set in authentication response and change of authorisation. There can be many records. Each can have many filters. Each filter is of the form of a letter possibly followed by number digits. The accounting start lists relevant filters that have been set, each in a separate filter-id AVP. Unknown filters are ignored.

Table F.9. Filter-ID

| Filter | meaning |
|---------------|--|
| Tn | Set routing table for payload traffic. This can be used for private routing, and for walled garden / credit control |
| An | Specify this connection is a member of a closed user group <i>n</i> (1-32767) but has normal IP access as well. This connection is not filtered by traffic can go to/from connections that are filtered in the same CUG |
| Rn | Specify this connection is a member of a closed user group <i>n</i> (1-32767) and is restricted to sending traffic to/from connections in the same CUG. |
| H | Sets the connection to send HDLC framing headers on all PPP packets. This adds 2 extra byte to the packet. This is the default setting. |
| h | Sets the connection not to send HDLC framing headers on all PPP packets. This is in accordance with the L2TP/PPP RFCs. This does not work on BT 21CN BRASs. |
| F | Sets TCP MTU fix flag which causes the MTU option in TCP SYN to be adjusted if necessary to fit MTU. |
| f | Sets no TCP MTU fix |
| M | Sets the connection to ignore the MRU. Actually, the MRU is used to generate ICMP errors for IPv6 and IPv4 with DF set, but otherwise full size packets are sent on the connection even if a lower MRU was advised. This is in accordance with the PPP RFC but breaks some routers that do not accept 1500 byte packets (e.g. PPPoE) |
| m | Sets the connection to fragment IPv4 packets with DF not set that are too big for the advised MRU. This is the default |
| L | This is not a filter and not confirmed back on accounting start and not valid on Change of Authorisation. It forces a restart of LCP negotiation. This is useful when BRASs lie about negotiated LCP (such as BTs 21CN BRASs) |
| l | This is not a filter and not confirmed back on accounting start and not valid on Change of Authorisation. It stops an LCP negotiation restart that may be planned, e.g. due to an MRU mismatch. |
| X | Pad packets to 74 bytes if length fields appears to be less - needed to work around bug in BT 20CN BRAS for IPv6 in IP over LCP mode |
| C | Send all IPv4 and IPv6 using the LCP type code (only works if FireBrick doing PPP at far end) |
| I | Mark session isolated from other L2TP sessions (no direct packets from other L2TP sessions allowed) |
| J | Mark session for latency adjust on bonding (adds half round trip latency to bonding calculations) |
| O | Mark session as <i>low-priority</i> (see shaper and damping) |
| P | Mark session as <i>premium</i> (see shaper and damping) |
| D | Mark session as <i>blackhole</i> (Normal IPv4/IPv6 routes are announced as black hole routes, and any BGP is not restricted to local-as, etc. Does not apply to 6over4 routes) |
| d | Mark session as not <i>blackhole</i> |
| b | Disable anti-spoofing source filtering |
| Sn | Set LCP echo rate to <i>n</i> seconds (default 1) |
| sn | Set LCP timeout rate to <i>n</i> seconds (default 10) |
| q[+] <i>n</i> | Specify [or add to] quota for tx bytes. Use either q or Q. Action depends on Terminate-Action. |
| Q[+] <i>n</i> | Specify [or add to] quota for total (tx+rx) bytes. |

For change of authorisation the absence of a filter has no effect. To set normal routing table 0 zero, send T0. To set not a member of a CUG send A0.

F.9. Notes

F.9.1. L2TP relay

L2TP relay means that an incoming call (ICRQ) is relayed to another L2TP endpoint. The decision of which calls to relay to what endpoint can be made in one of two ways:-

- Configured pattern match based on calling number, called number, or login.
- RADIUS response to initial authentication request advising new endpoint for connection.

A test is made against the config on the initial connection based on known data. This is calling number (if present), called number (if present) and login (proxy_auth_name if present). If a match is found the call is relayed with no additional PPP packets exchanged.

If there is no proxy LCP provided, or the provided negotiation conflicts with the configuration, then LCP negotiation is completed.

If there is no proxy authentication, PPP authentication is start until a response/login is received from the peer (assuming authentication is required in the config).

At this point a further check is made for a configured relay which can now be based on a login if one was not present before.

RADIUS authentication is completed, and if the response indicates a relay then the call is relayed.

The relayed call includes the incoming call parameters, and any LCP and authentication parameters that may have been negotiated at that point.

The Filter ID *T* which would normally set the payload table when terminating traffic is instead used to set the routing table for the outgoing relay tunnel.

F.9.2. LCP echo and CQM graphs

Each session gets a CQM graph which uses one second LCP requests and produces detailed loss/latency graphs for the session. The graph name is picked based on the first available of :-

- Chargeable-User-Identity sent in the RADIUS authentication response.
- Calling-Station-Id from L2TP.
- User-name in RADIUS authentication response.
- Proxy-Auth-Name from L2TP.
- Negotiated user name from PAP/CHAP.

If a second session starts with the same graph name as an existing session then the existing session is cleared with cause 13(Preempted). It is recommended that a unique circuit ID is passed as the Chargeable-User-Identity in the authentication response to allow simple location of graphs.

Normally the FireBrick sends regular LCP echos and generates a loss/latency graph. Where this is not enabled (LCP Rate 0 or no graph set at all) then graphs are created but with no loss/latency. On a relayed L2TP connection the LCP echos are normally generated by the far ends and passed through by the FireBrick. However, if a RADIUS reply sets the LCP rate/timeout and provides tunnel relay, then the incoming side of the relayed connection will use LCP echos from the FireBrick in the middle of the connection and not pass these through - this means on the o/g connection the FireBrick answers LCP echos from the relayed LNS.

F.9.3. IP over LCP

IP over LCP is a non standard coding of PPP packets for IPv4 and IPv6. The coding uses the LCP code (C021) instead of the IPv4 (0021) or IPv6 (0057) code. The first byte which would normally be the LCP type is 0x4X (IPv4) or 0x6X (IPv6). The FireBrick assumes any such LCP codes are IPv4/IPv6 when received, and using a RADIUS response can send IP packets using LCP. This is specifically to bypass any carrier IP specific shaping or DPI.

F.9.4. Closed User Group

Each session can have a CUG defined (1-32768) which may be *allow* or *restrict*. Interfaces (port/VLAN) may also be defined in the same way. A packet from an interface/session with a CUG is tagged with that packet. If the source is restricted that packet can only leave via an interface/session with the same CUG. Similarly if the target interface/session is restricted than only a packet tagged with the same CUG can be sent to it.

F.9.5. Routing table

The FireBrick operates independent routing cores allowing a totally independent routing table to be used for L2TP wrapper traffic and payload traffic. It is also possible to set the payload table in use on a per session basis from RADIUS thus allowing a *walled garden* to be set up, or a private network, or simple an unusable session.

Appendix G. FireBrick specific SNMP objects

This appendix details the SNMP objects that are specific to the FireBrick.

G.1. Conventions

To avoid writing out very repetitive (and long) OIDs this appendix uses a substitution notation when describing structures. The OIDs contain the names of indices and an X representing the structure field number given by the matching column in the following table. So if the OID given in the manual is `iso.1.2.3.4.X.anIndex` then the OID for the first item described in the table row with column X = 3 would be: `iso.1.2.3.4.3.1`

G.1.1. IP addresses as indices

In some structures IP addresses may be used as indices into SNMP tables. When this is the case the IP must be encoded into the OID for the resource. This encoding consists of the following separated by dots:

1. The IP type: 1 for IPv4, 2 for IPv6.
2. A length prefixed array of the bytes of the address in network order.

Example G.1.

For the IPv4 address 10.0.1.12 the IP type is 1 and the length is 4. Therefore the OID fragment `1.4.10.0.1.12` could be used as an index.

Example G.2.

For the IPv6 address 2001::32 the IP type is 2 and the length is 16. Therefore the OID fragment `2.16.32.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.50` could be used as an index.

G.2. Firebrick-specific structures for BGP

G.2.1. Structure definitions

G.2.1.1. The list of BGP peers for this Firebrick

OID: `iso.3.6.1.4.1.24693.100.179.1.1.X.fbBgpPeerAddressType.fbBgpPeerAddress`

Table G.1. Indices

| Name | Type | Meaning |
|----------------------|---|---|
| fbBgpPeerAddressType | InetAddressType | The address type of fbBgpPeerAddressAddr. |
| fbBgpPeerAddress | InetAddress (with length prefix, see Section G.1.1) | The internet address for the peer. The type of the address is determined by the value of the fbBgpPeerAddressType object. |

Table G.2. Fields

| X | MIB name | Type | Meaning |
|---|----------|------|---------|
|---|----------|------|---------|

| | | | |
|----|---------------------------------|----------------------------|---|
| 1 | fbBgpPeerAddressType | InetAddressType | The address type of fbBgpPeerAddressAddr. |
| 2 | fbBgpPeerAddress | InetAddress | The internet address for the peer. The type of the address is determined by the value of the fbBgpPeerAddressType object. |
| 3 | fbBgpPeerName | DisplayString | The name of the BGP Peer |
| 4 | fbBgpPeerState | FbBgpPeerState (Table G.3) | The current state of the BGP Peer |
| 5 | fbBgpPeerRemoteAS | Integer32 | The remote AS of the BGP Peer |
| 6 | fbBgpPeerReceivedIpv4Prefixes | Integer32 | The number of IPv4 prefixes received from the BGP Peer |
| 7 | fbBgpPeerSecondsSinceLastChange | Integer32 | The number of seconds since the last state change for the BGP Peer |
| 8 | fbBgpPeerReceivedIpv6Prefixes | Integer32 | The number of IPv6 prefixes received from the BGP Peer |
| 9 | fbBgpPeerExported | Integer32 | The number of prefixes exported to the BGP Peer |
| 10 | fbBgpPeerLocalAddressType | InetAddressType | The address type of fbBgpPeerLocalAddress. |
| 11 | fbBgpPeerLocalAddress | InetAddress | The local internet address used for this peer. The type of the address is determined by the value of the fbBgpPeerAddressType object. |
| 12 | fbBgpPeerLocalAS | Integer32 | The local AS number for the BGP Peer |
| 13 | fbBgpPeerTableId | Integer32 | The routing table number for this BGP Peer. |
| 14 | fbBgpPeerMaxPrefixes | Integer32 | Max prefixes accepted from this peer. |
| 15 | fbBgpPeerMaxPrefixesHit | TruthValue | Prefix limit was hit, so prefixes from this peer have been dropped. |

G.2.2. Enum Definitions

Table G.3. FbBgpPeerState - The state of a BGP peer

| Value | Meaning |
|-------|---|
| 0 | Disabled or in error backoff state or inbound only peer |
| 1 | Attempting to connect |
| 2 | Awaiting open message (incoming) |
| 3 | Awaiting response to open, or for TCP to establish (outgoing) |
| 4 | Awaiting confirmation of establishment |
| 5 | Connection established |
| 6 | Closing or awaiting cleanup |
| 7 | Withdrawing everything and waiting for propagation |
| 8 | Shutdown |

G.3. Firebrick-specific structures for L2TP

G.3.1. Structure definitions

G.3.1.1. fbL2tpGeneralTunnels

OID: iso.3.6.1.4.1.24693.100.1701.0.1.X

Table G.4. Fields

| X | MIB name | Type | Meaning |
|---|---------------------------|-----------|--|
| 0 | fbL2tpFreeTunnels | Integer32 | Number of tunnels in FREE state |
| 1 | fbL2tpOpeningTunnels | Integer32 | Number of tunnels in OPENING state |
| 2 | fbL2tpLiveIncomingTunnels | Integer32 | Number of tunnels in LIVE (Incoming) state |
| 3 | fbL2tpLiveOutgoingTunnels | Integer32 | Number of tunnels in LIVE (Outgoing) state |
| 4 | fbL2tpClosingTunnels | Integer32 | Number of tunnels in CLOSING state |
| 5 | fbL2tpFailedTunnels | Integer32 | Number of tunnels in FAILED state |
| 6 | fbL2tpClosedTunnels | Integer32 | Number of tunnels in CLOSED state |

G.3.1.2. fbL2tpGeneralSessions

OID: iso.3.6.1.4.1.24693.100.1701.0.2.X

Table G.5. Fields

| X | MIB name | Type | Meaning |
|----|-----------------------------------|-----------|--|
| 0 | fbL2tpFreeSessions | Integer32 | Number of sessions in FREE state |
| 1 | fbL2tpWaitingSessions | Integer32 | Number of sessions in WAITING state |
| 2 | fbL2tpOpeningSessions | Integer32 | Number of sessions in OPENING state |
| 3 | fbL2tpNegotiatingSessions | Integer32 | Number of sessions in NEGOTIATING state |
| 4 | fbL2tpAuthPendingSessions | Integer32 | Number of sessions in AUTH-PENDING state |
| 5 | fbL2tpStartedSessions | Integer32 | Number of sessions in STARTED state |
| 6 | fbL2tpLiveSessions | Integer32 | Number of sessions in LIVE state |
| 7 | fbL2tpAcctPendingSessions | Integer32 | Number of sessions in ACCT-PENDING state |
| 8 | fbL2tpClosingSessions | Integer32 | Number of sessions in CLOSING state |
| 9 | fbL2tpClosedSessions | Integer32 | Number of sessions in CLOSED state |
| 10 | fbL2tpSessionNegotiationSlotsFree | Integer32 | Number of session negotiation slots free |

G.3.1.3. The list of L2TP peers for this Firebrick

OID: iso.3.6.1.4.1.24693.100.1701.1.1.X.fbL2tpPeerAddressType.fbL2tpPeerAddress

Table G.6. Indices

| Name | Type | Meaning |
|------|------|---------|
|------|------|---------|

| | | |
|-----------------------|---|--|
| fbL2tpPeerAddressType | InetAddressType | The internet address type for the peer. |
| fbL2tpPeerAddress | InetAddress (with length prefix, see Section G.1.1) | The internet address for the peer. The type of the address is determined by the value of the fbL2tpAddressType object. |

Table G.7. Fields

| X | MIB name | Type | Meaning |
|----|------------------------|-----------------|--|
| 1 | fbL2tpPeerAddressType | InetAddressType | The internet address type for the peer. |
| 2 | fbL2tpPeerAddress | InetAddress | The internet address for the peer. The type of the address is determined by the value of the fbL2tpAddressType object. |
| 3 | fbL2tpPeerLoginName | DisplayString | The login name of the L2TP Peer |
| 4 | fbL2tpPeerHostName | DisplayString | The host name of the L2TP Peer |
| 5 | fbL2tpPeerInTunnels | Integer32 | The number of inbound tunnels from the L2TP peer |
| 6 | fbL2tpPeerOutTunnels | Integer32 | The number of outbound tunnels from the L2TP peer |
| 7 | fbL2tpPeerOldestUptime | Integer32 | The uptime of the oldest tunnel from the L2TP peer |
| 8 | fbL2tpPeerLiveTunnels | Integer32 | The number of live tunnels from the L2TP peer |
| 9 | fbL2tpPeerSessions | Integer32 | The number of sessions from the L2TP peer |
| 10 | fbL2tpPeerGraphName | DisplayString | The graph name of the L2TP peer |

G.4. Firebrick CPU usage

G.4.1. Structure definitions

G.4.1.1. CPU usage for this Firebrick

OID: iso.3.6.1.4.1.24693.100.2.1.1.X.fbCpuPeriod.fbCpuCore

Table G.8. Indices

| Name | Type | Meaning |
|-------------|-----------|---|
| fbCpuPeriod | Integer32 | The period in minutes covered by this table entry. Zero indicates that an instantaneous value is required. Max is 30 minutes. |
| fbCpuCore | Integer32 | The CPU core number covered by this table entry. The numbering starts at 1, so CPU0 1, CPU1 is 2 etc. |

Table G.9. Fields

| X | MIB name | Type | Meaning |
|---|----------|---------|---|
| 1 | fbCpuIRQ | Gauge32 | The percentage of CPU time spent in interrupt processing for this period. If period is 0 the instantaneous usage in the |

| | | | |
|---|--------------|---------|--|
| | | | last second is used. Units are 100ths of a percent, so 10000 indicates 100%. |
| 2 | fbCpuAll | Gauge32 | The total percentage of CPU time spent non-idle for this period. If period is 0 the instantaneous usage in the last second is used. Units are 100ths of a percent, so 10000 indicates 100%. |
| 3 | fbCpuIRQPeak | Gauge32 | The peak percentage of CPU time in interrupt processing during this period. If period is 0 the peak usage in the current minute is used. Units are 100ths of a percent, so 10000 indicates 100%. |
| 4 | fbCpuAllPeak | Gauge32 | The peak percentage of CPU time non-idle during this period. If period is 0 the peak usage in the current minute is used. Units are 100ths of a percent, so 10000 indicates 100%. |

G.5. Firebrick system stats

G.5.1. Structure definitions

G.5.1.1. The table of runtime stats for this Firebrick

OID: iso.3.6.1.4.1.24693.100.3.1.1.X.fbRunCore

Table G.10. Indices

| Name | Type | Meaning |
|-----------|-----------|---|
| fbRunCore | Integer32 | The CPU core number covered by this table entry. The numbering starts at 1, so CPU0 is 1 and CPU1 is 2 etc. |

Table G.11. Fields

| X | MIB name | Type | Meaning |
|---|---------------------------|---------|---|
| 1 | fbRunBuffers [deprecated] | Gauge32 | The count of buffers which are currently globally free in the system. |

G.6. Monitoring for general system features

G.6.1. Structure definitions

G.6.1.1. The list of readings for this Firebrick

OID: iso.3.6.1.4.1.24693.100.1.1.1.X.fbMonReadingIndex

Table G.12. Indices

| Name | Type | Meaning |
|-------------------|-----------|----------------------------------|
| fbMonReadingIndex | Integer32 | The index for the readings table |

Table G.13. Fields

| X | MIB name | Type | Meaning |
|---|-------------------|---------------|---------------------------|
| 2 | fbMonReadingType | DisplayString | The type of this reading |
| 3 | fbMonReadingName | DisplayString | The name of this reading |
| 4 | fbMonReadingValue | Integer32 | The value of this reading |

G.7. System wide status

G.7.1. Structure definitions

G.7.1.1. fbGlobalMemory

OID: iso.3.6.1.4.1.24693.100.4.1.X

Table G.14. Fields

| X | MIB name | Type | Meaning |
|---|------------|---------|-----------------|
| 1 | fbTotalMem | Gauge32 | Total RAM (KiB) |
| 2 | fbFreeMem | Gauge32 | Free RAM (KiB) |

G.7.1.2. fbGlobalBuffers

OID: iso.3.6.1.4.1.24693.100.4.2.X

Table G.15. Fields

| X | MIB name | Type | Meaning |
|---|--------------------|---------|---------------------------------------|
| 1 | fbFreeBuffers | Gauge32 | Small buffers in the global free pool |
| 2 | fbFreeLargeBuffers | Gauge32 | Large buffers in the global free pool |

G.8. Monitoring information (deprecated)

General monitoring information. These OIDs are deprecated, please migrate to the new ones in Section G.6.

Table G.16. iso.3.6.1.4.1.24693.1.X.Y

| X.Y | Type | Meaning |
|-----|--------------|--|
| 1.1 | Integer (mV) | Voltage: "A" power supply. Should be around 12V. May show a few volts when no power connected |
| 1.2 | Integer (mV) | Voltage: "B" power supply. Should be around 12V. May show a few volts when no power connected |
| 1.3 | Integer (mV) | Voltage: Combined 12V supply. Will be slightly lower than highest of "A" and "B" due to diode drop |
| 1.4 | Integer (mV) | Voltage: 3.3V reference |
| 1.5 | Integer (mV) | Voltage: 1.8V reference |
| 1.6 | Integer (mV) | Voltage: 1.2V reference |
| 1.7 | Integer (mV) | Voltage: 1.1V reference |

FireBrick specific SNMP objects

| | | |
|-----|---------------|-------------------------------------|
| 1.8 | Integer (mV) | Voltage: 3.3V fan power, if present |
| 1.9 | Integer (mV) | Voltage: 1.2V fan power, if present |
| 2.1 | Integer (mC) | Temperature: Fan controller |
| 2.2 | Integer (mC) | Temperature: CPU |
| 2.3 | Integer (mC) | Temperature: RTC |
| 3.1 | Integer (rpm) | Speed: Fan 1 |
| 3.2 | Integer (rpm) | Speed: Fan 2 |

Appendix H. Command line reference

H.1. General commands

H.1.1. Trace off

```
troff
```

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

H.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.

H.1.3. Uptime

```
uptime  
show uptime
```

Shows how long since the FB6000 restarted.

H.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.

H.1.5. Memory usage

```
show memory
```

Shows memory usage summary.

H.1.6. Process/task usage

```
show tasks
```

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

H.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.

H.1.8. Logout

```
logout
quit
exit
```

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

H.1.9. See XML configuration

```
show run
show configuration
```

Dumps the full XML configuration to the screen

H.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 configurations. This command is provided as a last resort for emergency use, so use with care.

H.1.11. Show profile status

```
show profiles
```

Shows profiles and current status.

H.1.12. Enable profile control switch

```
enable profile <string>
```

Turns a named profile control switch on.

H.1.13. Disable profile control switch

```
disable profile <string>
```

Turns a named profile control switch off.

H.1.14. Show RADIUS servers

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

Shows details of RADIUS servers currently in use

H.1.15. Show DNS resolvers

```
show dns
```

Shows current DNS resolver list and status.

H.2. Networking commands

H.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.

H.2.2. Renegotiate DHCP for a subnet

```
dhcpc renegotiate subnet=<integer>
```

When the Firebrick is getting its own IP address(es) from an external DHCP source, you may sometimes want it to renegotiate this address (e.g. if you have disconnected and connected to a different DHCP server). This command will force the DHCP client to renegotiate for a particular subnet ID. You can find the ID number of your subnets with the **show subnets** command.

NB This is separate from the Firebrick's own DHCP server, which can allocate addresses to machines on a LAN (see the **dhcpc** commands below).

This command is only available to ADMIN (and DEBUG) user levels. It should be used with caution as it will interrupt your connection to the external network, and it is very possible that you may be allocated a different IP address.

H.2.3. 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.

H.2.4. 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 supplied, the route that would be used is shown. But if a prefix is supplied then the route for that prefix is shown, even though there may be more specific routes in use within it.

H.2.5. List routes

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

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

H.2.6. List routing next hops

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

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

H.2.7. See DHCP allocations

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

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

H.2.8. Clear DHCP allocations

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

Allows you to remove one or more DHCP allocations.

H.2.9. 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.

H.2.10. Unlock DHCP allocations

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

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

H.2.11. Name DHCP allocations

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

Allows you to set a name for a DHCP allocation, overriding the client name that was sent.

H.2.12. Show ARP/ND status

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

Shows details of ARP and Neighbour discovery cache.

H.2.13. Show VRRP status

```
show vrrp
```

Lists all VRRP in use and current status.

H.2.14. Send Wake-on-LAN packet

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

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

H.2.15. 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.

H.3. BGP commands

H.3.1. Show BGP

Required user level: user

```
show bgp
```

H.3.2. Show BGP Peer

Required user level: user

```
show bgp peer <IPNameAddr>
```

Show peer by name/ip.

H.3.3. Show BGP Summary

Required user level: user

```
show bgp summary
```

Show summary of BGP peers.

H.3.4. Show BGP Routes

Required user level: user

```
show bgp routes [route=<IPFilter>] [table=<routetable>] [imported=<IPNameAddr>]  
[exported=<IPNameAddr>] [limit=<integer>]
```

Show routes.

H.3.5. Clear BGP

Required user level: user

```
clear bgp peer=<IPNameAddr>
```

Clear BGP peer(s).

H.3.6. Refresh BGP

Required user level: user

```
refresh bgp peer=<IPNameAddr> in
```

Refresh BGP peer(s) incoming.

H.3.7. Refresh BGP

Required user level: user

```
refresh bgp peer=<IPNameAddr> out
```

Refresh BGP peer(s) outgoing.

H.4. L2TP commands

H.4.1. Show L2TP

Required user level: user

```
show l2tp
```

Show a summary of L2TP tunnels and sessions.

H.4.2. Show L2TP Tunnels

Required user level: user

```
show l2tp tunnels [detail]
```

Show information about L2TP tunnels. The optional detail specifier shows the counts of sessions in each state for each tunnel.

H.4.3. Clear L2TP All

Required user level: user

```
clear l2tp all
```

Sends a manual termination to all live sessions in a staggered manner to spread reconnection load.

H.4.4. Show L2TP Tunnel

Required user level: user

```
show l2tp tunnel <tun-id> [detail]
```

Show information about an L2TP tunnel. The optional detail specifier shows the counts of sessions in each state.

H.4.5. Show L2TP Tunnel

Required user level: user

```
show l2tp tunnel <tun-id> sessions
```

Show information about an L2TP tunnel. The optional detail specifier shows the counts of sessions in each state.

H.4.6. Show L2TP Sessions

Required user level: user

```
show l2tp sessions
```

Show details of all L2TP sessions.

H.4.7. Show L2TP Session

Required user level: user

```
show l2tp session <string>
```

Show L2TP session details by name or number.

H.4.8. Clear L2TP Tunnel

Required user level: user

```
clear l2tp tunnel <tun-id>
```

Clear L2TP tunnel by ID or IP/name.

H.4.9. Clear L2TP Tunnel

Required user level: user

```
clear l2tp tunnel <IPNameAddr>
```

Clear L2TP tunnel by ID or IP/name.

H.4.10. Clear L2TP Session

Required user level: user

```
clear l2tp session <string>
```

Clear L2TP session by graph name.

H.5. Advanced commands

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

H.5.1. 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**

H.5.2. 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.

H.5.3. Screen width

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

This allows you to set the screen width.

H.5.4. 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.

H.5.5. Show command sessions

```
show command sessions
```

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

H.5.6. 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.

H.5.7. 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.

H.5.8. 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.

H.5.9. Boot log

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

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

H.5.10. 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 I. 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 - typically used for broadband lines on L2TP .

- 100 second interval statistics available graphically as svg or png and in text as csv covering at least the last 25 hours (one day)
- Loss latency stats where available (e.g. LCP echos on L2TP broadband lines) including minimum, average, and maximum latency for the 100 second sample, and percentage packet loss.
- Throughput stats where available (e.g. interfaces, shapers , L2TP broadband lines) 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 L2TP broadband line has both. An interface or shaper normally has only throughput data.

I.1. Broadband back-haul providers

When using the FB6000 as an LNS the CQM graphs are invaluable for diagnosing line faults. They are useful to the ISP but also useful to the back-haul provider which is often a separate company (e.g. BT or Be). We recommend that you consider providing access to graphs for live circuits and archived data to your back-haul provider when discussing faults with them. FireBrick are working with several ISPs to ensure back-haul providers are aware of the CQM graphs and how to use them to assist in diagnosis.

I.2. Access to graphs and csvs

Graphs can be accessed via 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) defines the port, 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.

I.2.1. 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 I.1. File types

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

I.2.2. Dated information

Without any date the data returned is the latest. This includes 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`.

I.2.3. Authenticated access

Authenticated 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 hash (32 character hex) can also be used instead, but SHA1 is the preferred method.

I.3. 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.

Note

These attributes apply to both png and svg output, however it is also possible to override the svg style and use a CSS style sheet from a URL instead. In such cases none of the colour settings from the config or the graph URL will apply. See an actual svg output for classes that are used in the graphs.

I.3.1. Scaleable Vector Graphics

The graphs are available in PNG and SVG. We recommend using SVG graphs.

The SVG version is usually a lot quicker to load, and has a lot more detail. It works well on high resolution monitors and devices that allow scaling (such as mobile phones).

The SVG is designed to allow some post processing or manipulation with specific paths for each part of the graph, such as "tx", "rx", "min", "ave", "max", etc. These data paths are also presented using a fixed scale, and then scaled and cropped by a wrapping group. This allows tools to merge graphs from different sources and maintain scales, or allow scales to be changed later. If you actually want access to the raw data, then you should use the XML output.

There is an option for an external CSS to be referenced, rather than local style based on the config settings. This allows even more control and flexibility for display.

There is also an option to include an overlay providing tool tip (title) information for each 100 second sample. This makes for much larger files though.

I.3.2. 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 I.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 |

I.3.3. 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 I.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 of 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 |

Note that z and z do not normally have any arguments, but if they do, then it sets the SVG CSS to use, e.g. `z(mystyle.css)` but an empty string, e.g. `z()`, will use the standard config based style. Setting z forces no style and no CSS so parent document can set styles.

I.3.4. 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 I.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 are shown based on space at top and bottom. |
| B | Defines a number of pixels to be provided on the bottom of the graph. Time axes are shown 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 | Mouseover title text in svg (depending on browser, this may only work if you embed the svg in a page rather than as img tag) |
| e | No mouseover title text in svg |

I.4. Overnight archiving

The system is designed to make it easy to archive all graphs or svg, png, xml, etc files overnight. 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 overnight archiving of the previous day is done just after midnight.

The recommended command to run just after midnight is `wget --quiet --no-parent --mirror http://host:port/cqm/`date +%F -dyesterday`/z/` as this will create a directory for the server, cqm, date, and z, and then the files. The use of lowercase z clears text off the graphs to make them clean. Using uppercase z does SVG with no style included, so expecting to be used with a separate style sheet when presented.

The above `wget` command to fetch multiple files can only be used with trusted access. If you don't have any trusted IP addresses, then individual graphs must be requested separately, with each using a hash made from the graph name.

I.4.1. Full URL format

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

Table I.5. URL formats

| URL | Meaning |
|--------------------|--|
| /cqm/ | All CQM URLs start with this |
| 40-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 they all must have the same options, since 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 than & 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. If omitted, an index will be served (see below). |
| .ext | Extension for the file type required. Optional if no graph name is supplied. |
| ?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.

If a graph name is specified, then an extension must also be given.

A blank graph is available by accessing simply .png (i.e. an extension with 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.

I.4.2. 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.

I.5. 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.

I.6. 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.

Graphs can also be created dynamically in some cases, e.g. L2TP based graphs are made based on the Chargeable-User-Id, Calling-Station-Id or User-Name for a connected line, and so can be defined from the RADIUS authentication response. It is recommended that the circuit ID is used where available, e.g. from BT platform RADIUS.

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.

Appendix J. Hashed passwords

The configuration XML encodes passwords and OTP seeds using hashes, this makes it extremely difficult to extract them from the config.

Caution

It is still important to keep the configuration hashes safe, as someone could use the hashes to try millions of passwords off-line before trying to log in to a FireBrick. For this reason it is also important to use good passwords that cannot be guessed, and are not simply made from normal dictionary words.

J.1. Password hashing

The `user` section of the configuration has a `password` field. You will note that is it mostly a lot of hexadecimal data, the hash, as described below.

It is possible to put a new password into the configuration directly in the `password` field and save the config. It will be hashed automatically so when you access the configuration you will see the hashed version in the `password` field. However, this is not possible if you also have an `otp-seed` defined, unless also setting a new `otp-seed` field or removing it. For this reason there is also a web page to allow a user to change their password.

Tip

We recommend you use the web page to change the password for a user, and in fact, that the user themselves do this, so as the administrator does not know the password.

Note

Entering a password directly in the config does have a limit on the length of the password that can be accepted, though it is over 100 characters.

The FireBrick supports a number of hash functions for passwords, but on any successful login may change the config to use the current preferred password hash function. This allows FireBrick to move to more secure password hash functions in future whilst maintaining backward compatibility.

If making a configuration file independantly you can generate the hashes yourself in most cases. The supported hash codings are as follows. For salted hashes, the salt is the additional bytes after the number of bytes for the hash.

- `FB105#[10 bytes of hex]`: A legacy for the old FB105 password hashing, used by the FB105 conversion tool.
- `MD5#[16 to 19 bytes of hex]`: The first 16 bytes are an MD5 hash of the password appended with up to 3 bytes of salt.
- `SHA1#[20 to 31 bytes of hex]`: The first 20 bytes are an SHA1 hash of the password appended with up to 11 bytes of salt.
- `SHA256#[32 to 47 bytes of hex]`: The first 32 bytes are an SHA256 hash of the password appended with up to 15 bytes of salt.

The preferred hash is SHA256 with 15 bytes of salt. However, this may change in the future to more robust password functions.

J.1.1. Salt

A hash function simply takes some data, and generates a hash from it - as a *one-way* process. This ensures that, given the hash, you cannot work out the original string (normally a password).

However, a particular string (e.g. password) always generates the same hash. As such it is possible for people to have huge tables of pre-calculated hashes for common passwords and dictionary words. This allows such (poor) passwords to simply be looked up from a hash.

There is also the problem that two people using the same password end up with the same hash, and so can see that the other person is using the same password.

To solve this, we use *salt*. Salt is simply a number of random bytes. These are appended to the end of the original data before making a hash. This means the hash is not just of a password, but of a password and a random string (the salt).

The salt is stored, along with the hash. This means it is possible to check a password by appending the salt and calculating the hash and checking it matches the stored hash.

The password hashes all have an option of salt. If making your own password we recommend the latest hash method and as much salt as allowed. The salt should be random.

The OTP seed is scrambled using a separate salt on the same password, else the encryption would simply be using the hash you see in the `password` field, which would not be very secure!

J.2. One Time Password seed hashing

The configuration also holds the OTP seed used for One Time Password authenticator codes. However, this is stored in an encrypted format so that the seed cannot be accessed. This is especially important as the OTP seed allows the OTP sequence to be generated, not simply checked (as is the case with a password hash).

The issue with encrypting the OTP seed is that the FireBrick has to be able to decrypt it so as to check the OTP sequence used. To ensure that no secret encryption key is embedded in the FireBrick firmware, the encryption is done using the users password. Once again, this means that it is important to have a good password. This system means that the password hash and encrypted OTP seed can be saved, and restored and even moved to another FireBrick configuration if needed without ever having to know the seed or password itself.

Caution

This means that if someone knows (or finds out) the password and has access to the configuration file then they could extract the OTP seed and use it to log in, even though they do not have the OTP device/app. For this reason it is important to keep the password and OTP seed data in the configuration safe.

You can enter a new OTP seed into the `otp-seed` field in the config, if you wish. This should be a BASE32 string (which is the common format for usch strings). If the seed is for 60 second periods not the default 30 then append `/60`. If the seed is not for 6 digit codes, you can add a time (`/30` or `/60`) and then `/N` where N is the number of digits (4-8). Once saved you will see the seed changes to a base64 coded string. If you do this you should immediately test the authenticator by having the user log in. Until then, the seed is not encrypted in the configuration and could be recovered. Once you have logged in, if you normally save / archive the config, this would be a good time to ensure you have the encrypted version saved.

Note

The old OTP system used this field (called just `otp`) as the serial number of a separately stored OTP seed that was not held in the config. This is no longer supported, but if you have such a configuration you may see simply the serial number in this field until the user first logs in and it is replaced with the encrypted OTP seed.

Once encoded the format is a # followed by base64 coding of a series of bytes. If making a configuration file independantly then you can generate the seed data directly if you wish. The format is as follows.

- 1 byte: total length of data including this byte
- 1 byte: seed type `0xD`T where D=digits and T is time in 10 seconds, so `0x63` is normal for 6 digits every 30 seconds.

- 1 byte: hash type, 0-15 means not encrypted but space for up to 15 bytes of salt. 32-47 means SHA256 hash encrypted with up to 15 bytes of salt.
- N bytes: The OTP seed XOR with the hash made from the password with salt appended. If seed is longer than hash then only initial hash length bytes are XOR'd.
- S bytes: Seed bytes, should be random.
- 1 byte: 2's complement checksum making sum of all bytes 0x00.

Appendix K. Configuration Objects

This appendix defines the object definitions used in the FireBrick FB6202 L2TP configuration. Copyright © 2008-2023 FireBrick Ltd.

K.1. Top level

K.1.1. config: Top level config

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

Table K.1. config: Attributes

| Attribute | Type | Default | Description |
|-----------|----------|---------|---|
| ip | IPAddr | - | Config store IP address |
| patch | integer | - | Internal use, for s/w updates that change config syntax |
| serial | string | - | Serial number |
| timestamp | dateTime | - | Config store time, set automatically when config is saved |
| version | string | - | Code version |
| who | string | - | Config store username |

Table K.2. config: Elements

| Element | Type | Instances | Description |
|------------|-------------|----------------------|--|
| bgp | bgp | Optional, up to 100 | BGP config |
| bgp-filter | namedbgpmap | Optional, unlimited | Mapping and filtering rules for use with BGP peers |
| blackhole | blackhole | Optional, unlimited | Black hole (dropped packets) networks |
| cqm | cqm | Optional | Constant Quality Monitoring config |
| dhcp-relay | dhcp-relay | Optional, unlimited | DHCP server settings for remote / relayed requests |
| eap | eap | Optional, unlimited | User access control via EAP |
| 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 |
| l2tp | l2tp | Optional | L2TP settings |
| log | log | Optional, up to 63 | Log target controls |
| loopback | loopback | Optional, unlimited | Extra local addresses |
| network | network | Optional, unlimited | Locally originated networks |
| nowhere | blackhole | Optional, unlimited | Dead end (icmp error) networks |
| port | portdef | Optional, up to 2 | Port grouping and naming |
| ppp | pppoe | Optional, up to 50 | PPPoE settings |

| | | | |
|----------------|---------------|---------------------|-------------------------|
| profile | profile | Optional, unlimited | Control profiles |
| route | route | Optional, unlimited | Static routes |
| routing-tables | routing-table | Optional, unlimited | Routing table settings |
| sampling | sampling | Optional | Sampling parameters |
| services | services | Optional | General system services |
| shaper | shaper | Optional, unlimited | Named traffic shapers |
| system | system | Optional | System settings |
| user | user | Optional, unlimited | Admin users |

K.2. Objects

K.2.1. system: System settings

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

Table K.3. system: Attributes

| Attribute | Type | Default | Description |
|-------------------------|-----------------|--|--|
| acme-directory | string | https://acme-v02.api.letsencrypt.org/directory | ACME server directory |
| acme-hostname | List of string | - | Public hostname(s) for FireBrick for HTTPS |
| acme-keygen | boolean | true | Automatically obtain private keys as needed |
| acme-profile | NMTOKEN | - | Profile for when to do ACME renewals |
| acme-renew | positiveInteger | 30 | Renewal before expiry (days) |
| acme-source-ip | IP46Addr | - | Source IP for ACME renewal |
| acme-terms-agreed-email | string | - | Put your email if you agree CA terms |
| auto-backup-url | string | - | URL to http POST after config changed |
| comment | string | - | Comment |
| contact | string | - | Contact name |
| email | string | - | Contact email |
| eth-rx-batch | unsignedInt | 20 | Max packets serviced on one port before rechecking other port for idle |
| eth-rx-qsize | unsignedInt | 2000 | Size of eth driver Rx queue |
| eth-tx-qsize | unsignedInt | 2000 | Size of eth driver Tx queue |
| intro | string | - | Home page text |
| location | string | - | Location description |
| log | NMTOKEN | Web/console | Log system events |
| log-acme | NMTOKEN | - | Log ACME |
| log-acme-debug | NMTOKEN | - | Log ACME debug |
| log-acme-error | NMTOKEN | - | Log ACME errors |
| log-config | NMTOKEN | Web/Flash/console | Log config load |

| | | | |
|-------------------|--|-------------------|---|
| log-debug | NMTOKEN | Not logging | Log system debug messages |
| log-diagnostic | NMTOKEN | Not logging | Log system diagnostic 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-ppp-dump | ppp-dump | - | PPP dump format |
| log-route-nexthop | NMTOKEN | Not logged | Log next hop changes |
| log-stats | NMTOKEN | Not logging | Log one second stats |
| log-support | NMTOKEN | Web logs | Log support messages (e.g. stack trace) |
| log-tcp-debug | NMTOKEN | Not logging | Log TCP/TLS debug messages |
| login-intro | string | - | Login page text |
| name | string | - | System hostname |
| pre-reboot-url | string | - | URL to GET prior to s/w reboot (typically to warn nagios) |
| source | string | - | Source of data, used in automated config management |
| spoof-mac | (<i>hexBinary</i>) macspoof | - | Spoof MAC base address - use with caution! |
| sw-update | autoloadtype | false | Load new software automatically |
| sw-update-delay | (<i>unsignedByte 0-30</i>) fb-sw-update-delay | 0 | Number of days after release to wait before automatically upgrading |
| sw-update-profile | NMTOKEN | - | Profile name for when to load new s/w |
| table | (<i>unsignedByte 0-99</i>) routetable | 0 | Routing table number for system functions (s/w updates, etc) |

Table K.4. system: Elements

| Element | Type | Instances | Description |
|---------|------|---------------------|-------------|
| link | link | Optional, unlimited | Intro links |

K.2.2. link: Web links

Links to other web pages

Table K.5. link: Attributes

| Attribute | Type | Default | Description |
|-----------|------------|---------|---|
| comment | string | - | Comment |
| level | user-level | GUEST | Login level required |
| name | string | - | Link name |
| profile | NMTOKEN | - | Profile name |
| same-tab | boolean | false | Open in same tab |
| source | string | - | Source of data, used in automated config management |
| text | string | - | Link text |

| | | | |
|-----|--------|---|--------------|
| url | string | - | Link address |
|-----|--------|---|--------------|

K.2.3. routing-table: Default source IP for services using a given table

Default source IP for traffic originated by this FireBrick

Table K.6. routing-table: Attributes

| Attribute | Type | Default | Description |
|-----------|--|---------------------|--------------------------------|
| name | string | - | Name |
| source-ip | IP46Addr | - | Default source IP for services |
| table | (<i>unsignedByte 0-99</i>) routetable | <i>Not optional</i> | Routing table number |

K.2.4. user: Admin users

User names, passwords and abilities for admin users

Table K.7. user: Attributes

| Attribute | Type | Default | Description |
|------------|--|---------------------|--|
| allow | <i>List of</i> 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 |
| local-only | boolean | false | Restrict access to locally connected Ethernet subnets only |
| name | (<i>NMTOKEN</i>) username | <i>Not optional</i> | User name |
| otp-seed | OTP | - | OTP seed (do not edit by hand) |
| password | Password | <i>Not optional</i> | User password |
| profile | NMTOKEN | - | Profile name |
| source | string | - | Source of data, used in automated config management |
| table | (<i>unsignedByte 0-99</i>) routetable | 0 | Restrict login to specific routing table |
| timeout | duration | 5:00 | Login idle timeout (zero to stay logged in, not recommended) |

K.2.5. eap: User access controlled by EAP

Identities, passwords and access methods for access controlled with EAP

Table K.8. eap: Attributes

| Attribute | Type | Default | Description |
|-----------|------|---------|-------------|
|-----------|------|---------|-------------|

| | | | |
|-----------|--------------------------|---------------------|---|
| comment | string | - | Comment |
| full-name | string | - | Full name |
| methods | <i>Set of eap-method</i> | <i>Not optional</i> | Allowed methods |
| name | string | <i>Not optional</i> | User or account name |
| password | Secret | <i>Not optional</i> | User password |
| profile | NMTOKEN | - | Profile name |
| source | string | - | Source of data, used in automated config management |
| subsystem | eap-subsystem | <i>Not optional</i> | Access controlled subsystem |

K.2.6. log: Log target controls

Named logging target

Table K.9. 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 | <i>Not optional</i> | Log target name |
| profile | NMTOKEN | - | Profile name |
| source | string | - | Source of data, used in automated config management |
| system | boolean | - | Include system logs on web/cli view |

Table K.10. log: Elements

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

K.2.7. log-syslog: Syslog logger settings

Logging to a syslog server

Table K.11. log-syslog: Attributes

| Attribute | Type | Default | Description |
|-----------|-----------------|---------|------------------|
| comment | string | - | Comment |
| facility | syslog-facility | LOCAL0 | Facility setting |
| port | unsignedShort | 514 | Server port |
| profile | NMTOKEN | - | Profile name |

| | | | |
|-------------|--|---------------------|---|
| server | IPNameAddr | <i>Not optional</i> | 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 |
| system-logs | boolean | - | Include generic system log messages as well |
| table | (<i>unsignedByte 0-99</i>) routetable | 0 | Routing table number for sending syslogs |

K.2.8. log-email: Email logger settings

Logging to email

Table K.12. 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 |
| profile | NMTOKEN | - | Profile name |
| 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 | (<i>unsignedByte 0-99</i>) routetable | 0 | Routing table number for sending email |
| to | string | <i>Not optional</i> | Target email address |

K.2.9. 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 K.13. services: Elements

| Element | Type | Instances | Description |
|---------|------|-----------|-------------|
|---------|------|-----------|-------------|

| | | | |
|--------|----------------|----------|------------------------------|
| dns | dns-service | Optional | DNS service settings |
| http | http-service | Optional | Web server settings |
| radius | radius-service | Optional | RADIUS server/proxy settings |
| snmp | snmp-service | Optional | SNMP server settings |
| telnet | telnet-service | Optional | Telnet server settings |
| time | time-service | Optional | System time server settings |

K.2.10. http-service: Web service settings

Web management pages

Table K.14. http-service: Attributes

| Attribute | Type | Default | Description |
|-----------------------------|----------------------------|---------------------------|--|
| access-control-allow-origin | string | - | Additional HTTP header |
| allow | <i>List of IPNameRange</i> | Allow from anywhere | List of IP ranges from which service can be accessed |
| allow-acme | boolean | true | Allow limited port 80 HTTP access for ACME during renewal |
| banner-background | Colour | #bd1220 | Override default colours |
| certlist | <i>List of</i> NMTOKEN | use any suitable | Certificate(s) to be used for HTTPS sessions |
| comment | string | - | Comment |
| config-boxes | Colour | from banner | Config editor colours |
| content-security-policy | string | - | Additional HTTP header |
| css-url | string | - | Additional CSS for web control pages |
| highlight-text | Colour | from banner | Override default colours |
| https-port | unsignedShort | 443 | Service port for HTTPS access |
| js-url | string | - | Additional javascript for web control pages (logged in/trusted-ip) |
| local-only | boolean | true | Restrict access to locally connected Ethernet subnets only |
| log | NMTOKEN | Not logging | Log events |
| log-client | NMTOKEN | Not logging | Log client accesses |
| log-client-debug | NMTOKEN | Not logging | Log client accesses (debug) |
| log-debug | NMTOKEN | Not logging | Log debug |
| log-error | NMTOKEN | Log as event | Log errors |
| mode | http-mode | redirect-to-https-if-acme | Security mode |
| port | unsignedShort | 80 | Service port for HTTP access |
| profile | NMTOKEN | - | Profile name |
| referrer-policy | string | no-referrer | Additional HTTP header |
| self-sign | boolean | true | Create self signed certificate for HTTPS when necessary |

| | | | |
|------------------------|--|---------------|--|
| source | string | - | Source of data, used in automated config management |
| table | (<i>unsignedByte 0-99</i>) routetable | All | Routing table number for access to service |
| trusted | <i>List of</i> IPNameRange | - | List of allowed IP ranges from which additional access to certain functions is available |
| x-content-type-options | string | nosniff | Additional HTTP header |
| x-frame-options | string | SAMEORIGIN | Additional HTTP header |
| x-xss-protection | string | 1; mode=block | Additional HTTP header |

K.2.11. dns-service: DNS service settings

DNS forwarding resolver service

Table K.15. dns-service: Attributes

| Attribute | Type | Default | Description |
|-----------------|--|------------------------|---|
| allow | <i>List of</i> 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 |
| auto-dhcp-new | string | - | Name to use for last new DHCP allocation (since last reboot) |
| caching | boolean | true | Cache relayed DNS entries locally |
| comment | string | - | Comment |
| domain | string | - | Our domain |
| fallback | boolean | true | For incoming requests, if no server in required table, relay to any DNS available |
| fallback-table | (<i>unsignedByte 0-99</i>) routetable | Don't fallback | For incoming requests, if no server in requesting table, relay to any DNS available in this table |
| 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 |
| log-interface | <i>List of</i> NMTOKEN | All interfaces | Only do normal log for specific interface(s) |
| profile | NMTOKEN | - | Profile name |
| resolvers | <i>List of</i> IPAddr | - | Recursive DNS resolvers to use |
| resolvers-table | (<i>unsignedByte 0-99</i>) routetable | as table / 0 | Routing table for specified resolvers |
| source | string | - | Source of data, used in automated config management |
| table | (<i>unsignedByte 0-99</i>) routetable | All | Routing table number for access to service |

Table K.16. 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 |

K.2.12. dns-host: Fixed local DNS host settings

DNS forwarding resolver service

Table K.17. 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) |
| profile | NMTOKEN | - | Profile name |
| restrict-interface | List of NMTOKEN | - | Only apply on certain interface(s) |
| restrict-to | 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 |
| table | (unsignedByte 0-99) routetable | any | Routing table applicable |
| ttl | unsignedInt | 60 | Time to live |

K.2.13. dns-block: Fixed local DNS blocks

DNS forwarding resolver service

Table K.18. 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) |
| profile | NMTOKEN | - | Profile name |
| restrict-interface | List of NMTOKEN | - | Only apply on certain interface(s) |
| restrict-to | List of IPNameRange | - | List of IP ranges to which this is served |
| source | string | - | Source of data, used in automated config management |
| table | (unsignedByte 0-99) routetable | any | Routing table applicable |
| ttl | unsignedInt | 60 | Time to live |

K.2.14. radius-service: RADIUS service definition

RADIUS server and proxy definitions

Table K.19. radius-service: Attributes

| Attribute | Type | Default | Description |
|------------------------------|--------------------------------|--------------|--|
| acct-port | unsignedShort | 1813 | Accounting UDP port |
| allow | List of IPNameRange | - | Allowed source IP address of RADIUS request |
| aruba-vlan | (unsignedShort 0-4095) vlan | Don't send | Aruba VLAN |
| auth-port | unsignedShort | 1812 | Authentication UDP port |
| authenticator | boolean | - | Require message authenticator |
| backup-ip | List of IPNameAddr | - | Target IP(s) or hostname for backup L2TP connection |
| class | string | - | Class field to send |
| comment | string | - | Comment |
| control-port | unsignedShort | 3799 | Control UDP port (CoA/DM) |
| dummy-ip | boolean | true | Send dummy framed IP response |
| erx-egress-policy-name | string | - | Juniper attribute 11 |
| erx-ingress-policy-name | string | - | Juniper attribute 10 |
| erx-tunnel-switch-profile | string | - | Juniper attribute 91 |
| erx-tunnel-virtual-router | string | - | Juniper attribute 8 |
| erx-virtual-router-name | string | - | Juniper attribute 1 (Also SIN502 Context-Name) |
| log | NMTOKEN | Not logging | Log events |
| log-debug | NMTOKEN | - | Log debug |
| log-error | NMTOKEN | Log as event | Log errors |
| nsn-conditional | boolean | - | Only send NSN settings if username is not same as calling station id |
| nsn-tunnel-override-username | unsignedByte | - | Additional response for GGSN usage |
| nsn-tunnel-user-auth-method | unsignedInt | - | Additional response for GGSN usage |
| order | radiuspriority | - | Priority tagging of endpoints sent |
| profile | NMTOKEN | - | Profile name |
| reject | boolean | - | Reject request (rarely what you want) |
| relay-ip | List of IPAddr | - | Address to copy RADIUS request |
| relay-port | unsignedShort | 1812 | Authentication UDP port for copy RADIUS request |
| relay-table | (unsignedByte 0-99) routetable | - | Routing table number for copy of RADIUS request |
| secret | Secret | - | Shared secret for RADIUS requests (needed for replies) |

| | | | |
|----------------------|---------------------------|---|--|
| source | string | - | Source of data, used in automated config management |
| tagged | boolean | - | Tag all attributes that support tagging |
| target-hostname | string | - | Hostname for L2TP connection |
| target-ip | <i>List of IPNameAddr</i> | - | Target IP(s) or hostname for primary L2TP connection |
| target-secret | Secret | - | Shared secret for L2TP connection |
| tunnel-assignment-id | string | - | Tunnel Assignment ID to send |
| tunnel-client-return | boolean | - | Return tunnel client as radius IP |

Table K.20. radius-service: Elements

| Element | Type | Instances | Description |
|---------|----------------------|---------------------|---------------------------------------|
| match | radius-service-match | Optional, unlimited | Matching rules for specific responses |
| server | radius-server | Optional, unlimited | RADIUS server settings |

K.2.15. radius-service-match: Matching rules for RADIUS service

Rules for matching incoming RADIUS requests

Table K.21. radius-service-match: Attributes

| Attribute | Type | Default | Description |
|-------------------------|------------------------------------|------------|---|
| allow | <i>List of IPNameRange</i> | - | Allowed source IP address of RADIUS request |
| ap-group | <i>List of string</i> | - | One or more patterns to match AP Group |
| aruba-vlan | <i>(unsignedShort 0-4095) vlan</i> | Don't send | Aruba VLAN |
| authenticator | boolean | - | Require message authenticator |
| backup-ip | <i>List of IPNameAddr</i> | - | Target IP(s) or hostname for backup L2TP connection |
| called-station-id | <i>List of string</i> | - | One or more patterns to match called-station-id |
| calling-station-id | <i>List of string</i> | - | One or more patterns to match calling-station-id |
| class | string | - | Class field to send |
| comment | string | - | Comment |
| device-type | <i>List of string</i> | - | One or more patterns to match Device Type |
| dummy-ip | boolean | true | Send dummy framed IP response |
| erx-egress-policy-name | string | - | Juniper attribute 11 |
| erx-ingress-policy-name | string | - | Juniper attribute 10 |

Configuration Objects

| | | | |
|------------------------------|--------------------------------|-------------|--|
| erx-tunnel-switch-profile | string | - | Juniper attribute 91 |
| erx-tunnel-virtual-router | string | - | Juniper attribute 8 |
| erx-virtual-router-name | string | - | Juniper attribute 1 (Also SIN502 Context-Name) |
| essid-name | List of string | - | One or more patterns to match ESSID Name |
| ip | List of IPNameRange | - | Match target IP address of RADIUS request |
| location-id | List of string | - | One or more patterns to match Location ID |
| log | NMTOKEN | Not logging | Log events matching this |
| mac-local | boolean | - | Match only local or non local MAC addresses if username is a MAC |
| name | string | - | Name |
| nas-ip | List of IPNameRange | - | Match NAS-IP address in RADIUS request |
| nsn-conditional | boolean | - | Only send NSN settings if username is not same as calling station id |
| nsn-tunnel-override-username | unsignedByte | - | Additional response for GGSN usage |
| nsn-tunnel-user-auth-method | unsignedInt | - | Additional response for GGSN usage |
| order | radiuspriority | - | Priority tagging of endpoints sent |
| profile | NMTOKEN | - | Profile name |
| reject | boolean | - | Reject request (rarely what you want) |
| relay-ip | List of IPAddr | - | Address to copy RADIUS request |
| relay-port | unsignedShort | 1812 | Authentication UDP port for copy RADIUS request |
| relay-table | (unsignedByte 0-99) routetable | - | Routing table number for copy of RADIUS request |
| secret | Secret | - | Shared secret for RADIUS requests (needed for replies) |
| source | string | - | Source of data, used in automated config management |
| stop | boolean | true | Stop checking if this matches |
| tagged | boolean | - | Tag all attributes that support tagging |
| target-hostname | string | - | Hostname for L2TP connection |
| target-ip | List of IPNameAddr | - | Target IP(s) or hostname for primary L2TP connection |
| target-secret | Secret | - | Shared secret for L2TP connection |
| tunnel-assignment-id | string | - | Tunnel Assignment ID to send |
| tunnel-client-return | boolean | - | Return tunnel client as radius IP |
| username | List of string | - | One or more patterns to match username |

K.2.16. radius-server: RADIUS server settings

Server settings for outgoing RADIUS

Table K.22. radius-server: Attributes

| Attribute | Type | Default | Description |
|---------------|--------------------------------|-------------------------------|--|
| allow | List of IPNameRange | Must match host | Allowed control request source IPs instead of host check |
| comment | string | - | Comment |
| host | List of IPNameAddr | Not optional | One or more hostname/IPs of RADIUS servers |
| max-timeout | duration | 10 | Maximum final timeout |
| min-timeout | duration | 2 | Minimum final timeout |
| name | string | - | Name |
| port | unsignedShort | From services/radius settings | UDP port |
| profile | NMTOKEN | - | Profile name |
| queue | unsignedInt | - | Concurrent requests over all of these servers (per type) |
| scale-timeout | unsignedByte | 2 | Timeout scaling factor |
| secret | Secret | Not optional | Shared secret for RADIUS requests |
| source | string | - | Source of data, used in automated config management |
| source-ip | IPAddr | - | Fix source IP |
| table | (unsignedByte 0-99) routetable | - | Routing table number |
| type | Set of radiustype | All | Server type |

K.2.17. telnet-service: Telnet service settings

Telnet control interface

Table K.23. 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 |
| profile | NMTOKEN | - | Profile name |
| prompt | string | system name | Prompt |

| | | | |
|--------|--|-----|---|
| source | string | - | Source of data, used in automated config management |
| table | (<i>unsignedByte 0-99</i>) routetable | All | Routing table number for access to service |

K.2.18. snmp-service: SNMP service settings

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

Table K.24. snmp-service: Attributes

| Attribute | Type | Default | Description |
|------------|--|----------------|--|
| allow | List of IPNameRange | Allow anywhere | List of IP ranges from which service can be accessed |
| comment | string | - | Comment |
| community | Secret | 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 |
| profile | NMTOKEN | - | Profile name |
| source | string | - | Source of data, used in automated config management |
| table | (<i>unsignedByte 0-99</i>) routetable | All | Routing table number for access to service |

K.2.19. time-service: System time server settings

The time settings define which NTP servers to synchronize the system clock from, and provide controls for daylight saving (summer time). The defaults are those that apply to the EU

Table K.25. time-service: Attributes

| Attribute | Type | Default | Description |
|-------------------|---------------------|----------------|--|
| allow | List of IPNameRange | Allow anywhere | List of IP ranges from which service can be accessed |
| comment | string | - | Comment |
| legacy-timeserver | boolean | true | Serve legacy TIME service on UDP port 37 |
| 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 |
| maxpoll | duration | 1024 | NTP maximum poll rate |
| minpoll | duration | 64 | NTP minimum poll rate |

| | | | | | |
|------------------------|--------------------------------|----|----------------------|------|--|
| ntp-control-allow | List of IPNameRange | of | Allow anywhere | from | List of IP ranges from which control (ntp) requests can be accessed |
| ntp-control-local-only | boolean | | true | | Restrict control (ntp) access to locally connected Ethernet subnets only |
| ntp-control-table | (unsignedByte 0-99) routetable | | All | | Routing table number for incoming control (ntp) requests |
| ntp-peer-table | (unsignedByte 0-99) routetable | | 0 | | Routing table number used for outgoing ntp peer requests |
| ntp-servers | List of IPNameAddr | | ntp.firebrick.ltd.uk | | List of NTP time servers (IP or hostname) from which time may be synchronized and served by ntp (Null list disables NTP) |
| profile | NMTOKEN | | - | | Profile name |
| source | string | | - | | Source of data, used in automated config management |
| table | (unsignedByte 0-99) routetable | | All | | Routing table number for access to service |
| 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 |

K.2.20. ethernet: Physical port controls

Physical port attributes

Table K.26. ethernet: Attributes

| Attribute | Type | Default | Description |
|-----------|-----------|---------------|-------------------------------|
| autoneg | boolean | true | Perform link auto-negotiation |
| clocking | LinkClock | prefer-slave | Gigabit clock setting |
| crossover | Crossover | auto | Port crossover configuration |
| flow | LinkFlow | none | Flow control setting |
| green | LinkLED-g | Link/Activity | Green LED setting |
| lacp | boolean | Auto | Send LACP packets |
| lldp | boolean | true | Send LLDP packets |
| optimise | boolean | true | enable PHY optimisations |

| | | | |
|--------------|-----------|---------------------|-------------------------|
| port | port | <i>Not optional</i> | Physical port |
| power-saving | LinkPower | full | enable PHY power saving |
| profile | NMTOKEN | - | Profile name |
| send-fault | LinkFault | - | Send fault status |
| yellow | LinkLED-y | Tx | Yellow LED setting |

K.2.21. sampling: Packet sampling configuration

Packet sampling configuration

Table K.27. sampling: Attributes

| Attribute | Type | Default | Description |
|------------------|--|--------------------------------|---|
| agent-ip | IPAddr | use source-ip | IP address used to identify this agent |
| collector-ip | IPAddr | <i>Not optional</i> | IP address of collector |
| collector-port | unsignedShort | 6343 for sFlow, 4739 for IPFIX | UDP port which collector listens on |
| comment | string | - | Comment |
| mtu | <i>(unsignedShort 576-2000)</i> mtu | 1500 | |
| name | string | - | Name |
| profile | NMTOKEN | - | Profile name |
| protocol | sampling-protocol | sflow | Protocol used to export sampling data |
| sample-flush | duration | 1 sec for sFlow; 30 for IPFIX | Sample max cache time |
| sample-rate | <i>(unsignedShort 100-10000)</i> sample-rate | 1000 | Sample rate (uniform random prob 1/N) |
| snap-length | unsignedShort | 64 | Packet header snap length |
| source | string | - | Source of data, used in automated config management |
| source-ip | IPAddr | - | Source IP address to use |
| source-port | unsignedShort | Use collector-port | UDP source port |
| stats-interval | duration | 60 | Stats export interval |
| table | <i>(unsignedByte 0-99)</i> routetable | 0 | Routing table number for sample data |
| template-refresh | duration | 600 | Template resend interval |

K.2.22. portdef: Port grouping and naming

Port grouping and naming

Table K.28. portdef: Attributes

| Attribute | Type | Default | Description |
|-----------|---------|---------------------|-------------|
| comment | string | - | Comment |
| name | NMTOKEN | <i>Not optional</i> | Name |

| | | | |
|--------|-------------|--------------|---|
| ports | Set of port | Not optional | Physical port(s) |
| source | string | - | Source of data, used in automated config management |
| trunk | trunk-mode | l2-hash | Trunk ports |

K.2.23. 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 K.29. interface: Attributes

| Attribute | Type | Default | Description |
|--------------|------------------------------|---|---|
| allow-6in4 | boolean | false | Handle 6in4 (protocol 41) packets |
| bgp | bgpmode | Auto | BGP announce mode for routes |
| comment | string | - | Comment |
| cug | (unsignedShort 1-32767) cug | - | Closed user group ID |
| cug-restrict | boolean | - | Closed user group restricted traffic (only to/from same CUG ID) |
| dhcp-relay | IP4Addr | - | Relay any unresolved requests to external server |
| fast-l2tp | boolean | - | Set on interfaces that are mainly terminating L2TP traffic |
| graph | (token) graphname | - | Graph name |
| link | NMTOKEN | - | Interface to which this is linked at layer 2 |
| log | NMTOKEN | Not logging | Log events |
| log-debug | NMTOKEN | Not logging | Log debug |
| log-dhcp | NMTOKEN | Not logging | Log DHCP events not related to a pool |
| log-error | NMTOKEN | Log as event | Log errors |
| mac-suffix | (hexBinary) macsuffix | - | Interface MAC ends with this hex value |
| mtu | (unsignedShort 576-2000) mtu | 1500 | MTU for this interface |
| name | NMTOKEN | - | Name |
| pd | boolean | If not WAN and no ra-subnet-templates and no ra subnets | Available for IPv6 prefix delegation |
| ping | IPAddr | - | Ping address to add loss/latency to graph for interface |
| port | NMTOKEN | Not optional | Port group name |
| profile | NMTOKEN | - | Profile name |
| restrict-mac | boolean | - | Use only one MAC on this interface |
| sampling | sampling-mode | off | Perform sampling |
| source | string | - | Source of data, used in automated config management |

| | | | |
|---------------------|--|-----------------|--|
| source-filter | sfoption | - | Source filter traffic received via this interface |
| source-filter-table | (<i>unsignedByte 0-99</i>) routetable | interface table | Routing table to use for source filtering checks |
| table | (<i>unsignedByte 0-99</i>) routetable | 0 | Routing table applicable |
| vlan | (<i>unsignedShort 0-4095</i>) vlan | 0 | VLAN ID (0=untagged) |
| wan | boolean | - | Do not consider this interface 'local' for 'local-only' checks |

Table K.30. interface: Elements

| Element | Type | Instances | Description |
|--------------------|-----------------|---------------------|------------------------------|
| dhcp | dhcps | Optional, unlimited | DHCP server settings |
| ra-subnet-template | subnet-template | Optional, unlimited | Subnet options for RA client |
| subnet | subnet | Optional, unlimited | IP subnet on the interface |
| vrrp | vrrp | Optional, unlimited | VRRP settings |

K.2.24. subnet: Subnet settings

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

Table K.31. 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 |
| bgp | bgpmode | Auto | BGP announce mode for routes |
| broadcast | boolean | false | If broadcast address allowed |
| comment | string | - | Comment |
| dhcp-class | string | FB-type | DHCP client option 60 (Class) |
| dhcp-client-id | string | MAC | DHCP client option 61 (Client-Identifier) |
| gateway | <i>List of</i> IPAddr | - | One or more gateways to install |
| ip | <i>List of</i> IPSubnet | Automatic by DHCP | One or more IP/len |
| localpref | unsignedInt | 4294967295 | Localpref for subnet (highest wins) |
| mac-suffix | (<i>hexBinary</i>) macsuffix | - | Subnet MAC ends with this hex value |
| mtu | (<i>unsignedShort 576-2000</i>) mtu | As interface | MTU for subnet |
| name | string | - | Name |
| profile | NMTOKEN | - | Profile 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-autonomous | boolean | If managed not set | RA 'A' (autonomous) flag |

| | | | |
|---------------|-------------------------------|-----------|--|
| ra-dns | List of IP6Addr | Our IP | List of recursive DNS servers in route announcements |
| ra-dnssl | List of string | - | List of DNS search domains in route announcements |
| ra-managed | boolean | - | RA 'M' (managed) flag |
| ra-max | (unsignedShort 4-1800) ra-max | 600 | Max RA send interval |
| ra-min | (unsignedShort 3-1350) ra-min | ra-max/3 | Min RA send interval |
| ra-mtu | unsignedShort | As subnet | MTU to use on RA |
| ra-onlink | boolean | true | RA 'L' (onlink) flag |
| ra-other | boolean | - | RA 'O' (other) flag |
| ra-profile | NMTOKEN | - | Profile, if inactive then forces low priority RA |
| simple-dhcpv6 | boolean | - | Simple DHCPv6 server (fixed addresses) |
| 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 |

K.2.25. subnet-template: Subnet option templates for RA

Table K.32. subnet-template: Attributes

| Attribute | Type | Default | Description |
|---------------|---------------------|---------------------------|--|
| accept-dns | boolean | True if not set elsewhere | Accept DNS servers specified by DHCP/SLAAC |
| comment | string | - | Comment |
| gateway-match | List of IPNameRange | Any IP | Apply only to received RAs with a gateway in these IPs |
| name | string | - | Name |
| profile | NMTOKEN | - | Profile name |
| source | string | - | Source of data, used in automated config management |

K.2.26. vrrp: VRRP settings

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

Table K.33. 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 |
| profile | NMTOKEN | - | Profile name |
| source | string | - | Source of data, used in automated config management |
| 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 |

K.2.27. dhcps: DHCP server settings

Settings for DHCP server

Table K.34. dhcps: Attributes

| Attribute | Type | Default | Description |
|---------------|------------------|----------------------|--|
| boot | IP4Addr | - | Next/boot server |
| boot-file | string | - | Boot filename |
| broadcast | boolean | - | Broadcast replies even if not requested |
| circuit | string | - | Agent info circuit match |
| class | string | - | Vendor 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 |
| domain-search | string | - | DNS domain search list (list will be truncated to fit one attribute) |
| force | boolean | - | Send all options even if not requested |
| gateway | IP4Subnet | 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 |
| log-decline | NMTOKEN | Not logging | Log events (declined) |
| log-move | NMTOKEN | Not logging | Log events (moved) |

| | | | |
|-------------|-------------------------------------|-------------|---|
| log-new | NMTOKEN | Not logging | Log events (new) |
| log-release | NMTOKEN | Not logging | Log events (released) |
| log-renew | NMTOKEN | Not logging | Log events (renewed) |
| log-reuse | NMTOKEN | Not logging | Log events (reused) |
| mac | List up to 12 (hexBinary) macprefix | - | Partial or full client hardware (MAC) addresses (or client-id MAC if specified) |
| mac-local | boolean | - | Match only local or non local MAC addresses |
| name | string | - | Name |
| ntp | List of IP4Addr | Our IP | NTP server |
| profile | NMTOKEN | - | Profile name |
| 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 K.35. dhcp: 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) |

K.2.28. dhcp-attr-hex: DHCP server attributes (hex)

Additional DHCP server attributes (hex)

Table K.36. dhcp-attr-hex: Attributes

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

K.2.29. dhcp-attr-string: DHCP server attributes (string)

Additional DHCP server attributes (string)

Table K.37. dhcp-attr-string: Attributes

| Attribute | Type | Default | Description |
|-----------|------|---------|-------------|
|-----------|------|---------|-------------|

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

K.2.30. dhcp-attr-number: DHCP server attributes (numeric)

Additional DHCP server attributes (numeric)

Table K.38. dhcp-attr-number: Attributes

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

K.2.31. dhcp-attr-ip: DHCP server attributes (IP)

Additional DHCP server attributes (IP)

Table K.39. dhcp-attr-ip: Attributes

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

K.2.32. pppoe: PPPoE settings

PPPoE endpoint settings

Table K.40. pppoe: Attributes

| Attribute | Type | Default | Description |
|-----------|--------|---|--------------------------|
| ac-name | string | Any a/c name as client, else same as 'name' | Access concentrator name |

Configuration Objects

| | | | |
|----------------|--|---------------------|---|
| accept-dns | boolean | true | Accept DNS servers specified by far end |
| auto-percent | unsignedByte | N/A | Try to set egress based on connect message, percentage |
| bgp | bgpmode | Auto | BGP announce mode for routes |
| calling-id | pppoe-calling | - | Add mac and/or vlan(s) after prefix |
| calling-prefix | string | - | Prefix on calling number (BRAS mode) |
| comment | string | - | Comment |
| cug | <i>(unsignedShort 1-32767)</i> cug | - | Closed user group ID |
| cug-restrict | boolean | - | Closed user group restricted traffic (only to/from same CUG ID) |
| eth | port | - | Physical port connected to modem (for port reset) |
| fast-retry | boolean | - | Aggressive re-connect |
| graph | <i>(token)</i> graphname | - | Graph name |
| incoming-vlans | <i>List of (unsignedShort 0-4095)</i> vlan | - | VLAN IDs to accept connections on |
| ip-over-lcp | boolean | auto | Sends all IP packets as LCP |
| lcp-rate | unsignedByte | 10 | LCP interval (seconds) |
| lcp-timeout | unsignedByte | 61 | LCP timeout (seconds) |
| local | IP4Addr | - | Local IPv4 address |
| localpref | unsignedInt | 4294967295 | Localpref for route (highest wins) |
| log | NMTOKEN | Not logging | Log events |
| log-debug | NMTOKEN | Not logging | Log debug |
| log-error | NMTOKEN | Not logging | Log as events |
| mac-suffix | <i>(hexBinary)</i> macsuffix | - | MAC ends with this hex value |
| mode | pppoe-mode | client | PPPoE server/client mode |
| mtu | <i>(unsignedShort 576-2000)</i> mtu | 1492 | MTU for link |
| name | NMTOKEN | - | Name |
| password | Secret | - | User password |
| port | NMTOKEN | <i>Not optional</i> | Port group name |
| profile | NMTOKEN | - | Profile name |
| remote | IP4Addr | - | Remote IPv4 address |
| rfc4638 | boolean | If over 1492 MTU | Send RFC4638 PPP-Max-Payload |
| routes | <i>List of</i> IPPrefix | Default gateway | Routes when link up |
| service | string | Any service | Service name |
| source | string | - | Source of data, used in automated config management |
| speed | unsignedInt | - | Default egress rate limit (b/s) |
| table | <i>(unsignedByte 0-99)</i> routetable | - | Routing table number for payload |

| | | | |
|-------------|-------------------------------------|------|---|
| tcp-mss-fix | boolean | true | Adjust MSS option in TCP SYN to fix session MSS |
| username | string | - | User name |
| vlan | (<i>unsignedShort</i> 0-4095) vlan | 0 | VLAN ID (0=untagged) |

Table K.41. pppoe: Elements

| Element | Type | Instances | Description |
|---------|-----------|---------------------|-------------------------------------|
| route | ppp-route | Optional, unlimited | Routes to apply when ppp link is up |

K.2.33. ppp-route: PPP routes

Routes that apply when link is up

Table K.42. ppp-route: Attributes

| Attribute | Type | Default | Description |
|-----------|------------------|--------------|---|
| bgp | bgpmode | Auto | BGP announce mode for routes |
| 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 |
| profile | NMTOKEN | - | Profile name |
| source | string | - | Source of data, used in automated config management |

K.2.34. 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 K.43. route: Attributes

| Attribute | Type | Default | Description |
|-----------|----------------------------|--------------|---|
| as-path | List up to 10 unsignedInt | - | Custom AS path as if network received |
| bgp | bgpmode | Auto | BGP announce mode for routes |
| comment | string | - | Comment |
| gateway | List of IPAddr | Not optional | One or more target gateway IPs |
| graph | (<i>token</i>) 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 |
| profile | NMTOKEN | - | Profile 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 |
| tag | List of Community | - | List of community tags |

K.2.35. network: Locally originated networks

Network blocks that are announced but not actually added to internal routes - note that blackhole and nowhere objects can also announce but not add routing.

Table K.44. network: Attributes

| Attribute | Type | Default | Description |
|-----------|-----------------------------------|--------------|---|
| as-path | List up to 10 unsignedInt | - | Custom AS path as if network received |
| bgp | bgpmode | true | BGP announce mode for routes |
| 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 |
| profile | NMTOKEN | - | Profile name |
| source | string | - | Source of data, used in automated config management |
| table | (unsignedByte 0-99) routetable | 0 | Routing table number |
| tag | List of Community | - | List of community tags |

K.2.36. blackhole: Dead end networks

Networks that go nowhere

Table K.45. blackhole: Attributes

| Attribute | Type | Default | Description |
|-----------|-----------------------------------|--------------|---|
| as-path | List up to 10 unsignedInt | - | Custom AS path as if network received |
| bgp | bgpmode | false | BGP announce mode for routes |
| 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 |
| no-fib | boolean | - | Route not in forwarding, only for EBGp |
| profile | NMTOKEN | - | Profile name |
| source | string | - | Source of data, used in automated config management |
| table | (unsignedByte 0-99) routetable | 0 | Routing table number |
| tag | List of Community | - | List of community tags |

K.2.37. loopback: Locally originated networks

Loopback addresses define local IP addresses

Table K.46. loopback: Attributes

| Attribute | Type | Default | Description |
|-----------|--------------------------------|--------------|---|
| as-path | List up to 10 unsignedInt | - | Custom AS path as if network received |
| bgp | bgpmode | Auto | BGP announce mode for routes |
| 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 |
| profile | NMTOKEN | - | Profile name |
| source | string | - | Source of data, used in automated config management |
| table | (unsignedByte 0-99) routetable | 0 | Routing table number |
| tag | List of Community | - | List of community tags |

K.2.38. namedbgpmap: Mapping and filtering rules of BGP prefixes

This defines a set of named rules for mapping and filtering of prefixes to/from a BGP peer.

Table K.47. namedbgpmap: Attributes

| Attribute | Type | Default | Description |
|-----------|---------|--------------|---|
| comment | string | - | Comment |
| name | NMTOKEN | Not optional | Name |
| source | string | - | Source of data, used in automated config management |

Table K.48. namedbgpmap: Elements

| Element | Type | Instances | Description |
|---------|---------|---------------------|----------------------------------|
| match | bgprule | Optional, unlimited | List rules, in order of checking |

K.2.39. bgprule: Individual mapping/filtering rule

An individual rule for BGP mapping/filtering

Table K.49. bgprule: Attributes

| Attribute | Type | Default | Description |
|------------|-------------|---------|--|
| as-origin | unsignedInt | - | AS that must be last in path to match |
| as-present | unsignedInt | - | AS that must be present in path to match |
| comment | string | - | Comment |

| | | | |
|--------------|-------------------|---|--|
| community | Community | - | Community that must be present to match |
| detag | List of Community | - | List of community tags to remove |
| drop | boolean | - | Do not import/export this prefix |
| localpref | unsignedInt | - | Set localpref (highest wins) |
| med | unsignedInt | - | Set MED |
| name | string | - | Name |
| no-community | Community | - | Community that must not be present to match |
| pad | unsignedByte | - | Pad (prefix stuff) our AS on export by this many, can be zero to not send our AS |
| prefix | List of IPFilter | - | Prefixes that this rule applies to |
| source | string | - | Source of data, used in automated config management |
| tag | List of Community | - | List of community tags to add |

K.2.40. bgp: Overall BGP settings

The BGP element defines general BGP settings and a list of peer definitions for the individual BGP peers.

Table K.50. bgp: Attributes

| Attribute | Type | Default | Description |
|---------------------|-----------------------------------|-------------|---|
| as | unsignedInt | - | Our AS |
| blackhole-community | Community | - | Community tag to mark black hole routes |
| cluster-id | IP4Addr | - | Our cluster ID |
| comment | string | - | Comment |
| dead-end-community | Community | - | Community tag to mark dead end routes |
| greyhole-community | Community | - | Community tag to mark black hole routes with no-fib |
| id | IP4Addr | - | Our router ID |
| log | NMTOKEN | Not logging | Log events |
| name | string | - | Name |
| source | string | - | Source of data, used in automated config management |
| table | (unsignedByte 0-99) routetable | 0 | Routing table number |

Table K.51. bgp: Elements

| Element | Type | Instances | Description |
|---------|---------|---------------------|--------------------------|
| peer | bgppeer | Optional, up to 500 | List of peers/neighbours |

K.2.41. bgppeer: BGP peer definitions

The peer definition specifies the attributes of an individual peer. Multiple IP addresses can be specified, typically for IPv4 and IPv6 addresses for the same peer, but this can be used for a group of similar peers.

Table K.52. bgppeer: Attributes

| Attribute | Type | Default | Description |
|-----------------------------|-------------------|--|---|
| add-own-as | boolean | - | Add our AS on exported routes |
| allow-export | boolean | true for customer | Ignore no-export community and export anyway |
| allow-only-their-as | boolean | - | Only accept routes that are solely the peers AS |
| allow-own-as | boolean | - | Allow our AS inbound |
| as | unsignedInt | - | Peer AS |
| blackhole-community | Community | Not announced on EBGP, our blackhole-community if IBGP | Egress community tag to mark black hole routes |
| capability-as4 | boolean | true | If supporting AS4 |
| capability-graceful-restart | boolean | true | If supporting Graceful Restart |
| capability-mpe-ipv4 | boolean | true | If supporting MPE for IPv4 |
| capability-mpe-ipv6 | boolean | true | If supporting MPE for IPv6 |
| capability-route-refresh | boolean | true | If supporting Route Refresh |
| clean-shutdown-wait | duration | - | Send peers low priority and delay on shutdown |
| clean-startup-wait | duration | - | Don't announce routes within this time of reboot |
| comment | string | - | Comment |
| drop-default | boolean | false | Ignore default route received |
| export-filters | List of NMTOKEN | - | Named export filters to apply |
| export-med | unsignedInt | - | Set MED on exported routes (unless export filter sets it) |
| holdtime | unsignedInt | 30 | Hold time |
| ignore-bad-optional-partial | boolean | true | Ignore routes with a recognised badly formed optional that is flagged partial |
| import-filters | List of NMTOKEN | - | Named import filters to apply |
| import-localpref | unsignedInt | - | Set localpref on imported routes (unless import filter sets it) |
| import-tag | List of Community | - | List of community tags to add in addition to any import filters |
| in-soft | boolean | - | Mark received routes as soft |
| ip | List of IPAddr | - | One or more IPs of neighbours (omit to allow incoming) |
| log-debug | NMTOKEN | Not logging | Log debug |
| max-prefix | unsignedInt | - | Limit prefixes (IPv4+IPv6) |
| md5 | Secret | - | MD5 signing secret |
| name | string | - | Name |
| next-hop-self | boolean | false | Force us as next hop outbound |

| | | | |
|------------------|---------------|-------------------------------|---|
| no-fib | boolean | - | Don't include received routes in packet forwarding |
| pad | unsignedByte | - | Pad (prefix stuff) our AS on export by this many |
| profile | NMTOKEN | - | Profile name |
| reduce-recursion | boolean | false | Override incoming next hop if not local subnet |
| restart-time | unsignedShort | - | Time to tell other end to expect us to take to restart (defaults to holdtime) |
| same-ip-type | boolean | true | Only accept/send IPv4 routes to IPv4 peers and IPv6 routes to IPv6 peers |
| send-default | boolean | false | Send a default route to this peer |
| send-no-routes | boolean | false | Don't send any normal routes |
| source | string | - | Source of data, used in automated config management |
| timer-idle | unsignedInt | 60 | Idle time after error |
| timer-openwait | unsignedInt | 10 | Time to wait for OPEN on connection |
| timer-retry | unsignedInt | 10 | Time to retry the neighbour |
| ttl-security | byte | - | Enable RFC5082 TTL security (if +ve, 1 to 127), i.e. 1 for adjacent router. If -ve (-1 to -128) set forced sending TTL, i.e. -1 for TTL of 1 sending, and not checking. |
| type | peertype | normal | Type of neighbour (affects some defaults) |
| use-vrrp-as-self | boolean | true if customer/transit type | Use VRRP address as self if possible |

Table K.53. bgppeer: Elements

| Element | Type | Instances | Description |
|---------|--------|-----------|---|
| export | bgpmap | Optional | Mapping and filtering rules of announcing prefixes to peer |
| import | bgpmap | Optional | Mapping and filtering rules of accepting prefixes from peer |

K.2.42. bgpmap: Mapping and filtering rules of BGP prefixes

This defines the rules for mapping and filtering of prefixes to/from a BGP peer.

Table K.54. bgpmap: Attributes

| Attribute | Type | Default | Description |
|-----------|-------------------|---------|----------------------------------|
| comment | string | - | Comment |
| detag | List of Community | - | List of community tags to remove |
| drop | boolean | - | Do not import/export this prefix |
| localpref | unsignedInt | - | Set localpref (highest wins) |
| med | unsignedInt | - | Set MED |

| | | | |
|--------|-------------------|---|---|
| prefix | List of IPFilter | - | Drop all that are not in this prefix list |
| source | string | - | Source of data, used in automated config management |
| tag | List of Community | - | List of community tags to add |

Table K.55. bgpmap: Elements

| Element | Type | Instances | Description |
|---------|---------|---------------------|----------------------------------|
| match | bgprule | Optional, unlimited | List rules, in order of checking |

K.2.43. 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 K.56. cqm: Attributes

| Attribute | Type | Default | Description |
|-------------------|--------------|----------|---|
| auto-refresh-list | boolean | true | Auto refresh graph list pages (for trusted IPs) |
| 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 | Ave | Label for average latency |
| label-damp | string | Damp% | Label for % shaper damping |
| 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 |

Configuration Objects

| | | | |
|-----------------|-----------------|-----------------|---|
| 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-shaper | string | Shaper | Label for shaper |
| 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 |
| marker-width | string | - | Stroke width for marker (+) on tx/rx (e.g. 4) |
| 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 |
| pppoe-dos-limit | unsignedInt | 10000 | Per poll tx packet drop limit for DOS protection on PPPoE incoming sessions |
| 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 SHA1 coded URLs |
| sent | Colour | #ff8 | Colour for polled seconds |
| share-interface | NMTOKEN | - | Interface on which to broadcast data for shaper sharing |
| share-secret | Secret | - | Secret to validate shaper sharing |
| stroke-width | string | 4 if no marker | Stroke line for tx/rx |
| subheading | string | - | Subheading of graph |
| svg-css | string | - | URL for SVG CSS instead of local style settings |
| svg-title | boolean | - | Include mouseover title text on svg |
| 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 |

K.2.44. l2tp: L2TP settings

L2TP settings for incoming L2TP connections

Table K.57. l2tp: Attributes

| Attribute | Type | Default | Description |
|---------------------|----------|---------|--|
| accounting-interval | duration | 1:00:00 | Periodic interim accounting interval |
| send-acct-delay | boolean | - | Send Acct-Delay as well as Event-Timestamp on accounting |

Table K.58. l2tp: Elements

| Element | Type | Instances | Description |
|----------|---------------|---------------------|---------------------------|
| incoming | l2tp-incoming | Optional, unlimited | Incoming L2TP connections |

K.2.45. l2tp-incoming: L2TP settings for incoming L2TP connections

L2TP tunnel settings for incoming L2TP connections

Table K.59. l2tp-incoming: Attributes

| Attribute | Type | Default | Description |
|--------------|--------------------------------------|---------------------|--|
| advise-speed | unsignedInt | - | Advise clients of their egress rate (may be overridden by RADIUS) (b/s) - This is a FireBrick specific mechanism |
| allow | <i>List of</i> IPNameRange <i>of</i> | - | List of IP ranges from which connects can be made |
| bgp | bgpmode | Auto | BGP announce mode for routes |
| comment | string | - | Comment |
| damping | boolean | false | Apply damping to sessions if limiting on shaper |
| dhcpv6dns | <i>List of</i> IP6Addr | System resolvers | DNS List of IPv6 DNS servers |
| dos-limit | unsignedInt | 10000 | Per second per session tx packet drop limit for DOS protection |
| fail-lockout | unsignedByte | 60 | Interval kept in failed state |
| graph | string | - | Graph name |

Configuration Objects

| | | | |
|---------------------|---------------------------------------|--------------|--|
| hdlc | boolean | true | Send HDLC header (FF03) on all PPP frames |
| hello-interval | unsignedByte | 60 | Interval between HELLO messages |
| icmp-ppp | boolean | false | Use PPP endpoint for ICMP |
| ipv6ep | IP4Addr | - | Local end IPv4 for IPv6 tunnels |
| lcp-data-len | unsignedByte | - | LCP data field length |
| lcp-mru-fix | boolean | false | Restart LCP if RAS negotiated MRU is too high |
| lcp-rate | unsignedByte | 1 | LCP interval (seconds) |
| lcp-timeout | unsignedByte | 10 | LCP timeout (seconds) |
| local-hostname | string | System name | Hostname quoted on reply |
| local-ppp-ip | IP4Addr | - | Local end PPP IPv4 |
| log | NMTOKEN | Not logging | Log events |
| log-debug | NMTOKEN | Not logging | Log debug |
| log-error | NMTOKEN | Log as event | Log errors |
| mtu | <i>(unsignedShort 576-2000)</i> mtu | - | Default MTU for sessions in this tunnel |
| name | string | - | Name |
| open-timeout | unsignedByte | 60 | Interval before OPEN considered failed |
| payload-table | <i>(unsignedByte 0-99)</i> routetable | 0 | Routing table number for payload traffic |
| ppp-final-timeout | unsignedByte | 60 | PPP total timeout (seconds) |
| ppp-init-timeout | unsignedByte | 10 | PPP initial timeout (seconds) |
| pppdns1 | IP4Addr | - | PPP DNS1 IPv4 default |
| pppdns2 | IP4Addr | - | PPP DNS2 IPv4 default |
| profile | NMTOKEN | - | Profile name |
| radius | string | - | Name for RADIUS server config to use |
| radius-nas-ip | radius-nas | lac | Pass remote (LAC) or local (LNS) as RADIUS NAS IP / port |
| receive-window | unsignedShort | Not sent | Receive window to advise on connection |
| remote-hostname | string | - | Hostname expected on connection |
| require-platform | boolean | false | All sessions require a platform RADIUS first |
| require-radius-acct | boolean | - | Close session if cannot do RADIUS accounting |
| retry-timeout | unsignedByte | 60 | Interval to retry sending control messages before fail |
| secret | Secret | - | Shared secret (for far end to check) |
| session-timeout | duration | - | Default session timeout |
| shutdown | boolean | false | Refuse all new sessions or tunnels |
| source | string | - | Source of data, used in automated config management |
| source-ip | IPAddr | - | IP of our end for relayed (on same table) |

| | | | |
|-------------|--|-------|---|
| speed | unsignedInt | - | Default egress rate limit (b/s) |
| table | (<i>unsignedByte 0-99</i>) routetable | Any | Routing table number for L2TP session |
| tcp-mss-fix | boolean | false | Adjust MSS option in TCP SYN to fix session MSS |

Table K.60. l2tp-incoming: Elements

| Element | Type | Instances | Description |
|---------|------------|---------------------|---|
| match | l2tp-relay | Optional, unlimited | Rules for relaying connections and local authentication |

K.2.46. l2tp-relay: Relay and local authentication rules for L2TP

Rules for relaying L2TP or local authentication

Table K.61. l2tp-relay: Attributes

| Attribute | Type | Default | Description |
|--------------------|--|----------------------|--|
| called-station-id | List of string | - | One or more patterns to match called-station-id |
| calling-station-id | List of string | - | One or more patterns to match calling-station-id |
| comment | string | - | Comment |
| graph | (<i>token</i>) graphname | - | Graph name |
| ip-over-lcp | boolean | - | Send IP over LCP (local auth) |
| lcp-echo-mim | boolean | - | Handle LCP echos in the middle on relayed connection |
| localpref | unsignedInt | 4294967295 | Localpref for remote-ppp-ip/routes (highest wins) |
| name | string | - | Name |
| password | Secret | - | Password check |
| payload-table | (<i>unsignedByte 0-99</i>) routetable | As per l2tp-incoming | Routing table number for payload traffic (or L2TP relay) |
| profile | NMTOKEN | - | Profile name |
| relay-hostname | string | - | Hostname for L2TP connection |
| relay-ip | List of IPAddr | - | Target IP(s) for L2TP connection |
| relay-pick | boolean | - | If set, try one of the relay IPs at random first |
| relay-secret | Secret | - | Shared secret for L2TP connection |
| remote-netmask | IP4Addr | - | Remote end PPP Netmask (local auth) |
| remote-ppp-ip | IP4Addr | - | Remote end PPP IPv4 (local auth) |
| routes | List of IPPrefix | - | Additional routes when link up (local auth) |
| source | string | - | Source of data, used in automated config management |
| username | List of string | - | One or more patterns to match username |

K.2.47. profile: Control profile

General on/off control profile used in various places in the config.

Table K.62. profile: Attributes

| Attribute | Type | Default | Description |
|----------------------|-----------------------------------|---------------------|---|
| and | List of NMTOKEN | - | Active if all specified profiles are active as well as all other tests passing, including 'not' |
| comment | string | - | Comment |
| control-switch-users | List of NMTOKEN | Any users | Restrict users that have access to control switch |
| dhcp | List of IPNameAddr | - | Test passes if any specified addresses are active in DHCP |
| expect | boolean | none | Defines state considered 'Good' and shown green on status page |
| initial | boolean | true | Defines state at system startup, or new config, where not known/fixed |
| interval | duration | 1 | Time between tests |
| invert | boolean | - | Invert final result of testing |
| log | NMTOKEN | Not logging | Log target |
| log-debug | NMTOKEN | Not logging | Log additional information |
| name | NMTOKEN | <i>Not optional</i> | Profile name |
| not | NMTOKEN | - | Active if specified profile is inactive as well as all other tests passing, including 'and' |
| or | List of NMTOKEN | - | Active if any of these other profiles are active regardless of other tests (including 'not' or 'and') |
| ports | Set of port | - | Test passes if any of these physical ports are up |
| ppp | List of NMTOKEN | - | PPP link state (any of these are up) |
| recover | duration | 1 | Time before recover (i.e. how long test has been passing) |
| route | List of IPAddr | - | Test passes if all specified addresses are routeable |
| set | switch | - | Manual override. Test settings ignored; Control switches can use and/or/not/invert |
| source | string | - | Source of data, used in automated config management |
| table | (unsignedByte 0-99) routetable | - | Routing table for ping/route/dhcp |
| timeout | duration | 10 | Time before timeout (i.e. how long test has been failing) |
| uptime | unsignedShort | - | Minimum uptime (seconds) |
| vrrp | List of NMTOKEN | - | VRRP state (any of these is master) |

Table K.63. profile: Elements

| Element | Type | Instances | Description |
|---------|--------------|---------------------|--|
| date | profile-date | Optional, unlimited | Test passes if within any date range specified |
| ping | profile-ping | Optional | Test passes if address is answering pings |
| time | profile-time | Optional, unlimited | Test passes if within any time range specified |

K.2.48. profile-date: Test passes if within any of the time ranges specified

Time range test in profiles

Table K.64. profile-date: Attributes

| Attribute | Type | Default | Description |
|-----------|----------|---------|---|
| comment | string | - | Comment |
| source | string | - | Source of data, used in automated config management |
| start | dateTime | - | Start (YYYY-MM-DDTHH:MM:SS) |
| stop | dateTime | - | End (YYYY-MM-DDTHH:MM:SS) |

K.2.49. profile-time: Test passes if within any of the date/time ranges specified

Time range test in profiles

Table K.65. profile-time: Attributes

| Attribute | Type | Default | Description |
|-----------|-------------------|---------|---|
| comment | string | - | Comment |
| days | <i>Set of day</i> | - | Which days of week apply, default all |
| source | string | - | Source of data, used in automated config management |
| start | time | - | Start (HH:MM:SS) |
| stop | time | - | End (HH:MM:SS) |

K.2.50. profile-ping: Test passes if any addresses are pingable

Ping targets

Table K.66. profile-ping: Attributes

| Attribute | Type | Default | Description |
|-----------|--------|---------|-------------|
| comment | string | - | Comment |

| | | | |
|-----------|---------------|---------------------|--|
| flow | unsignedShort | - | Flow label (IPv6) |
| gateway | IPAddr | - | Ping via specific gateway (bypasses session tracking if set) |
| ip | IPAddr | <i>Not optional</i> | Target IP |
| source | string | - | Source of data, used in automated config management |
| source-ip | IPAddr | - | Source IP |
| ttl | unsignedByte | - | Time to live / Hop limit |

K.2.51. shaper: Traffic shaper

Settings for a named traffic shaper

Table K.67. shaper: Attributes

| Attribute | Type | Default | Description |
|--------------|--|---------------------|---|
| comment | string | - | Comment |
| name | <i>(token)</i> graphname | <i>Not optional</i> | Graph name |
| rx | unsignedLong | - | Rx rate limit/target (b/s) |
| rx-limit | <i>(unsignedShort 0-1000)</i> shaper-limit | 400ms | Rx low level burst limit (ms) - ½ for large packets |
| rx-max | unsignedLong | - | Rx rate limit max |
| rx-min | unsignedLong | - | Rx rate limit min |
| rx-min-burst | duration | - | Rx minimum allowed burst time |
| rx-step | unsignedLong | - | Rx rate reduction per hour |
| share | boolean | false | If shaper is shared with other devices |
| source | string | - | Source of data, used in automated config management |
| tx | unsignedLong | - | Tx rate limit/target (b/s) |
| tx-limit | <i>(unsignedShort 0-1000)</i> shaper-limit | 400ms | Tx low level burst limit (ms) - ½ for large packets |
| tx-max | unsignedLong | - | Tx rate limit max |
| tx-min | unsignedLong | - | Tx rate limit min |
| tx-min-burst | duration | - | Tx minimum allowed burst time |
| tx-step | unsignedLong | - | Tx rate reduction per hour |

Table K.68. shaper: Elements

| Element | Type | Instances | Description |
|----------|-----------------|---------------------|--|
| override | shaper-override | Optional, unlimited | Profile specific variations on main settings |

K.2.52. shaper-override: Traffic shaper override based on profile

Settings for a named traffic shaper

Table K.69. shaper-override: Attributes

| Attribute | Type | Default | Description |
|--------------|--|---------------------|---|
| comment | string | - | Comment |
| profile | NMTOKEN | <i>Not optional</i> | Profile name |
| rx | unsignedLong | - | Rx rate limit/target (b/s) |
| rx-limit | <i>(unsignedShort 0-1000) shaper-limit</i> | 400ms | Rx low level burst limit (ms) - ½ for large packets |
| rx-max | unsignedLong | - | Rx rate limit max |
| rx-min | unsignedLong | - | Rx rate limit min |
| rx-min-burst | duration | - | Rx minimum allowed burst time |
| rx-step | unsignedLong | - | Rx rate reduction per hour |
| source | string | - | Source of data, used in automated config management |
| tx | unsignedLong | - | Tx rate limit/target (b/s) |
| tx-limit | <i>(unsignedShort 0-1000) shaper-limit</i> | 400ms | Tx low level burst limit (ms) - ½ for large packets |
| tx-max | unsignedLong | - | Tx rate limit max |
| tx-min | unsignedLong | - | Tx rate limit min |
| tx-min-burst | duration | - | Tx minimum allowed burst time |
| tx-step | unsignedLong | - | Tx rate reduction per hour |

K.2.53. ip-group: IP Group

Named IP group

Table K.70. ip-group: Attributes

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

K.2.54. dhcp-relay: DHCP server settings for remote / relayed requests

Settings for DHCP server for relayed connections

Table K.71. dhcp-relay: Attributes

| Attribute | Type | Default | Description |
|------------------|---------------------------------------|--------------------------------|---|
| allocation-table | <i>(unsignedByte 0-99) routetable</i> | Allocate same as request table | Routing table for allocations - suggest using separate tables for remote DHCP |

| | | | | | |
|-------|--|-----------|----------------|------|--|
| allow | <i>List</i> IPNameRange | <i>of</i> | Allow anywhere | from | IPs allowed (e.g. allocated IPs for renewal) |
| relay | <i>List</i> IPNameRange | <i>of</i> | Any relay | | Relay server IP(s) |
| table | <i>(unsignedByte 0-99)</i> routetable | | Allow any | | Routing table applicable |

Table K.72. dhcp-relay: Elements

| Element | Type | Instances | Description |
|---------|-------|---------------------|----------------------|
| dhcp | dhcps | Optional, unlimited | DHCP server settings |

K.3. Data types

K.3.1. user-level: User login level

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

Table K.73. 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 |

K.3.2. ppp-dump: PPP dump format

Table K.74. ppp-dump: PPP dump format

| Value | Description |
|-------------|------------------|
| default | Mixed hex/decode |
| decoded | Decoded only |
| decoded+raw | Decoded + raw |
| raw | Raw hex |

K.3.3. autoloadtype: Type of s/w auto load

Table K.75. 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 |
|-------|--------------------|

K.3.4. config-access: Type of access user has to config

Table K.76. 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) |
| demo | Full view and edit access but can only test config, not save |
| test | Full view and edit access but must test save config first |
| full | Full view and edit access |

K.3.5. eap-subsystem: Subsystem with EAP access control

Table K.77. eap-subsystem: Subsystem with EAP access control

| Value | Description |
|-------|-----------------|
| IPsec | IPsec/IKEv2 VPN |

K.3.6. eap-method: EAP access method

Table K.78. eap-method: EAP access method

| Value | Description |
|----------|---------------|
| MD5 | MD5 Challenge |
| MSChapV2 | MS Challenge |

K.3.7. syslog-severity: Syslog severity

Log severity - different loggable events log at different levels.

Table K.79. 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 |
|------------|------------|

K.3.8. syslog-facility: Syslog facility

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

Table K.80. 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 |

K.3.9. http-mode: HTTP/HTTPS security mode

Table K.81. http-mode: HTTP/HTTPS security mode

| Value | Description |
|---------------------------|---|
| http-only | No HTTPS access |
| http+https | Both HTTP and HTTPS access |
| https-only | No HTTP access |
| redirect-to-https | HTTP accesses are redirected to use HTTPS |
| redirect-to-https-if-acme | HTTP accesses are redirected to use HTTPS if ACME set up for hostname |

| | |
|----------------------------------|--|
| redirect-to-https-except-trusted | HTTP accesses are redirected to use HTTPS (except trusted IPs) |
|----------------------------------|--|

K.3.10. radiuspriority: Options for controlling platform RADIUS response priority tagging

Table K.82. radiuspriority: Options for controlling platform RADIUS response priority tagging

| Value | Description |
|----------|---|
| equal | All the same priority |
| strict | In order specified |
| random | Random order |
| calling | Hashed on calling station id |
| called | Hashed on called station id |
| username | Hashed on full username |
| user | Hashed on username before @ |
| realm | Hashed on username after @ |
| prefix | Hashed on username initial letters and numbers only |

K.3.11. radiustype: Type of RADIUS server

Table K.83. radiustype: Type of RADIUS server

| Value | Description |
|----------------|----------------------------------|
| authentication | Authentication server |
| accounting | Accounting server |
| control | Allowed to send control (CoA/DM) |

K.3.12. month: Month name (3 letter)

Table K.84. 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 |
|-----|----------|

K.3.13. day: Day name (3 letter)

Table K.85. day: Day name (3 letter)

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

K.3.14. port: Physical port

Table K.86. port: Physical port

| Value | Description |
|-------|----------------|
| 0 | Port 0 (left) |
| 1 | Port 1 (right) |

K.3.15. Crossover: Crossover configuration

Physical port crossover configuration.

Table K.87. Crossover: Crossover configuration

| Value | Description |
|-------|---------------------------------------|
| auto | Crossover is determined automatically |
| MDI | Force no crossover |

K.3.16. LinkFlow: Physical port flow control setting

Table K.88. 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 |

K.3.17. LinkClock: Physical port Gigabit clock master/slave setting

Table K.89. 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 |

K.3.18. LinkLED-y: Yellow LED setting

Table K.90. LinkLED-y: Yellow LED setting

| Value | Description |
|----------------|---|
| Link/Collision | On when link up; blink when 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 |

K.3.19. LinkLED-g: Green LED setting

Table K.91. LinkLED-g: Green LED setting

| Value | Description |
|---------------|---|
| Link/Activity | On when link up; blink when Tx or Rx activity |
| Collision | Blink when collisions detected |
| Rx | Blink when Rx activity |
| Off | Permanently off |
| On | Permanently on |

K.3.20. LinkPower: PHY power saving options

Table K.92. LinkPower: PHY power saving options

| Value | Description |
|-------|-------------------|
| none | No power saving |
| full | Full power saving |

K.3.21. LinkFault: Link fault type to send

Table K.93. 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) |

K.3.22. sampling-protocol: Sampling protocol

Table K.94. sampling-protocol: Sampling protocol

| Value | Description |
|--------------|--------------------------------|
| sflow | Use sFlow protocol |
| ipfix-psamp | Use IPFIX/PSAMP protocol |
| ipfix-legacy | Use legacy (Cisco-style) IPFIX |

K.3.23. trunk-mode: Trunk port mode

Table K.95. trunk-mode: Trunk port mode

| Value | Description |
|----------|---------------------------|
| false | Not trunking |
| random | Random trunking |
| l2-hash | L2 hashed trunking |
| l23-hash | L2 and L3 hashed trunking |
| l3-hash | L3 hashed trunking |

K.3.24. ramode: IPv6 route announce level

IPv6 route announcement mode and level

Table K.96. 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 |

K.3.25. bgpmode: BGP announcement mode

BGP mode defines the default advertisement mode for prefixes, based on well-known community tags

Table K.97. bgpmode: BGP announcement mode

| Value | Description |
|--------------|---|
| false | Not included in BGP at all |
| no-advertise | Not included in BGP, not advertised at all |
| no-export | Not normally exported from local AS/confederation |
| local-as | Not exported from local AS |
| no-peer | Exported with no-peer community tag |
| true | Exported as normal with no special tags added |

K.3.26. sampling-mode: Sampling mode

Table K.98. sampling-mode: Sampling mode

| Value | Description |
|---------|--------------------------------------|
| off | Don't perform sampling |
| ingress | Sample incoming traffic |
| egress | Sample outgoing traffic |
| both | Sample incoming and outgoing traffic |

K.3.27. sfoption: Source filter option

Table K.99. sfoption: Source filter option

| Value | Description |
|-----------|-----------------------------------|
| false | No source filter checks |
| blackhole | Check replies blackholed |
| nowhere | Check replies valid |
| self | Check replies valid and not self |
| true | Check replies down same port/vlan |

K.3.28. pppoe-mode: Type of PPPoE connection

Table K.100. pppoe-mode: Type of PPPoE connection

| Value | Description |
|-----------|---|
| client | Normal PPPoE client connects to access controller |
| bras-l2tp | PPPoE server mode linked to L2TP operation |

K.3.29. pppoe-calling: Additional prefix on PPPoE calling ID

Table K.101. pppoe-calling: Additional prefix on PPPoE calling ID

| Value | Description |
|----------|--|
| none | None |
| mac | MAC |
| vlan | Inner VLAN |
| mac-vlan | MAC and inner VLAN |
| vlanvlan | Outer and inner VLANs padded to 4 digits |

K.3.30. peertype: BGP peer type

Peer type controls many of the defaults for a peer setting. It allows typical settings to be defined with one attribute that reflects the type of peer.

Table K.102. peertype: BGP peer type

| Value | Description |
|-------|-------------|
|-------|-------------|

| | |
|-------------|---|
| normal | Normal BGP operation |
| transit | EBGP Mark received as no-export |
| peer | EBGP Mark received as no-export, only accept peer AS |
| customer | EBGP Allow export as if confederate, only accept peer AS |
| internal | IBGP allowing own AS |
| reflector | IBGP allowing own AS and working in route reflector mode |
| confederate | EBGP confederate |
| ixp | Internet exchange point peer on route server, soft routes EBGP only |

K.3.31. radius-nas: NAS IP to report

Table K.103. radius-nas: NAS IP to report

| Value | Description |
|-------|-------------------------------------|
| false | Local LNS IP (<i>deprecated</i>) |
| lns | Local LNS IP |
| both | Send NAS IP twice (LAC then LNS) |
| lac | Remote LAC IP |
| true | Remote LAC IP (<i>deprecated</i>) |

K.3.32. switch: Profile manual setting

Manual setting control for profile

Table K.104. switch: Profile manual setting

| Value | Description |
|----------------|--|
| false | Profile set to OFF |
| true | Profile set to ON |
| control-switch | Profile set based on control switch on home page |

K.4. Basic types

Table K.105. 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) |
| unsignedLong | unsigned long 64 bit integer (0-9223372036854775807) |
| unsignedInt | unsigned integer (0-4294967295) |
| unsignedShort | unsigned short integer (0-65535) |
| byte | byte integer (-128-127) |
| 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 |
| IP46Addr | IPv4 + 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 |
| IP4Subnet | IPv4 address / bitlen |
| IPFilter | Route filter |
| Password | Password |
| OTP | OTP |
| Community | xxx:xxx community |
| PortRange | xxx-xxx port range |
| Colour | #rgb #rrggbb #rgba #rrggbbaa colour |
| Secret | Secret/passphrase |
| duration | Period [[HH:]MM:]SS |
| fb-sw-update-delay | Number of days to delay upgrade by (0-30) (<i>unsignedByte</i>) |
| stringlist | List of strings (<i>string</i>) |
| macspooft | Spoof MAC base address (<i>hexBinary</i>) |
| percentage | Percentage (0 .. 100) (0-100) (<i>unsignedByte</i>) |
| routetable | Route table number (0-99) (<i>unsignedByte</i>) |
| username | Login name (<i>NMTOKEN</i>) |
| ipnamerangelist | List of IP ranges or ip groups (<i>IPNameRange</i>) |
| nmtokenlist | List of NMTOKEN (<i>NMTOKEN</i>) |
| iplist | List of IP addresses (<i>IPAddr</i>) |
| ipnamelist | List of IP addresses or domain names (<i>IPNameAddr</i>) |
| vlan | VLAN ID (0=untagged) (0-4095) (<i>unsignedShort</i>) |
| datenum | Day number in month (1-31) (<i>unsignedByte</i>) |
| sample-rate | Sampling rate (100-10000) (<i>unsignedShort</i>) |
| mtu | Max transmission unit (576-2000) (<i>unsignedShort</i>) |
| subnetlist | List of subnets (<i>IPSubnet</i>) |
| ra-max | Route announcement max interval (seconds) (4-1800) (<i>unsignedShort</i>) |

| | |
|-----------------|---|
| ra-min | Route announcement min interval (seconds) (3-1350) (<i>unsignedShort</i>) |
| ip6list | List of IPv6 addresses (<i>IP6Addr</i>) |
| macsuffix | MAC suffix (<i>hexBinary</i>) |
| ip4rangelist | List of IP4ranges (<i>IP4Range</i>) |
| macprefixlist | List of strings (<i>macprefix</i>) |
| macprefix | MAC prefix (<i>hexBinary</i>) |
| ip4list | List of IPv4 addresses (<i>IP4Addr</i>) |
| graphname | Graph name (<i>token</i>) |
| cug | CUG ID (1-32767) (<i>unsignedShort</i>) |
| prefixlist | List of IP Prefixes (<i>IPPrefix</i>) |
| vlanset | Set of VLAN IDs (<i>vlan</i>) |
| aslist | List of AS numbers (<i>unsignedIntList</i>) |
| unsignedIntList | List of integers (<i>unsignedInt</i>) |
| communitylist | List of BGP communities (<i>Community</i>) |
| filterlist | List of IP Prefix filters (<i>IPFilter</i>) |
| shaper-limit | Shaper limit (ms) (0-1000) (<i>unsignedShort</i>) |
| iprangelist | List of IPranges (<i>IPRange</i>) |
| userlist | List of user names (<i>username</i>) |
| prefix4list | List of IPv4 Prefixes (<i>IP4Prefix</i>) |
| portlist | List of protocol port ranges (<i>PortRange</i>) |
| protolist | List of IP protocols (<i>unsignedByte</i>) |
| routetableset | Set of routetables (<i>routetable</i>) |
| vlan-nz | VLAN ID (1-4095) (<i>unsignedShort</i>) |
| dates | Set of dates (<i>datenum</i>) |
| tun-id | Local tunnel ID (1-20000) (<i>unsignedShort</i>) |
| ses-id | Local session ID (1-65535) (<i>unsignedShort</i>) |
| hostname | Host name (<i>NMTOKEN</i>) |
| sip-error | SIP error code (400-699) (<i>unsignedShort</i>) |

Index

B

- BGP
 - overview, 76
- Boot process
 - overview, 29
- Breadcrumbs, 13

C

- Configuration
 - backing up and restoring, 17
 - categories (user interface), 13
 - Data types in config, 10
 - methods, 9
 - overview, 8
 - overview of using XML, 17
 - transferring using HTTP client, 20
 - using web user interface, 12
- Configuration Basic data types , 186
- Configuration Data types , 178
- Configuration Field definitions , 140

D

- DHCP
 - configuring server, 38
 - configuring subnet with DHCP client, 37
 - Relay Agent, 41
- Diagnostics
 - Access check, 68
 - Packet dumping, 69
- DNS
 - configuring resolver(s) to use, 64

E

- Ethernet Ports
 - configuring LED indication modes, 41
 - configuring speed and/or duplex modes, 41
 - relationship with interfaces, 35
 - sequenced flashing of LEDs, 29
- Event logging
 - external logging, 31
 - overview, 30
 - viewing logs, 33

G

- Graphs, 51

H

- Hostname
 - setting, 25
- HTTP service
 - configuration, 62

I

- Interfaces
 - defining, 36
 - Ethernet, 35
 - relationship with physical ports, 35
- Internet Service Providers
 - overview, 83
- IPv6 PD
 - configuring subnet via IPv6 prefix delegation, 38

L

- L2TP tunnels
 - overview, 59
- Log targets, 30
- Logging (see Event logging)

N

- Navigation buttons
 - in user interface, 16
- NTP (Network Time Protocol)
 - configuring time servers to use, 65

O

- Object Hierarchy
 - overview, 8
- Object Model
 - definition of, 8
 - formal definition, 9

P

- Packet dumping, 69
 - Example using curl and tcpdump, 71
- PPPoE
 - configuring, 55
 - overview, 54
- Profiles
 - defining, 47
 - overview, 47
 - viewing current state, 47

R

- RADIUS
 - configuring service, 66
- Route
 - definition of, 43
- Routing
 - route targets, 44

S

- Shapers, 52
- SLAAC
 - configuring subnet via SLAAC, 38, 38
- SNMP
 - configuring service, 66
- Software

identifying current version, 27

Software upgrades

- breakpoint releases, 27
- controlling auto-upgrade behaviour, 28
- overview, 26
- software release types, 26

System name (see Hostname)

System services

- checking access to, 68
- configuring, 61
- definition of, 61
- list of, 61

T

Telnet service

- configuration, 64

Time-out

- login sessions, 23

Traffic shaping

- overview, 51

U

User Interface

- customising layout, 17
- general layout, 13
- navigation, 16
- overview, 12

Users

- creating / configuring, 22
- login level, 22
- restricting logins by IP address, 24

V

Virtual Router Redundancy Protocol (VRRP), 73

- virtual router, definition of, 73
- VRRP versions, 74

VLANs

- introduction to, 99

X

XML

- introduction to, 18

XML Schema Document (XSD) file, 9