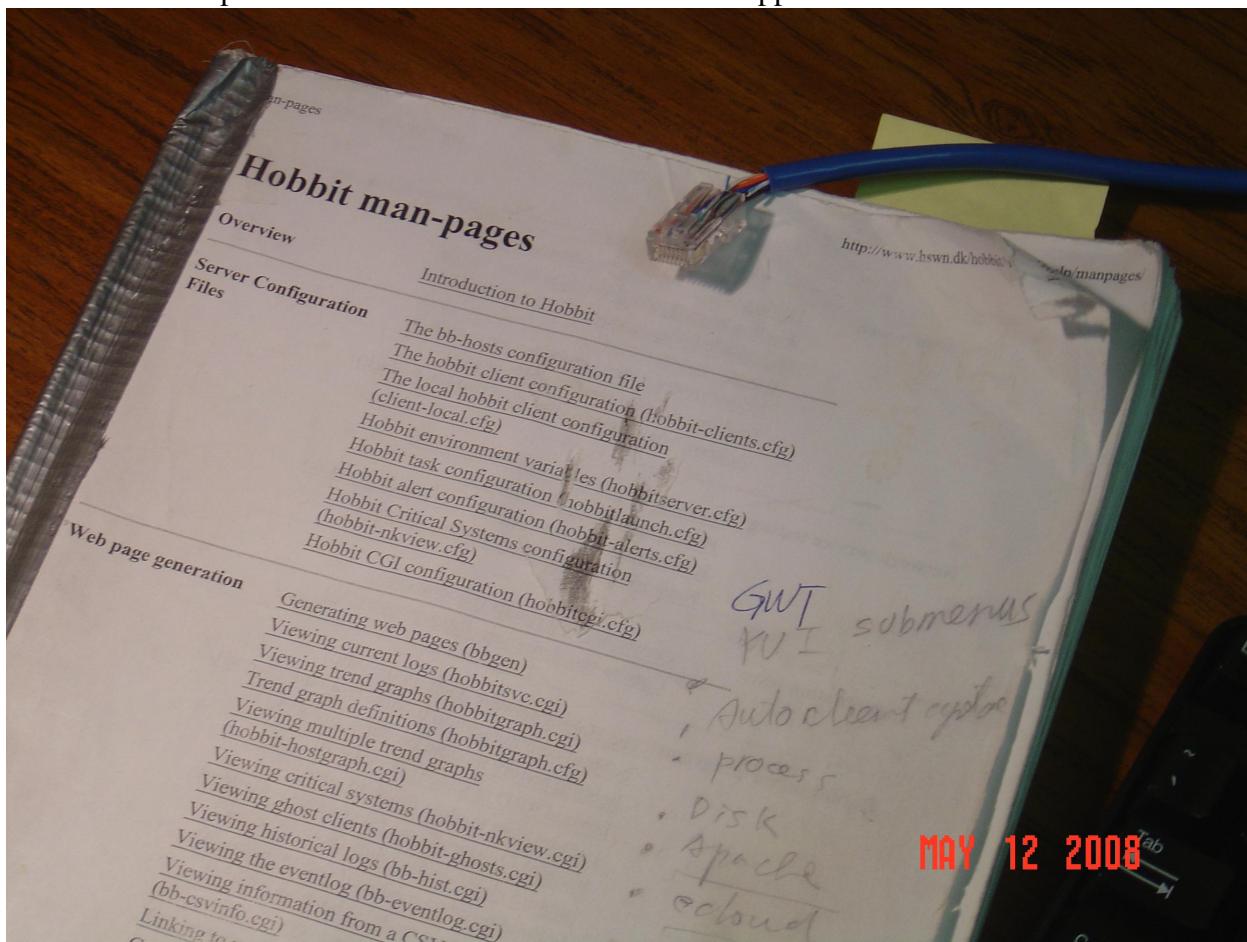


Read xymon Manpages

xymon Community

Copyright © 2008 GNU

This material may be distributed only subject to the terms and conditions set forth in version 1.0 of the Open Publication License. Please refer to Appendix B for the license text.



Contents

Contents	i
1 Preface	2
1.1 This book is a work in progress	2
1.2 xymon Documentation Road Map	2
1.3 Revision History	2
1.4 Colophon—this book is Free	3
2 xymon Overview	4
2.1 xymon - Introduction to the Xymon Monitor	4
2.2 FEATURES	4
2.3 SECURITY	6
2.4 DEMONSTRATION SITE	6
2.5 PREREQUISITES	6
2.6 INSTALLATION	6
2.7 SUPPORT and MAILING LISTS	7
2.8 xymon Wiki Book	7
2.9 HOBBIT SERVER DAEMONS	7
2.10 HOBBIT NETWORK TEST TOOLS	8
2.11 HOBBIT TOOLS HANDLING THE WEB USER-INTERFACE	8
2.12 CLIENT-SIDE TOOLS	9
2.13 HOBBIT COMMUNICATION TOOLS	9
2.14 OTHER TOOLS	9
2.15 VERSIONS	10
2.16 COPYRIGHT	10
2.17 SEE ALSO	10
3 How to setup custom graphs	13
3.1 How to setup custom graphs	13
3.1.1 Make a script to collect the data	13
3.1.2 Get xymonlaunch to run the script	13
3.1.3 Check that the script data arrives in xymon	14
3.1.4 Arrange for the data to be collected into an RRD file	14
3.1.5 Check that the RRD collects data	15
3.1.6 Setup a graph definition	15
3.1.7 Add the graph to the collection of graphs on the trends column	16
3.1.8 Common problems and pitfalls	16

4	Installing xymon	17
4.1	Installing xymon	17
4.1.1	Prerequisites - before you install xymon	17
4.1.2	Building xymon	18
4.1.3	Configuring your webserver	19
4.1.4	Starting xymon	19
4.1.5	Next steps	20
5	About the xymon Monitor	22
5.1	About xymon	22
6	Configuring xymon Alerts	26
6.1	Configuring xymon Alerts	26
7	Configuring xymon Monitoring	30
7.1	Configuring xymon Monitoring	30
7.2	Setting up monitoring of hosts	31
7.3	Upgrading from Big Brother to xymon	34
8	Integrating MRTG data into xymon	36
8.1	Integrating MRTG data into xymon	36
9	Critical Systems	38
9.1	Critical Systems	38
9.2	The Critical Systems editor	38
9.3	The Critical Systems view	41
10	Known issues in xymon	48
10.1	Known issues in xymon	48
11	Server Configuration files	51
11.1	xymonserver.cfg - xymon Environment Variables	51
11.1.1	DESCRIPTION	51
11.1.2	ENVIRONMENT AREAS	51
11.1.3	GENERAL SETTINGS	51
11.1.4	DIRECTORIES	52
11.1.5	SYSTEM FILES	53
11.1.6	URLS	53
11.1.7	SETTINGS FOR SENDING MESSAGES TO HOBBIT	54
11.1.8	HOBBITD SETTINGS	54
11.1.9	HOBBITD_HISTORY SETTINGS	55
11.1.10	HOBBITD_ALERT SETTINGS	55
11.1.11	HOBBITD_RRD SETTINGS	56
11.1.12	BBTEST-NET NETWORK TEST SETTINGS	56
11.1.13	BBGEN WEBPAGE GENERATOR SETTINGS	57
11.1.14	FILES	58
11.1.15	SEE ALSO	58
11.2	bb-hosts - Main xymon configuration file	58
11.2.1	SYNOPSIS	58
11.2.2	DESCRIPTION	58
11.2.3	FILE FORMAT	58
11.2.4	TAGS RECOGNIZED BY ALL TOOLS	59
11.2.5	GENERAL PER-HOST OPTIONS	59

11.2.6 BBGEN DISPLAY OPTIONS	59
11.2.7 HOBBIT TAGS FOR THE CRITICAL SYSTEMS OVERVIEW PAGE	62
11.2.8 HOBBIT TAGS FOR THE WML (WAP) CARDS	62
11.2.9 HOBBIT STATUS PROPAGATION OPTIONS	63
11.2.10 HOBBIT AVAILABILITY REPORT OPTIONS	63
11.2.11 NETWORK TEST SETTINGS	64
11.2.12 CONNECTIVITY (PING) TEST	65
11.2.13 SIMPLE NETWORK TESTS	66
11.2.14 DNS SERVER TESTS	67
11.2.15 OTHER NETWORK TESTS	67
11.2.16 HTTP TESTS	67
11.2.17 LDAP (DIRECTORY SERVER) TESTS	71
11.2.18 PERFORMANCE MONITORING TESTS	71
11.2.19 DEFAULT HOST	72
11.2.20 SENDING SUMMARIES TO REMOTE HOBBIT SERVERS	72
11.2.21 OTHER TAGS	73
11.2.22 FILES	73
11.2.23 SEE ALSO	73
11.3 xymon-alerts.cfg - Configuration for the hobbitd_alert module	74
11.3.1 SYNOPSIS	74
11.3.2 DESCRIPTION	74
11.3.3 FILE FORMAT	74
11.3.4 RULES	74
11.3.5 RECIPIENTS	75
11.3.6 MACROS	75
11.3.7 ALERT SCRIPTS	75
11.3.8 SEE ALSO	76
11.4 xymon-clients.cfg - Configuration file for the hobbitd_client module	77
11.4.1 SYNOPSIS	77
11.4.2 DESCRIPTION	77
11.4.3 FILE FORMAT	77
11.4.4 CPU STATUS COLUMN SETTINGS	77
11.4.5 DISK STATUS COLUMN SETTINGS	77
11.4.6 MEMORY STATUS COLUMN SETTINGS	78
11.4.7 PROCS STATUS COLUMN SETTINGS	78
11.4.8 MSGS STATUS COLUMN SETTINGS	79
11.4.9 FILES STATUS COLUMN SETTINGS	79
11.4.10 PORTS STATUS COLUMN SETTINGS	81
11.4.11 CHANGING THE DEFAULT SETTINGS	81
11.4.12 RULES TO SELECT HOSTS	82
11.4.13 DIRECTING ALERTS TO GROUPS	82
11.4.14 RULES: APPLYING SETTINGS TO SELECTED HOSTS	82
11.4.15 NOTES	83
11.4.16 SEE ALSO	83
11.5 HOBBITCGI.CFG	84
11.5.1 SYNOPSIS	84
11.5.2 DESCRIPTION	84
11.5.3 SETTINGS	84
11.5.4 SEE ALSO	85
11.6 HOBBIT-NKVIEW.CGI	86
11.6.1 SYNOPSIS	86
11.6.2 DESCRIPTION	86

11.6.3 RELATION TO BBNK PAGE	86
11.6.4 OPTIONS	86
11.6.5 ENVIRONMENT VARIABLES	86
11.6.6 SEE ALSO	86
11.7 CLIENT-LOCAL.CFG	87
11.7.1 NAME	87
11.7.2 SYNOPSIS	87
11.7.3 DESCRIPTION	87
11.7.4 PROPAGATION TO CLIENTS	87
11.7.5 FILE FORMAT	87
11.7.6 LOGFILE CONFIGURATION ENTRIES	87
11.7.7 COUNTING LOGENTRIES	88
11.7.8 FILE CONFIGURATION ENTRIES	88
11.7.9 DIRECTORY CONFIGURATION ENTRIES	88
11.7.10 NOTES	88
11.7.11 SEE ALSO	89
12 Web page generation	90
12.1 BBGEN	90
12.1.1 SYNOPSIS	90
12.1.2 DESCRIPTION	90
12.1.3 OPTIONS	90
12.1.4 GENERAL OPTIONS	90
12.1.5 PAGE LAYOUT OPTIONS	91
12.1.6 COLUMN SELECTION OPTIONS	92
12.1.7 STATUS PROPAGATION OPTIONS	92
12.1.8 PURPLE STATUS OPTIONS	93
12.1.9 ALTERNATE PAGESET OPTIONS	93
12.1.10 ALTERNATE OUTPUT FORMATS	93
12.1.11 OPTIONS USED BY CGI FRONT-ENDS	94
12.1.12 DEBUGGING OPTIONS	94
12.1.13 BUILDING ALTERNATE PAGESETS	94
12.1.14 USING BBGEN FOR REPORTS	95
12.1.15 SLA REPORTING	96
12.1.16 PRE-GENERATED REPORTS	96
12.1.17 SEE ALSO	97
12.2 BB-HIST.CGI	98
12.2.1 SYNOPSIS	98
12.2.2 DESCRIPTION	98
12.2.3 PARAMETERS	98
12.2.4 OPTIONS	99
12.2.5 SEE ALSO	99
12.3 HOBBIT-NKVIEW.CGI	100
12.3.1 SYNOPSIS	100
12.3.2 DESCRIPTION	100
12.3.3 RELATION TO BBNK PAGE	100
12.3.4 OPTIONS	100
12.3.5 ENVIRONMENT VARIABLES	100
12.3.6 SEE ALSO	100
12.4 BBHOSTGREP	101
12.4.1 SYNOPSIS	101
12.4.2 DESCRIPTION	101

12.4.3 OPTIONS	101
12.4.4 EXAMPLE	101
12.4.5 USE IN EXTENSION SCRIPTS	102
12.4.6 ENVIRONMENT VARIABLES	102
12.4.7 FILES	102
12.4.8 SEE ALSO	102
12.5 HOBBITD	103
12.5.1 SYNOPSIS	103
12.5.2 DESCRIPTION	103
12.5.3 OPTIONS	103
12.5.4 HOW ALERTS TRIGGER	105
12.5.5 CHANNELS	105
12.5.6 SIGNALS	105
12.5.7 BUGS	106
12.5.8 FILES	106
12.5.9 SEE ALSO	106
12.6 HOBBITPING	107
12.6.1 SYNOPSIS	107
12.6.2 DESCRIPTION	107
12.6.3 SUID-ROOT INSTALLATION REQUIRED	107
12.6.4 OPTIONS	107
12.6.5 SEE ALSO	108
12.7 HOBBITD_ALERT	109
12.7.1 SYNOPSIS	109
12.7.2 DESCRIPTION	109
12.7.3 OPTIONS	109
12.7.4 HOW HOBBIT DECIDES WHEN TO SEND ALERTS	109
12.7.5 ENVIRONMENT	110
12.7.6 FILES	110
12.7.7 SEE ALSO	110
12.8 BBHOSTSHOW	111
12.8.1 SYNOPSIS	111
12.8.2 DESCRIPTION	111
12.8.3 OPTIONS	111
12.8.4 ENVIRONMENT VARIABLES	111
12.8.5 SEE ALSO	111
12.9 HOBBITD_CAPTURE	112
12.9.1 SYNOPSIS	112
12.9.2 DESCRIPTION	112
12.9.3 OPTIONS	112
12.9.4 SEE ALSO	112
12.10 HOBBIT-STATUSREPORT.CGI	113
12.10.1 SYNOPSIS	113
12.10.2 DESCRIPTION	113
12.10.3 EXAMPLES	113
12.10.4 OPTIONS	113
12.10.5 SEE ALSO	114
12.11 BBMESSAGE.CGI	115
12.11.1 SYNOPSIS	115
12.11.2 DESCRIPTION	115
12.11.3 SECURITY	115
12.11.4 SEE ALSO	115

12.12 BB-WEBPAGE	116
12.12.1 SYNOPSIS	116
12.12.2 DESCRIPTION	116
12.12.3 OPTIONS	116
12.12.4 SEE ALSO	116
13 Report Generation	117
13.1 HOBBIT-CONFREPORT.CGI	118
13.1.1 SYNOPSIS	118
13.1.2 DESCRIPTION	118
13.1.3 OPTIONS	118
13.1.4 BUGS	118
13.1.5 SEE ALSO	118
13.2 BB-REP.CGI	119
13.2.1 SYNOPSIS	119
13.2.2 DESCRIPTION	119
13.2.3 OPTIONS	119
13.2.4 FILES	119
13.2.5 ENVIRONMENT VARIABLES	119
13.2.6 SEE ALSO	120
13.3 HOBBITD_FILESTORE	121
13.3.1 SYNOPSIS	121
13.3.2 DESCRIPTION	121
13.3.3 OPTIONS	121
13.3.4 FILES	122
13.3.5 SEE ALSO	122
14 xymon Server Administration	123
14.1 HOBBIT-ENADIS.CGI	124
14.1.1 SYNOPSIS	124
14.1.2 DESCRIPTION	124
14.1.3 OPTIONS	124
14.1.4 BUGS	124
14.1.5 SEE ALSO	124
14.2 BB-ACK.CGI	125
14.2.1 SYNOPSIS	125
14.2.2 DESCRIPTION	125
14.2.3 PARAMETERS	125
14.2.4 OPTIONS	125
14.2.5 FILES	125
14.2.6 ENVIRONMENT VARIABLES	125
14.2.7 BUGS	126
14.2.8 SEE ALSO	126
14.3 HOBBIT-GHOSTS.CGI	127
14.3.1 SYNOPSIS	127
14.3.2 DESCRIPTION	127
14.3.3 OPTIONS	127
14.3.4 SEE ALSO	127
14.4 TRIMHISTORY	128
14.4.1 NAME	128
14.4.2 DESCRIPTION	128
14.4.3 OPTIONS	128

14.4.4 FILES	128
14.4.5 ENVIRONMENT VARIABLES	129
14.4.6 SEE ALSO	129
14.5 HOBBIT-ACKINFO.CGI	130
14.5.1 SYNOPSIS	130
14.5.2 DESCRIPTION	130
14.5.3 OPTIONS	130
14.5.4 ENVIRONMENT VARIABLES	130
14.5.5 SEE ALSO	130
15 Network Service Testing	131
15.1 BBTEST-NET	131
15.1.1 SYNOPSIS	131
15.1.2 DESCRIPTION	131
15.1.3 GENERAL OPTIONS	131
15.1.4 OPTIONS FOR TESTS OF THE SIMPLE TCP SERVICES	133
15.1.5 OPTIONS FOR THE PING TEST	133
15.1.6 OPTIONS FOR HTTP (WEB) TESTS	133
15.1.7 OPTIONS FOR SSL CERTIFICATE TESTS	133
15.1.8 DEBUGGING OPTIONS	134
15.1.9 INFORMATIONAL OPTIONS	134
15.1.10 USING COOKIES IN WEB TESTS	134
15.1.11 ABOUT SSL CERTIFICATE CHECKS	134
15.1.12 LDAP TESTS	134
15.1.13 USING MULTIPLE NETWORK TEST SYSTEMS	136
15.1.14 BBTEST-NET INTERNALS	136
15.1.15 ENVIRONMENT VARIABLES	136
15.1.16 FILES	137
15.1.17 SEE ALSO	137
15.2 BB-SERVICES	138
15.2.1 SYNOPSIS	138
15.2.2 DESCRIPTION	138
15.2.3 FILE FORMAT	138
15.2.4 FILES	138
15.2.5 SEE ALSO	139
15.3 HOBBITFETCH	140
15.3.1 NAME	140
15.3.2 SYNOPSIS	140
15.3.3 DESCRIPTION	140
15.3.4 OPTIONS	140
15.3.5 SEE ALSO	140
15.4 bbcmd	141
15.4.1 SYNOPSIS	141
15.4.2 DESCRIPTION	141
15.4.3 SEE ALSO	141
15.5 CLIENTLAUNCH.CFG	142
15.5.1 SYNOPSIS	142
15.5.2 DESCRIPTION	142
15.5.3 FILE FORMAT	142
15.5.4 SEE ALSO	142

16 Combination Tests	143
16.1 BBCOMBOTEST	143
16.1.1 SYNOPSIS	143
16.1.2 DESCRIPTION	143
16.1.3 OPTIONS	143
16.1.4 FILES	143
16.1.5 SEE ALSO	143
16.2 HOBBITGRAPH.CGI	144
16.2.1 NAME	144
16.2.2 SYNOPSIS	144
16.2.3 DESCRIPTION	144
16.2.4 OPTIONS	144
16.2.5 ENVIRONMENT	144
16.2.6 FILES	144
16.2.7 SEE ALSO	145
16.3 BBCOMBOTEST.CFG	146
16.3.1 NAME	146
16.3.2 SYNOPSIS	146
16.3.3 DESCRIPTION	146
16.3.4 FILE FORMAT	146
16.3.5 EXAMPLE	146
16.3.6 FILES	147
16.3.7 SEE ALSO	147
16.4 HOBBITGRAPH.CGI	148
16.4.1 NAME	148
16.4.2 SYNOPSIS	148
16.4.3 DESCRIPTION	148
16.4.4 OPTIONS	148
16.4.5 ENVIRONMENT	148
16.4.6 FILES	148
16.4.7 SEE ALSO	149
16.5 BB-SNAPSHOT.CGI	150
16.5.1 NAME	150
16.5.2 SYNOPSIS	150
16.5.3 DESCRIPTION	150
16.5.4 OPTIONS	150
16.5.5 ENVIRONMENT VARIABLES	150
16.5.6 SEE ALSO	150
16.6 HOBBITD_SAMPLE	151
16.6.1 NAME	151
16.6.2 SYNOPSIS	151
16.6.3 DESCRIPTION	151
16.6.4 OPTIONS	151
16.6.5 SEE ALSO	151
16.7 BB-CSVINFO.CGI	152
16.7.1 NAME	152
16.7.2 SYNOPSIS	152
16.7.3 DESCRIPTION	152
16.7.4 Example usage	152
16.7.5 SEE ALSO	152
16.8 HOBBIT-HOSTGRAPHS.CGI	153
16.8.1 SYNOPSIS	153

16.8.2 DESCRIPTION	153
16.8.3 OPTIONS	153
16.8.4 BUGS	153
16.8.5 SEE ALSO	153
16.9 BB-DATEPAGE.CGI	154
16.9.1 SYNOPSIS	154
16.9.2 DESCRIPTION	154
16.9.3 OPTIONS	154
16.9.4 ENVIRONMENT VARIABLES	154
16.9.5 SEE ALSO	155
16.10 BBDIGEST	156
16.10.1 SYNOPSIS	156
16.10.2 DESCRIPTION	156
16.10.3 EXAMPLE	156
16.10.4 SEE ALSO	156
16.11 HOBBITLAUNCH.CFG	157
16.11.1 SYNOPSIS	157
16.11.2 DESCRIPTION	157
16.11.3 FILE FORMAT	157
16.11.4 SEE ALSO	158
16.12 BB-EVENTLOG.CGI	159
16.12.1 SYNOPSIS	159
16.12.2 DESCRIPTION	159
16.12.3 OPTIONS	159
16.12.4 SEE ALSO	159
16.13 HOBBIT-MAILACK	160
16.13.1 SYNOPSIS	160
16.13.2 DESCRIPTION	160
16.13.3 ADDING INFORMATION TO THE REPLY MAIL	160
16.13.4 USE WITH PROCMAIL	160
16.13.5 USE WITH QMAIL	160
16.13.6 OPTIONS	160
16.13.7 SEE ALSO	160
16.14 BB-FINDHOST.CGI	161
16.14.1 SYNOPSIS	161
16.14.2 DESCRIPTION	161
16.14.3 OPTIONS	161
16.14.4 FILES	161
16.14.5 ENVIRONMENT VARIABLES	161
16.14.6 SEE ALSO	161
16.15 HOBBITCLIENT.CFG	162
16.15.1 DESCRIPTION	162
16.15.2 SETTINGS	162
16.15.3 INHERITED SETTINGS	162
16.15.4 SEE ALSO	162
17 xymon/BB Messages Protocols	163
17.1 BNF	163
17.1.1 xymon messages syntax	163
17.1.2 Big Brother messages syntax	164
17.2 bb - xymon client communication program	165
17.2.1 SYNOPSIS	165

17.2.2 DESCRIPTION	165
17.2.3 OPTIONS AND PARAMETERS	165
17.2.4 HOBBIT MESSAGE SYNTAX	165
17.2.5 EXAMPLE	168
17.2.6 NOTES	169
17.2.7 SEE ALSO	169
18 xymon server programs	170
18.1 HOBBITD_CHANNEL	170
18.1.1 SYNOPSIS	170
18.1.2 DESCRIPTION	170
18.1.3 OPTIONS	170
18.1.4 FILES	171
18.1.5 SEE ALSO	171
18.2 BB-HOSTSVC.CGI	172
18.2.1 SYNOPSIS	172
18.2.2 DESCRIPTION	172
18.2.3 OPTIONS	172
18.2.4 FILES	173
18.2.5 ENVIRONMENT	173
18.2.6 SEE ALSO	173
18.3 BBPROXY	174
18.3.1 SYNOPSIS	174
18.3.2 DESCRIPTION	174
18.3.3 OPTIONS	174
18.3.4 REPORT OUTPUT	175
18.3.5	176
18.3.6 SIGNALS	176
18.3.7 SEE ALSO	176
18.4 HOBBITD_CLIENT	177
18.4.1 SYNOPSIS	177
18.4.2 DESCRIPTION	177
18.4.3 OPTIONS	177
18.4.4 FILES	177
18.4.5 SEE ALSO	177
18.5 HOBBITWEB	178
18.5.1 DESCRIPTION	178
18.5.2 SEE ALSO	179
19 xymon Client	180
19.1 “/opt/bin/logfetch” - xymon client data collector	180
19.1.1 SYNOPSIS	180
19.1.2 DESCRIPTION	180
19.1.3 SECURITY	180
19.1.4 ENVIRONMENT VARIABLES	180
19.1.5 FILES	180
19.1.6 SEE ALSO	181
19.2 CLIENTUPDATE	182
19.2.1 NAME	182
19.2.2 SYNOPSIS	182
19.2.3 DESCRIPTION	182
19.2.4 OPTIONS	182

19.2.5 USING CLIENTUPDATE IN HOBBIT	182
19.2.6 OPERATION	183
19.2.7 ENVIRONMENT VARIABLES	183
19.2.8 SEE ALSO	183
19.3 BB-REPOLOG.CGI	184
19.3.1 SYNOPSIS	184
19.3.2 DESCRIPTION	184
19.3.3 OPTIONS	184
19.3.4 SEE ALSO	184
19.4 HOBBITD_HISTORY	185
19.4.1 SYNOPSIS	185
19.4.2 DESCRIPTION	185
19.4.3 OPTIONS	185
19.4.4 ENVIRONMENT	185
19.4.5 FILES	185
19.4.6 SEE ALSO	185
19.5 MSGCACHE	186
19.5.1 SYNOPSIS	186
19.5.2 DESCRIPTION	186
19.5.3 RESTRICTIONS	186
19.5.4 OPTIONS	186
19.5.5 SEE ALSO	186
19.6 BBRETEST-NET.SH	187
19.6.1 SYNOPSIS	187
19.6.2 DESCRIPTION	187
19.6.3 INSTALLATION	187
19.6.4 FILES	187
19.6.5 SEE ALSO	187
19.7 HOBBITD_HOSTDATA	188
19.7.1 SYNOPSIS	188
19.7.2 DESCRIPTION	188
19.7.3 FILES	188
19.7.4 SEE ALSO	188
19.8 ORCAHOBBIT	189
19.8.1 SYNOPSIS	189
19.8.2 NOTICE	189
19.8.3 DESCRIPTION	189
19.8.4 OPTIONS	189
19.8.5 SEE ALSO	189
20 Data Backend	190
20.1 HOBBITD_RRD	190
20.1.1 SYNOPSIS	190
20.1.2 DESCRIPTION	190
20.1.3 OPTIONS	190
20.1.4 ENVIRONMENT	191
20.1.5 COLLECTED DATA	191
20.1.6 CUSTOM RRD DATA IN NAME-COLON-VALUE (NCV) FORMAT	192
20.1.7 CUSTOM RRD DATA VIA SCRIPTS	193
20.1.8 COMPATIBILITY	194
20.1.9 SEE ALSO	195

21	Miscellaneous programs	196
21.1	HOBBITLAUNCH	197
21.1.1	SYNOPSIS	197
21.1.2	DESCRIPTION	197
21.1.3	OPTIONS	197
21.1.4	STARTING TASKS	197
21.1.5	SEE ALSO	198
A	Making of this book	199
A.1	Using Mercurial's book template to build our book or	199
A.2	On Windows	200
B	Open Publication License	201
B.1	Requirements on both unmodified and modified versions	201
B.2	Copyright	201
B.3	Scope of license	201
B.4	Requirements on modified works	202
B.5	Good-practice recommendations	202
B.6	License options	202
Bibliography		203
Index		203

List of Figures

4.1	xymonprocs.png	20
4.2	xymonmain.png	21
9.1	critview-disk.png	39
9.2	editor-main.png	40
9.3	test	40
9.4	editor-nohost.png	41
9.5	editor-diskchanged.png	42
9.6	critview-disk.png	43
9.7	editor-clonemaster.png	44
9.8	editor-makeclone.png	44
9.9	editor-showclone.png	45
9.10	critview-disk.png	45
9.11	critview-detail-ackform.png	46
9.12	critview-detail-acked.png	46
9.13	mainview-acked.png	47
9.14	stdview-detail-acked.png	47

List of Tables

7.1	Monitored Network Services	32
7.2	Monitored Network Services with SSL enabled	32
7.3	Other Monitored Network Services	32

Chapter 1

Preface

System Monitoring is an important subject in a IT department. With good system monitoring tool like xymon, it provides your IT staff a tool to provide proactive system monitoring service.

Service outages can be detected quickly and even prevented by monitoring xymon closely.

I am turning xymon man pages into a book using latex because following reasons

- RTFM: Read The Fine Manpage is the most direct and efficient way of learning a Unix subject. but it is hard if you trying to read a set manualpages.
- when man pages about xymon is 58 form author Henry.
- Problems of this approach is that manpage is not a book. it has no Table of Content,Indexes to locate a term easily.

1.1 This book is a work in progress

I am writing this book about xymon from perspective of manpages.

I am releasing this xymon RTFM book while I am still writing it, in the hope that it will prove useful to others. I also hope that readers will contribute as they see fit.

1.2 xymon Documentation Road Map

- xymon RTFM.
- xymon User Guide.
- xymon Developer Guide.
- xymon Administration Guide.
- xymon Slides.

1.3 Revision History

- Henrik Storner
 - 1. Wrote the original manpages in troff
- T.J. Yang:

1. Import from troff sourcefile to tex
2. fix tex file into a chapter base tex syntax.

•

1.4 Colophon—this book is Free

This book is licensed under the Open Publication License, and is produced entirely using Free Software tools. It is typeset with LATEX; illustrations are drawn and rendered with Inkscape.

The complete source code for this book is published as a Mercurial repository, at <http://hg.serpentine.com/mercurial/book>.

Chapter 2

xymon Overview

2.1 xymon - Introduction to the Xymon Monitor

[Wau] xymon is a tool for monitoring the health of your networked servers and the applications running on them. It provides a simple, intuitive way of checking the health of your systems from a webbrowser, and can also alert you to any problems that arise through alarms sent as e-mail, SMS messages, via a pager or by other means.

xymon is Open Source software, licensed under the GNU GPL. This means that you are free to use xymon as much as you like, and you are free to re-distribute it and change it to suit your specific needs. However, if you change it then you must make your changes available to others on the same terms that you received xymon originally. See the file COPYING in the xymon source-archive for details.

xymon initially began life as an enhancement to Big Brother called “bbgen”. Over a period of 5 years, xymon has evolved from a small add-on to a full-fledged monitoring system with capabilities far exceeding what was in the original Big Brother package. xymon does still maintain some compatibility with Big Brother, so it is possible to migrate from Big Brother to xymon without too much trouble.

Migrating to xymon will give you a significant performance boost, and provide you with much more advanced monitoring. The xymon tools are designed for installations that need to monitor a large number of hosts, with very little overhead on the monitoring server. Monitoring of thousands of hosts with a single xymon server is possible - it was developed to handle just this task.

2.2 FEATURES

These are some of the core features in xymon:

Monitoring of hosts and networks xymon collects information about your systems in two ways: From querying network services (Web, LDAP, DNS, Mail etc.), or from scripts that run either on the xymon server or on the systems you monitor. The xymon package includes a xymon client which you can install on the servers you monitor; it collects data about the CPU-load, disk- and memory-utilisation, logfiles, network ports in use, file- and directory-information and more. All of the information is stored inside xymon, and you can define conditions that result in alerts, e.g. if a network service stops responding, or a disk fills up.

Centralized configuration All configuration of xymon is done on the xymon server. Even when monitoring hundreds or thousands of hosts, you can control their configuration centrally on the xymon server - so there is no need for you to login to a system just to change e.g. which processes are monitored.

Works on all major platforms The xymon server works on all Unix-like systems, including Linux, Solaris, FreeBSD, AIX, HP-UX and others. The xymon client supports all major Unix platforms, and there are other Open Source projects - e.g. BBWin, see <http://bbwin.sourceforge.net/> - providing support for Microsoft Windows based systems.

A simple, intuitive web-based front-end “Green is good, red is bad”. Using the xymon webpages is as simple as that. The hosts you monitor can be grouped together in a way that makes sense in your organisation and presented in a tree-structure. The webpages use many techniques to convey information about the monitored systems, e.g. different icons can be used for recently changed statuses; links to subpages can be listed in multiple columns; different icons can be used for dialup-tests or reverse-tests; selected columns can be dropped or unconditionally included on the webpages to eliminate unwanted information, or always include certain information; user-friendly names can be shown for hosts regardless of their true hostname. You can also have automatic links to on-line documentation, so information about your critical systems is just a click away.

Integrated trend analysis, historical data and SLA reporting xymon stores trend- and availability-information about everything it monitors. So if you need to look at how your systems behave over time, xymon has all of the information you need: Whether is response times of your webpages during peak hours, the CPU utilisation over the past 4 weeks, or what the availability of a site was compared to the SLA - it's all there inside xymon. All measurements are tracked and made available in time-based graphs.

When you need to drill down into events that have occurred, xymon provides a powerful tool for viewing the event history for each statuslog, with overviews of when problems have occurred during the past and easy-to-use zoom-in on the event.

For SLA reporting, You can configure planned downtime, agreed service availability level, service availability time and have xymon generate availability reports directly showing the actual availability measured against the agreed SLA. Such reports of service availability can be generated on-the-fly, or pre-generated e.g. for monthly reporting.

Role-based views You can have multiple different views of the same hosts for different parts of the organisation, e.g. one view for the hardware group, and another view for the webmasters - all of them fed by the same test tools.

If you have a dedicated Network Operations Centre, you can configure precisely which alerts will appear on their monitors - e.g. a simple anomaly in the system logfile need not trigger a call to 3rd-level support at 2 AM, but if the on-line shop goes down you do want someone to respond immediately. So you put the webcheck for the on-line shop on the NOC monitor, and leave out the log-file check.

Also for the techies The xymon user-interface is simple, but engineers will also find lots of relevant information. E.g. the data that clients report to xymon contain the raw output from a number of system commands. That information is available directly in xymon, so an administrator no longer needs to login to a server to get an overview of how it is behaving - the very commands they would normally run have already been performed, and the results are on-line in xymon.

Easy to adapt to your needs xymon includes a lot of tests in the core package, but there will always be something specific to your setup that you would like to watch. xymon allows you to write test scripts in your favourite scripting language and have the results show up as regular status columns in xymon. You can trigger alerts from these, and even track trends in graphs just by a simple configuration setting.

Real network service tests The network test tool knows how to test most commonly used protocols, including HTTP, SMTP (e-mail), DNS, LDAP (directory services), and many more. When checking websites, it is possible to not only check that the webserver is responding, but also that the response looks correct by matching the response against a pre-defined pattern or a checksum. So you can test that a network service is really working and supplying the data you expect - not just that the service is running.

Protocols that use SSL encryption such as https-websites are fully supported, and while checking such services the network tester will automatically run a check of the validity of the SSL server certificate, and warn about certificates that are about to expire.

Highly configurable alerts You want to know when something breaks. But you don't want to get flooded with alerts all the time. xymon lets you define several criteria for when to send out an alert, so you only get alerts when there is really something that needs your attention right away. While you are handling an incident, you can tell xymon about it so it stops sending more alerts, and so that everyone else can check with xymon and know that the problem is being taken care of.

Combined super-tests and test interdependencies If a single test is not enough, combination tests can be defined that combine the result of several tests to a single status-report. So if you need to monitor that at least 3 out of 5 servers are running at any time, xymon can do that for you and generate the necessary availability report.

Tests can also be configured to depend on each other, so that when a critical router goes down you will get alerts only for the router - and not from the 200 hosts behind the router.

2.3 SECURITY

All of the xymon server tools run under an unprivileged user account. A single program - the *xymonping(1)* network connectivity tester - must be installed setuid-root, but has been written so that it drops all root privileges immediately after performing the operation that requires root privileges.

It is recommended that you setup a dedicated account for xymon.

Communications between the xymon server and Xymon clients use the Big Brother TCP port 1984. If the xymon server is located behind a firewall, it must allow for inbound connections to the xymon server on tcp port 1984. Normally, xymon clients - i.e. the servers you are monitoring - must be permitted to connect to the xymon server on this port. However, if that is not possible due to firewall policies, then xymon includes the *xymonfetch(8)* and *msgcache(8)* tools to allows for a pull-style way of collecting data, where it is the xymon server that initiates connections to the clients.

The xymon webpages are dynamically generated through CGI programs.

Access to the xymon webpages is controlled through your webserver access controls, e.g. you can require a login through some form of HTTP authentication.

2.4 DEMONSTRATION SITE

A site running this software can be seen at <http://www.hswn.dk/xymon/>

2.5 PREREQUISITES

You will need a Unix-like system (Linux, Solaris, HP-UX, AIX, FreeBSD, Mac OS X or similar) with a webserver installed. You will also need a C compiler and some additional libraries, but many systems come with the required development tools and libraries pre-installed. The required libraries are:

1. **RRDtool** This library is used to store and present trend-data. It is required.
2. **libpcre** This library is used for advanced pattern-matching of text strings in configuration files. This library is required.
3. **OpenSSL** This library is used for communication with SSL-enabled network services. Although optional, it is recommended that you install this for xymon since many network tests do use SSL.
4. **OpenLDAP** This library is used for testing LDAP servers. Use of this is optional.

2.6 INSTALLATION

For more detailed information about xymon system requirements and how to install xymon, refer to the online documentation “Installing xymon” available from the Xymon webserver (via the “Help” menu), or from the “docs/install.html” file in the xymon source archive.

2.7 SUPPORT and MAILING LISTS

xymon@hswn.dk is an open mailing list for discussions about xymon. If you would like to participate, send an e-mail to **xymon-subscribe@hswn.dk** to join the list.

An archive of the mailing list is available at <http://www.hswn.dk/xymonon/>

If you just want to be notified of new releases of xymon, please subscribe to the xymon-announce mailing list. This is a moderated list, used only for announcing new xymon releases. To be added to the list, send an e-mail to **xymon-announce-subscribe@hswn.dk**.

2.8 xymon Wiki Book

[Yan] System Monitoring with xymon wiki book.

It is a community effort for xymon documentation.

1. User Guide http://en.wikibooks.org/wiki/System_Monitoring_with_xymon/User_Guide
2. User Guide Administration Guide http://en.wikibooks.org/wiki/System_Monitoring_with_xymon/Administration_Guide
3. Developer Guide http://en.wikibooks.org/wiki/System_Monitoring_with_xymon/Developer_Guide
4. Other Docs http://en.wikibooks.org/wiki/System_Monitoring_with_xymon/Other_Docs

2.9 HOBBIT SERVER DAEMONS

These daemons implement the core functionality of the xymon server:

1. *xymond(8)* is the core daemon that collects all reports about the status of your hosts. It uses a number of helper modules to implement certain tasks such as updating logfiles and sending out alerts: *xymond_client*, *hobbitd_history*, *hobbitd_alert* and *xymond_rrd*. There is also a *hobbitd_filestore* module for compatibility with Big Brother.
2. *xymond_channel(8)* Implements the communication between the xymon daemon and the other Xymon server modules.
3. *xymond_history(8)* Stores historical data about the things that xymon monitors.
4. *xymond_rrd(8)* Stores trend data, which is used to generate graphs of the data monitored by xymon.
5. *xymond_alert(8)* handles alerts. When a status changes to a critical state, this module decides if an alert should be sent out, and to whom.
6. *xymond_client(8)* handles data collected by the xymon clients, analyzes the data and feeds back several status updates to Xymon to build the view of the client status.
7. *xymond_hostdata(8)* stores historical client data when something breaks. E.g. when a webpage stops responding *xymond_hostdata* will save the latest client data, so that you can use this to view a snapshot of how the system state was just prior to it failing.

2.10 HOBBIT NETWORK TEST TOOLS

These tools are used on servers that execute tests of network services.

1. *xymonping(1)* performs network connectivity (ping) tests.
2. *bbtest-net(1)* runs the network service tests.
3. *bbretest-net.sh(1)* is an extension script for re-doing failed network tests with a higher frequency than the normal network tests. This allows xymon to pick up the recovery of a network service as soon as it happens, resulting in less downtime being recorded.

2.11 HOBBIT TOOLS HANDLING THE WEB USER-INTERFACE

These tools take care of generating and updating the various xymon web-pages.

1. *bbgen(1)* takes care of updating the xymon webpages.
2. *xymonsvc.cgi(1)* This CGI program generates an HTML view of a single status log. It is used to present the xymon status-logs.
3. *xymongraph.cgi(1)* This CGI program generates graphs of the trend-data collected by xymon.
4. *xymon-hostgraphs.cgi(1)* When you want to combine multiple graphs into one, this CGI lets you combine graphs so you can e.g. compare the load on all of the nodes in your server farm.
5. *xymon-nkview.cgi(1)* Generates the Critical Systems view, based on the currently critical systems and the configuration of what systems and services you want to monitor when.
6. *bb-hist.cgi(1)* This CGI program generates a webpage with the most recent history of a particular host+service combination.
7. *bb-eventlog.cgi(1)* This CGI lets you view a log of events that have happened over a period of time, for a single host or test, or for multiple systems.
8. *bb-ack.cgi(1)* This CGI program allows a user to acknowledge an alert he received from xymon about a host that is in a critical state. Acknowledging an alert serves two purposes: First, it stops more alerts from being sent so the technicians are not bothered with more alerts, and secondly it provides feedback to those looking at the xymon webpages that the problem is being handled.
9. *xymon-mailack(8)* is a tool for processing acknowledgements sent via e-mail, e.g. as a response to an e-mail alert.
10. *xymon-enadis.cgi(8)* is a CGI program to disable or re-enable hosts or individual tests. When disabling a host or test, you stop alarms from being sent and also any outages do not affect the SLA calculations. So this tool is useful when systems are being brought down for maintenance.
11. *bb-findhost.cgi(1)* is a CGI program that finds a given host in the xymon webpages. As your xymon installation grows, it can become difficult to remember exactly which page a host is on; this CGI script lets you find hosts easily.
12. *bb-rep.cgi(1)* This CGI program triggers the generation of xymon availability reports, using *bbgen(1)* as the reporting back-end engine.
13. *bb-replog.cgi(1)* This CGI program generates the detailed availability report for a particular host+service combination.

14. *bb-snapshot.cgi(1)* is a CGI program to build the xymon webpages in a “snapshot” mode, showing the look of the webpages at a particular point in time. It uses *bbgen(1)* as the back-end engine.
15. *xymon-statusreport.cgi(1)* is a CGI program reporting test results for a single status but for several hosts. It is used to e.g. see which SSL certificates are about to expire, across all of the xymon webpages.
16. *bb-csvinfo.cgi(1)* is a CGI program to present information about a host. The information is pulled from a CSV (Comma Separated Values) file, which is easily exported from any spreadsheet or database program.

2.12 CLIENT-SIDE TOOLS

1. *logfetch(1)* is a utility used by the xymon Unix client to collect information from logfiles on the client. It can also monitor various other file-related data, e.g. file metadata or directory sizes.
2. *clientupdate(1)* Is used on xymon clients, to automatically update the client software with new versions. Through this tool, updates of the client software can happen without an administrator having to logon to the server.
3. *msgcache(8)* This tool acts as a mini xymon server to the client. It stores client data internally, so that the *xymonfetch(8)* utility can pick it up later and send it to the xymon server. It is typically used on hosts that cannot contact the xymon server directly due to network- or firewall-restrictions.

2.13 HOBBIT COMMUNICATION TOOLS

These tools are used for communications between the xymon server and the xymon clients. If there are no firewalls then they are not needed, but it may be necessary due to network or firewall issues to make use of them.

1. *bbproxy(8)* is a proxy-server that forwards xymon messages between clients and the xymon server. The clients must be able to talk to the proxy, and the proxy must be able to talk to the xymon server.
2. *xymonfetch(8)* is used when the client is not able to make outbound connections to neither bbproxy nor the xymon server (typically, for clients located in a DMZ network zone). Together with the *msgcache(8)* utility running on the client, the xymon server can contact the clients and pick up their data.

2.14 OTHER TOOLS

1. *xymonlaunch(8)* is a program scheduler for xymon. It acts as a master program for running all of the xymon tools on a system. On the Xymon server, it controls running all of the server tasks. On a xymon client, it periodically launches the client to collect data and send them to the xymon server.
2. *bb(1)* is the tool used to communicate with the xymon server. It is used to send status reports to the xymon server, through the custom xymon/BB protocol, or via HTTP. It can be used to query the state of tests on the central xymon server and retrieve Xymon configuration files. The server-side script *bbmessage.cgi(1)* used to receive messages sent via HTTP is also included.
3. *bbcmd(1)* is a wrapper for the other xymon tools which sets up all of the environment variables used by xymon tools.
4. *bbhostgrep(1)* is a utility for use by xymon extension scripts. It allows an extension script to easily pick out the hosts that are relevant to a script, so it need not parse a huge *bb-hosts* file with lots of unwanted test-specifications.
5. *bbhostshow(1)* is a utility to dump the full *bb-hosts(5)* file following any “include” statements.
6. *bbdigest(1)* is a utility to compute message digest values for use in content checks that use digests.

7. *bbcombotest(1)* is an extension script for the xymon server, allowing you to build complicated tests from simpler xymon test results. E.g. you can define a test that uses the results from testing your webserver, database server and router to have a single test showing the availability of your enterprise web application.
8. *trimhistory(8)* is a tool to trim the xymon history logs. It will remove all log entries and optionally also the individual status-logs for events that happened before a given time.

2.15 VERSIONS

1. *Version 1* of bbgen was released in November 2002, and optimized the webpage generation on Big Brother servers.
2. *Version 2* of bbgen was released in April 2003, and added a tool for performing network tests.
3. *Version 3* of bbgen was released in September 2004, and eliminated the use of several external libraries for network tests, resulting in a significant performance improvement.
4. *Version 4.0* released on March 30 2005, the project was de-coupled from Big Brother, and the name changed to xymon. This version was the first full implementation of the xymon server, but it still used the data collected by Big Brother clients for monitoring host metrics.
5. *Version 4.1* was released in July 2005 included a simple xymon client for Unix. Logfile monitoring was not implemented.
6. *Version 4.2* was released in July 2006, and includes a fully functional xymon client for Unix.

2.16 COPYRIGHT

xymon is Copyright(C)2002-2007,HenrikStorner<henrik@storner.dk> Parts of the xymon sources are from public-domain or other freely available sources. These are the the Red-Black tree implementation, and the MD5-, SHA1-, SHA2- and RIPEMD160-implementations. Details of the license for these is in the README file included with the xymon sources. All other files are released under the GNU General Public License version 2, with the additional exemption that compiling, linking, and/or using OpenSSL is allowed. See the file COPYING for details.

2.17 SEE ALSO

1. xymond(8)
2. xymond_channel(8)
3. xymond_history(8)
4. xymond_rrd(8)
5. xymond_alert(8)
6. xymond_client(8)
7. xymond_hostdata(8)
8. xymonping(1)
9. bbtest-net(1)
10. bbretest-net.sh(1)

11. bbgen(1)
12. xymonsvc.cgi(1)
13. xymongraph.cgi(1)
14. xymon-hostgraphs.cgi(1)
15. xymon-nkview.cgi(1)
16. bb-hist.cgi(1)
17. bb-eventlog.cgi(1)
18. bb-ack.cgi(1)
19. xymon-mailack(8)
20. xymon-enadis.cgi(8)
21. bb-findhost.cgi(1)
22. bb-rep.cgi(1)
23. bb-replog.cgi(1)
24. bb-snapshot.cgi(1)
25. xymon-statusreport.cgi(1)
26. bb-csvinfo.cgi(1)
27. logfetch(1)
28. clientupdate(1)
29. msgcache(8)
30. bbproxy(8)
31. xymonfetch(8)
32. xymonlaunch(8)
33. bb(1)
34. bbmessage.cgi(1)
35. bbcmd(1)
36. bbhostgrep(1)
37. bbhostshow(1)
38. bbdigest(1)
39. bbcombotest(1)
40. trimhistory(8)
41. bb-hosts(5)
42. xymonlaunch.cfg(5)

- 43. xymonserver.cfg(5)
- 44. xymon-alerts.cfg(5)
- 45. xymon-clients.cfg(5)
- 46. client-local.cfg(5)

Chapter 3

How to setup custom graphs

3.1 How to setup custom graphs

This document walks you through the setup of custom graphs in your xymon installation. Although Xymon comes with pre-defined setups for a lot of common types of graphs, it is also extensible allowing you to add your own tests. For many kinds of tests, it is nice to view them over a period of time in a graph - this document tells you how to do that.

3.1.1 Make a script to collect the data

First create your test data. Typically, this is an extension script that sends in some data to xymon, using a **status** or **data** command. If you use **status**, it will show up as a separate column on the display, with a green/yellow/red color that can trigger alerts. If you use **data**, xymon just collects the data into a graph - you must go to the **trends** column to see the graph. For this example, we'll use **status**.

So we create an extension script. Here is an example script; it picks two numbers out of the Linux kernel's memory statistics, and reports these to xymon.

```
#!/bin/sh

cat /proc/slabinfo | \
  egrep "dentry_cache|inode_cache" | \
    awk '{print $1 " : " $3*$4}' >/tmp/slab.txt

$BB $BBDISP "status $MACHINE.slab green `date`"

`cat /tmp/slab.txt'
"

exit 0
```

3.1.2 Get xymonlaunch to run the script

Save this script in `xymon/client/ext/slab`, and add a section to the `hobbit/client/etc/clientlaunch.cfg` to run it every 5 minutes:

```
[slabinfo]
```

```

ENVFILE /usr/lib/xyomon/client/etc/hobbitclient.cfg
CMD /usr/lib/xyomon/client/ext/slab
INTERVAL 5m

```

(On the xyomon server itself, you must add this to the file `xyomon/server/etc/hobbitlaunch.cfg`)

3.1.3 Check that the script data arrives in xyomon

After a few minutes, a **slab** column should appear on your xyomon view of this host, with the data it reports. The output looks like this:

```
Sun Nov 20 09:03:44 CET 2005
```

```
inode_cache : 330624
dentry_cache : 40891068
```

3.1.4 Arrange for the data to be collected into an RRD file

This is obviously a name-colon-value formatted report, so we'll use the NCV module in xyomon to handle it. Xyomon will find two datasets here: The first will be called **inodecache**, and the second **dentrycache** (note that xyomon strips off any part of the name that is not a letter or a number; Xyomon also limits the length of the dataset name to 19 letters max. since RRD will not handle longer names). To enable this, on the Hobbit server edit the `xyomon/server/etc/hobbitserver.cfg` file. The TEST2RRD setting defines how Hobbit tests (status columns) map to RRD datafiles. So you add the new test to this setting, by adding **slab=ncv** at the end:

```
TEST2RRD="cpu=la,disk,<...lots more stuff...>,xymond,mysql=ncv,slab=ncv"
```

slab is the status column name, and **=ncv** is a token that tells xyomon to send these data through the built-in NCV module.

By default, the xyomon NCV module expects data to be some sort of counter, e.g. number of bytes sent over a network - it uses the RRD **DERIVE** datatype by default, which is for data that is continuously increasing in value. Some data are not like that - the data in our test script is not - and for those data you'll have to make an extra setting to tell xyomon what RRD data type to use. The RRDtool rrdcreate(1) man-page has a detailed description of the various RRD datatypes. It is available online at <http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/doc/rrdcreate.en.html>

Our test script provides data that goes up and down in value (it is the number of bytes of memory used for a Linux kernel buffer), and for that kind of data we'll use the RRD **GAUGE** datatype. So we add an extra setting to `xyomonserver.cfg`:

```
NCV_slab="inodecache:GAUGE,dentrycache:GAUGE"
```

This tells the `xymond_rrd` module that it should create an RRD file with two datasets of type GAUGE instead of the default (DERIVE). The setting must be named `NCV_<columnname>`.

The `xyomonserver.cfg` file is not reloaded automatically, so you must restart xyomon after making these changes. Or at least, kill the `hobbit_rrd` processes (there are usually two) - `hobbitlaunch` will automatically restart them, and they will then pick up the new settings.

3.1.5 Check that the RRD collects data

The next time the **slab** status is updated, xymon will begin to collect the data. You can check this by looking for the **slab.rrd** file in the `xymon/data/rrd/HOSTNAME/` directory. If you want to check the data it collects, the **rrdtool dump** `hobbit/data/rrd/HOSTNAME/slab.rrd` will tell you what it got:

```
<!-- Round Robin Database Dump -->
<rrd>
<version> 0001 </version>
<step> 300 </step> <!-- Seconds -->
<lastupdate> 1132474725 </lastupdate> <!-- 2005-11-20 09:18:45 CET -->

<ds>
<name> inodecache </name>
RRD datatype----><type> GAUGE </type>
<minimal_heartbeat> 600 </minimal_heartbeat>
<min> 0.000000000e+00 </min>
<max> NaN </max>

<!-- PDP Status -->
current value----><last_ds> 330624 </last_ds>
<value> 0.000000000e+00 </value>
<unknown_sec> 0 </unknown_sec>
</ds>
```

If you go and look at the status page for the **slab** column, you should not see any graph yet, but a link to **xymon graph ncvs:slab**. One final step is missing.

3.1.6 Setup a graph definition

The final step is to tell xymon how to create a graph from the data in the RRD file. This is done in the `xymon/server/etc/hobbitgraph.cfg` file.

```
[slab]
TITLE Slab info
YAXIS Bytes
DEF:inode=slab.rrd:inodecache:AVERAGE
DEF:dentry=slab.rrd:dentrycache:AVERAGE
LINE2:inode#00CCCC:Inode cache
LINE2:dentry#FF0000:Dentry cache
COMMENT:\n
GPRINT:inode:LAST:Inode cache \: %5.1lf%s (cur)
GPRINT:inode:MAX: \: %5.1lf%s (max)
GPRINT:inode:MIN: \: %5.1lf%s (min)
GPRINT:inode:AVERAGE: \: %5.1lf%s (avg)\n
GPRINT:dentry:LAST:Dentry cache\: %5.1lf%s (cur)
GPRINT:dentry:MAX: \: %5.1lf%s (max)
GPRINT:dentry:MIN: \: %5.1lf%s (min)
GPRINT:dentry:AVERAGE: \: %5.1lf%s (avg)\n
```

[slab] is the name of this graph, and it must match the name of your status column if you want the graph to appear together with the status. The TITLE and YAXIS settings define the graph title and the legend on the Y-axis. The rest are definitions for the rrdgraph(1) tool - you should read the RRDtool docs if you want to know in detail how it works. For now, all you need to know is that you must pick out the data you want from the RRD file with a DEF line, like

```
DEF:inode=slab.rrd:inodecache:AVERAGE
```

which gives you an “inode” definition that has the value from the **inodecache** dataset in the slab.rrd file. This is then used to draw a line on the graph:

```
LINE2:inode#00CCCC:Inode cache
```

The line gets the color #00CCCC (red-green-blue), which is a light greenish-blue color. Note that you can have several lines in one graph, if it makes sense to compare them. You can also use other types of visual effects, e.g. stack values on top of each other (like the **vmstat** graphs do) - this is described in the rrdgraph man-page. An online version is at <http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/doc/rrdgraph.en.html>.

The GPRINT lines at the end of the graph definition also uses the **inode** value to print a summary line showing the current, maximum, minimum and average values from the data that has been collected.

Once you have added this section to xymongraph.cfg, refresh the status page in your browser, and the graph should show up.

3.1.7 Add the graph to the collection of graphs on the trends column

If you want the graph included with the other graphs on the **trends** column, you must add it to the GRAPHS setting in the xymon/server/etc/hobbitserver.cfg file.

```
GRAPHS="la,disk,<... lots more ...>,bbproxy,xymond,slab"
```

Save the file, and when you click on the **trends** column you should see the slab graph at the bottom of the page.

3.1.8 Common problems and pitfalls

If your graph nearly always shows 0

You probably used the wrong RRD datatype for your data - see step 4. By default, the RRD file expects data that is increasing constantly; if you are tracking some data that just varies up and down, you must use the RRD GAUGE datatype. Note that when you change the RRD datatype, you must delete any existing RRD files - the RRD datatype is defined when the RRD file is created, and cannot be changed on the fly.

No graph on the status page, but OK on the trends page

Make sure you have **ncv** listed in the GRAPHS setting in xymonserver.cfg. (Don’t ask why - just take my word that it must be there).

- o

Chapter 4

Installing xymon

4.1 Installing xymon

This describes how to setup a xymon server for monitoring your systems. It assumes that you are setting up a full xymon server - i.e. either you do not have a Big Brother server, or you will replace it completely with xymon.

Note to Big Brother users: Although some of the xymon tools have evolved from the bbgen toolkit that was used on top of a Big Brother server installation, the Xymon versions of these tools now require that you run Hobbit - not Big Brother. If you are migrating from Big Brother to Hobbit, then you should follow the migration guide.

4.1.1 Prerequisites - before you install xymon

There are a few things you should check before you begin to install xymon. Dont be scared of the number of items here - it is likely that you already have most or all of it in place. **A webbrowser capable of handling HTML 4, JavaScript and CSS**

This includes most browsers available today - Internet Explorer 5 or later, all Mozilla/Firefox versions, Konqueror, Netscape 6 and several others. The old Netscape 4.x browsers are known NOT to work. **A Unix-like operating system**

xymon is written for Unix-based systems, e.g. Linux, FreeBSD, or Solaris. It will probably work on any Unix-like system that supports the Unix System V IPC mechanisms (shared memory, semaphores) - that should be just about anything Unix-like you are likely to have. **Sufficient SYSV IPC ressources on your system**

xymon uses 8 shared memory segments, ranging in size from 32 KB to 512 KB (2336 KB total) in the default configuration; and 8 sets of 3 semaphores. Experience shows that some systems need tuning to provide the necessary IPC ressources that Xymon uses. Specifically, when installing on Solaris you must increase the "shmseg" kernel parameter from the default 6 to at least 8. Since other programs on your system may also use shared memory, a higher value may be required. See <http://www.hswn.dk/xymonon/2005/08/msg00183.html> for more information about these issues. **A webserver**

xymon is designed with a web-based front-end. So you should have a webserver such as Apache running on the server where you install Xymon. **A working C compiler, GNU make**.

xymon is written in C, so you need a working C compiler, e.g. gcc. You will also need a "make" utility - many systems have one by default, but you need to use the GNU make utility. On some systems, this is pre-installed as "gmake" or "gnumake". The configure-script checks this for you.

HP-UX users should note that the HP-supplied C compiler is known to mis-compile the lib/environ.c file, and produces an output file lib/environ.o of length 0 bytes. HP-UX users on the xymon mailing list agree that the default C compiler shipped with HP-UX should not be used to compile xymon - it is only for re-building the HP-UX kernel. The GNU C compiler works fine on HP-UX. More details in this e-mail from the Xymon mailing list. **PCRE, RRDtool, libpng, OpenSSL, OpenLDAP, Net-SNMP libraries**.

xymon relies on a number of Open-Source libraries - these must be installed before you start building Xymon. On many systems you already have these pre-installed - they are commonly installed by default on Linux systems, and FreeBSD has all of them in the "ports" collection.

Note: Although many systems have these libraries pre-installed, they often include only the run-time libraries and not the files that are needed to compile and build programs such as xymon. So if you think you have all of these libraries installed but Xymon will not build, do check that you have the development files installed as well. Often these are in packages called “something-dev”.

- PCRE - Perl Compatible Regular Expression library - is a library for matching text-strings. It is available from <http://www.pcre.org/>
- RRDtool is a library for handling the Round-Robin Databases used to hold the historical data xymon gathers. It is available from <http://oss.oetiker.ch/rrdtool/>. Xymon is known to work with RRDtool 1.0.x - if you prefer to use the newer RRDtool 1.2.x, make sure you use at least version 1.2.2.
- libpng is a library for generating images in the PNG format. It is used by RRDtool (and hence xymon). You can find it at <http://www.libpng.org/pub/png/libpng.html>
- OpenSSL is a library for communicating with network services, that use SSL encryption - e.g. secure websites. Although this library is not absolutely required for xymon, I strongly recommend that you install it because sooner or later you will probably need it anyway. It is available from <http://www.openssl.org/>. Note: If you are building on Solaris, you should check that you have a random-data generator, either the prngd daemon (available on Sun Freeware) or the Solaris /dev/random driver from Solaris patch 112438.
- OpenLDAP is used to query LDAP directory servers. If you would like to test that your directory server is up and running, you will need this library. It is available from <http://www.openldap.org/>
- Net-SNMP is used to query SNMP-based systems, typically network devices like routers, switches or firewalls. But nearly all types of computer systems have some sort of SNMP support. If you would like to collect data from SNMP-enabled systems you will need this library. It is available from <http://net-snmp.sourceforge.net/>

The configure-script will attempt to locate all of these libraries on your system, and complain if the required ones are missing. **A “xymon” userid on your system**

A core element of xymon is a network daemon. To keep your system secure and limit the amount of damage that can be done if someone finds a security problem in Xymon, I strongly recommend that you create a dedicated userid for the Hobbit programs. This user should *not* be a member of any other groups on your system.

xymon will install the xymonping tool as **setuid-root** (only on the xymon server). This program requires root privileges to be able to perform network “ping” tests. It will drop root privileges immediately after obtaining the network socket needed for this, and will not run with root privileges at all while handling network traffic or doing file I/O.

4.1.2 Building xymon

After unpacking xymon from the tar-file, run the configure script. This script asks a series of questions, but all of the questions have a reasonable default response. So if you are in doubt about what to answer, use the default setting. You can see what it looks like.

When the configure script finishes, it tells you to run make to build the xymon programs. If your default “make” tool is not GNU make, you should use the command for running GNU make instead, e.g. gmake. You will now see a lot of commands being run to build the programs, it usually takes a minute or two.

When it is finished, you finish the installation by running make install.

The first time you run make install, besides installing the xymon programs it also creates the default directory structure used by Xymon, and installs an initial set of configuration files that you can use as the basis for setting up monitoring of your entire network.

It is safe to run make install when upgrading a xymon server. It installs the programs, adds new template-files that were not present in your previous version, and updates your configuration files with any new sections that have been added. Any changes you have made yourself are preserved.

4.1.3 Configuring your webserver

xymon uses a web-based front-end. So you need to configure your webserver so that it knows where the Xymon webpages can be found, and what CGI scripts can run as part of Hobbit. This usually means adding a few lines to your webserver configuration that sets up a URL which points at the `/server/www/` directory, and which tells your webserver that the `/cgi-bin/` directory holds CGI scripts that the webserver should run when they are requested.

If you are using the Apache webserver, you will find the necessary additions to the Apache configuration in `/server/etc/xymon-apache.conf` - it looks like this. After changing the webserver configuration, you probably need to restart the webserver.

If you configured xymon to put the Administration CGI scripts into a separate directory (recommended for better security), you will also need to setup the password-file that controls access to this directory. Use the `htpasswd` command both to create the password file and to add or delete users:

```
# /usr/sbin/htpasswd -c /usr/local/xymon/server/etc/hobbitpasswd admin
New password:
Re-type new password:
Adding password for user admin
#
```

The `-c` option should only be used the first time, to create the password file. See the Apache documentation for details about how to use `htpasswd`.

4.1.4 Starting xymon

You can now login as the “xymon” user, and run the command `./server/xymon.sh start` to start xymon. After a few seconds, it should have started and you now have the following processes running:

Quite a few, but all of them controlled by the master xymonlaunch process. A quick run-down of what each of them does:

- xymond is the network daemon that receives status updates from the clients and the network test tool. It also provides the current status of all your systems to the tool that generates the webpages.
- xymond_channel provides the communication between hobbitd and all of the helper modules that implement other server-based functions.
- xymond_history takes care of recording the history of status changes for each item you monitor. This is used to track what has happened with a single status over time - when it was red, when it was green, what the error reported at 2:51 AM last Friday looked like. The history file format is compatible with the format used by the Big Brother package.
- xymond_filestore stores files with information about the current status of the systems monitored by xymon. There may be several of these running, but normally you will only need the one that stores information about hosts that have been disabled, which is the one you see here.
- xymond_alert takes care of sending out alerts when your servers begin to report a critical status.
- xymond_rrd updates the RRD database files with the numeric data collected from the status reports, to track e.g. how the disk utilization of a server changes over time. There are two of these processes, because the data can arrive in two different ways.

After a couple of minutes, you should have data available for the xymon server itself. If you open a webbrowser with the Xymon URL - usually <http://your.server/xymon/> - you should see something like this:

Each of the little faces indicate an item that is being monitored for this host. Here you see the default set of items that the xymon installation sets up for a xymon server:

Figure 4.1: xymonprocs.png

```

osiris:~/hobbit $ ps aux|grep hobbit|grep -v grep
27558 ? S 0:00 /usr/local/hobbit/server/bin/hobbitlaunch --config=/usr/local/hobbit/server/etc/hobbittasks.cfg --env=/usr/local/hobbit/server/etc/hobbitserver.env --log=/var/log/hobbit/hobbitlaunch.log --pidfile=/var/log/hobbit/hobbitlaunch.pid
27559 ? S 0:00 \_ hobbitd --restart=/usr/local/hobbit/server/tmp/hobbitd.chk --checkpoint-file=/usr/local/hobbit/server/tmp/hobbitd.chk --checkpoint-interval=600 --purple-conn=conn --log=/var/log/hobbit/hobbitd.log --admin-senders=127.0.0.1 127.0.0.1
27563 ? S 0:00 \_ hobbitd_channel --channel=stachg --log=/var/log/hobbit/history.log hobbitd_history
27564 ? S 0:00 | \_ hobbitd_history
27565 ? S 0:00 \_ hobbitd_channel --channel=enadis --log=/var/log/hobbit/enadis.log hobbitd_filestore --enadis
27566 ? S 0:00 | \_ hobbitd_filestore --enadis
27567 ? S 0:00 \_ hobbitd_channel --channel=page --log=/var/log/hobbit/page.log hobbitd_alert
27568 ? S 0:00 | \_ hobbitd_alert
27569 ? S 0:00 \_ hobbitd_channel --channel=status --log=/var/log/hobbit/larrd-status.log hobbitd_larrd --rrddir=/usr/local/hobbit/data/rrd
27570 ? S 0:00 | \_ hobbitd_larrd --rrddir=/usr/local/hobbit/data/rrd
27571 ? S 0:00 \_ hobbitd_channel --channel=data --log=/var/log/hobbit/larrd-data.log hobbitd_larrd --rrddir=/usr/local/hobbit/data/rrd
27572 ? S 0:00 \_ hobbitd_larrd --rrddir=/usr/local/hobbit/data/rrd
osiris:~/hobbit $ █

```

- *bbd* is the availability of the xymon network daemon.
- *bbgen* is the status of the bbgen tool, which updates the webpages.
- *bbtest* is the status of the bbtest-net network tester that performs all of the network tests you configure in xymon.
- *conn* is a simple “ping” test of the host.
- *xymond* is the status of the xymon daemon, with statistics about how many monitored items are being tracked.
- *http* is the status of the HTTP-server running on the xymon server.
- *info* contains information about how the host is configured in xymon, such as what IP-address it has, what network tests are being run against this host etc.
- *trends* is a collection of the various RRD graphs available for this host.

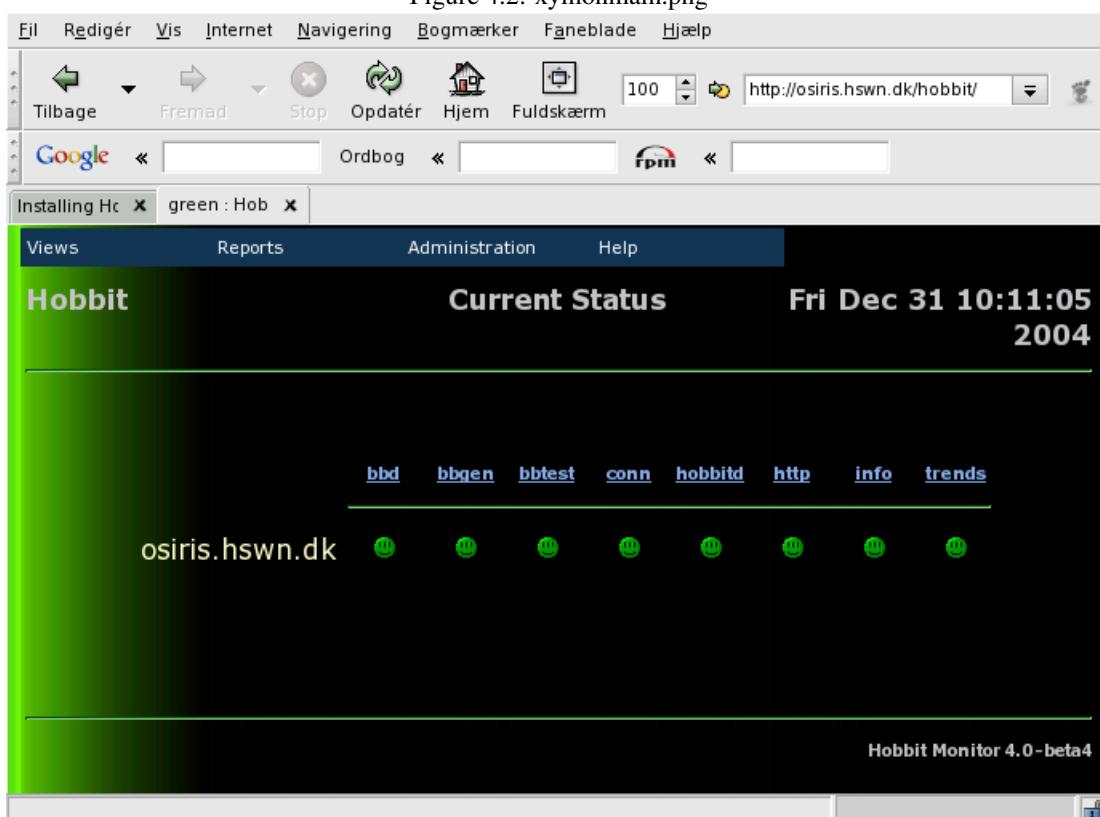
You can click on each of the green icons to see a more detailed status.

4.1.5 Next steps

Congratulations, you now have a running xymon server!

The next step is to configure it to monitor your servers and applications, and to set up the alerts to send you e-mail, call a pager, or send an SMS in case of trouble. For that, see the xymon configuration guide.

Figure 4.2: xymonmain.png



Chapter 5

About the xymon Monitor

5.1 About xymon

In this document:

- What is xymon ?
- Where can I download xymon ?
- Support
- Are there any other sites with xymon stuff?
- Who are you ?

What is xymon ?

xymon is a tool for monitoring servers, applications and networks. It collects information about the health of your computers, the applications running on them, and the network connectivity between them. All of this information is presented in a set of simple, intuitive webpages that are updated frequently to reflect changes in the status of your systems.

xymon is capable of monitoring a vast set of network services, e.g. mail-servers, web-servers (both plain HTTP and encrypted HTTPS), local server application logs, ressource utilisation and much more.

Much of the information is processed and stored in RRD files, which then form the basis for providing trend graphs showing how e.g. webserver response-times vary over time.

xymon was inspired by the Big Brother monitoring tool, a freely available tool from BB4 Technologies (now part of Quest Software) with some of the features that Xymon has. But Hobbit is better than Big Brother in many ways:

- xymon can handle monitoring lots of systems.

Big Brother is implemented mostly as shell-scripts, and performance suffers badly from this. In large networks where you need to monitor hundreds or thousands of hosts, processing of the data simply cannot keep up. Another problem with BB is that it stores all status-information in individual files; when you have lots of hosts and statuses, the amount of disk I/O triggered by this severely limits how many systems you can monitor with one BB server.

xymon avoids these performance bottlenecks by keeping most of the ever-changing data in memory instead of on-disk, and by being implemented in C rather than shell scripts.

- xymon has a centralized configuration.

xymon keeps **all** configuration data in one place: On the xymon server. Big Brother has lots of configuration files stored on the individual servers being monitored, so to change a setting you may need to logon to several servers and change each of them individually.

- xymon is easy to setup and deploy.

Big Brother has a huge number of add-ons, available from the www.deadcat.net site. This is both a blessing and a curse - you can find anything you need as an add-on, but many of the add-ons really ought to have been part of the base package. E.g. the ability to track historical performance data, simple things such as monitoring SSL-enabled services and SSL certificates, or just something as simple as a GUI for temporarily disabling monitoring of a system. Maintaining and improving all of these add-ons gets really complex.

xymon has all of these features built-in so you don't have to worry about getting the right add-ons and maintaining them - they come with the base package.

Also, when it comes to deploying the client-side packages, xymon clients require no configuration changes when you install them on multiple hosts. So you can setup a template client installation, and then blindly copy it to all of your hosts.

- xymon is actively being developed.

New xymon versions appear regularly, usually every 4-6 months. In contrast, development of Big Brother appears to have stopped - at least when it comes to the non-commercial (BTF) version.

- xymon is licensed as Open Source - Big Brother is not.

Although the BB "Better-than-Free" license permits the use of BB for non-commercial use without having to buy a license, it is still a non-free package in the Open Source sense. I fully respect the decision of the people behind Big Brother to choose the licensing terms they find best - just as I can choose the licensing terms that I find best for the software I develop. It is my sincere belief that an Open Source license works best for a project such as xymon, where community involvement is essential to get a tool capable of monitoring as many different systems as possible.

An interesting essay appeared recently, which tries to explain why Open Source is the natural way for a software product to evolve. If you are curious as to why the trend seems to be that more and more software exist in an Open Source version, I suggest you have a look at it.

Didn't you write something called "bbgen" ?

Yes I did. The **bbgen toolkit** was the name I used for xymon from 2002 until the end of 2004 (i.e. bbgen version 1.x, 2.x and 3.x). The bbgen versions relied on a Big Brother server to hold the monitoring data and status logs, and this turned out to be a real performance problem for me. So I needed to completely replace Big Brother with something more powerful. In March 2005 version 4 was ready and capable of operating without any need for a Big Brother server, so I decided to change the name to avoid any misunderstanding about whether this was an add-on to Big Brother, or a replacement for it. Xymon no longer has any relation to Big Brother.

Why did you call it xymon ?

Choosing a name is *hard*. I wanted a name that was easy to remember; could be interpreted as a somewhat meaningful acronym; and one that did not refer directly to the Big Brother origin.

"xymon" could mean "High-performance Open-source BB ImplementTation" but it might as well just be a name. If you're familiar with the Xymon's in Tolkien's books, you will know that xymons are very fond of things that are green - just like any systems- or network-administrator prefers his monitoring screen to be. They also pay a great deal of attention to what is happening around them, and are capable of doing things that you would not think they could when you first saw them. All of these characteristics apply well to the Hobbit monitor.

Wht should I use xymon ? My Big Brother setup works just fine.

It is your choice. I think xymon has many improvements over BB, so I would of course say 'Yes, I think you should'. But in the end it is You who have to deal with the hassle of setting up and learning a new system, so if you are comfortable with what Big Brother is doing for you now, I am not forcing you to switch. If you want to see what some of the Xymon users think about changing to Hobbit, check out this thread (continued here) from the Hobbit mailing list archive. The executive summary of those messages is that You won't regret switching.

So where can I download xymon?

The xymon sources are available on the project page at Sourceforge.

Support

There are two mailing lists about xymon:

- The **xymon@hswn.dk** mailing list is for general discussion about xymon. To avoid spam you must be a subscriber to the list before you are allowed to post mesages. To subscribe to the list, send an e-mail to xymon-subscribe@hswn.dk.
There is an archive of the list.
- The **xymon-announce** list is an announcement-list where new versions of xymon will be announced. You can subscribe to the list by sending an e-mail to xymon-announce-subscribe@hswn.dk.

If you have a specific problem with something that is not working, first check the list of known issues, and try to search the list archive. If you don't find the answer, post a message to the xymon mailing list - I try to answer questions about Xymon in that forum.

Are there any other sites with xymon stuff?

Several projects have sprung up around xymon:

- **BBWin** is a client for Microsoft Windows systems. It is available from the BBWin project page at Sourceforge.
- **DevMon** is a tool to collect data from SNMP-capable devices. It is available from the DevMon project page at SourceForge.
- **The Shire** is a repository of xymon add-on scripts, utilities, sample configurations etc. which is currently being established. At the time of writing (August 2006) this is just being started, but you can check out The Shire project page at SourceForge.
- The xymon Wiki has some information about Xymon usage.
- **Deadcat** is a repository for Big Brother extensions. Although these were written for Big Brother, most of these can be used with xymon with little or no extra work since Xymon is compatible with the Big Brother extensions. See the Deadcat site.

Who are you ?

My name is Henrik Storner. I was born in 1964, and live in Copenhagen, the capital of Denmark which is a small country in the northern part of Europe. I have a M.Sc. in Computer Science from the University of Copenhagen, and have been working with computers and Unix systems professionally since 1984. I have been developing bits and pieces of Open Source software for the past 10 years - you'll find my name in the Linux kernel CREDITS file - and I am actively involved in the local Linux Users Group SSLUG, one of the largest LUG's world-wide, where I am a systems administrator for their Internet servers (web, e-mail, news).

I started using Big Brother around 1998, for monitoring a bunch of servers that I was administering. In late 2001 I began working for the CSC Managed Web Services division in Copenhagen, and one of my first tasks was to improve on the monitoring and SLA reporting. After looking at what the standard tools could do, I decided to setup a Big Brother system as a demonstration of what could be done. This was an immediate success. Systems were rapidly added to the Big Brother monitor, and I began to see some of the scalability problems that happen when you go from monitoring 50 servers to monitoring 500 (not to mention the 2500 hosts we are currently - 2006 - keeping tabs on). So I decided it was time to do something about it, and during the autumn and early winter 2002 bbgem was born. The rest is history.

Chapter 6

Configuring xymon Alerts

6.1 Configuring xymon Alerts

When something breaks, you want to know about it. Since you probably don't have the xymon webpages in view all of the time, Xymon can generate alerts to draw your attention to problems. Alerts can go out as e-mail, or Hobbit can run a script that takes care of activating a pager, sending an SMS, or however you prefer to get alerted.

- A simple alert configuration
- Configuration file keywords
- Using regular expressions for names
- Alerting via a script
- Using macros
- There are rules ... and exceptions: IGNORE

A simple alert configuration

The configuration file for the xymon alert module is `/server/etc/xymon-alerts.cfg`. This file consists of a number of *rules* that are matched against the name of the host that has a problem, the name of the service, the time of day and a number of other criteria. Each rule then has a number of *recipients* that receive the alert. For each recipient you can further refine the rules that need to be matched. An example:

```
HOST=www.foo.com
MAIL webmaster@foo.com SERVICE=http REPEAT=1h
MAIL unixsupport@foo.com SERVICE(cpu,disk,memory)
```

The first line defines a *rule* for alerting when something breaks on the host “www.foo.com”. There are two recipients: webmaster@foo.com is notified if it is the “http” service that fails, and the notification is repeated once an hour until the problem is resolved. unixsupport@foo.com is notified if it is the “cpu”, “disk” or “memory” tests that report a failure. Since there is no “REPEAT” setting for this recipient, the default is used which is to repeat the alert every 30 minutes.

OK, suppose now that the webmaster complains about getting e-mails at 4 AM in the morning. The webserver is not supposed to be running between 9 PM and 8 AM, so even though there is a problem, he doesn't want to hear about it until 7:30 - that gives him just enough time to fix the problem. So you must modify the rule so that it doesn't send out alerts until 7:30 AM:

```
HOST=www.foo.com
MAIL webmaster@foo.com SERVICE=http REPEAT=1h TIME=*:0730:2100
MAIL unixsupport@foo.com SERVICE(cpu,disk,memory)
```

Adding the *TIME* setting on the recipient causes the alerts *for this recipient* to be suppressed, unless the time of day is within the interval. So with this setup, the webmaster gets his sleep.

What would have happened if you put the *TIME* setting on the *rule* instead of on the *recipient*? Like this:

```
HOST=www.foo.com TIME=*:0730:2100
MAIL webmaster@foo.com SERVICE=http REPEAT=1h
MAIL unixsupport@foo.com SERVICE(cpu,disk,memory)
```

Well, the webmaster would still have his nights to himself - but the *TIME* setting would then also apply to the alerts that go out when there is a problem with the “cpu”, “disk” or “memory” services. So there would not be any mails going to *unixsupport@foo.com* when a disk fills up during the night.

Keywords in rules and recipients

These are the keywords for setting up rules:

These are the keywords for specifying a recipient:

Wildcards - regular expressions

So now we can setup an alert. But using explicit hostnames is bothersome, if you have many hosts. There is a smarter way:

```
HOST=%(www|intranet|support|mail).foo.com
MAIL webmaster@foo.com SERVICE=http REPEAT=1h
MAIL unixsupport@foo.com SERVICE(cpu,disk,memory)
```

The percent-sign indicates that the hostname should not be taken literally - instead, *(www—intranet—support—mail).foo.com* is a *Perl-compatible regular expression*. This particular expression matches “*www.foo.com*”, “*intranet.foo.com*”, “*support.foo.com*” and “*mail.foo.com*”. You can use regular expressions to match hostnames, service-names and page-names.

If you want to test how your alert configuration handles a specific host, you can run *xymond_alert* in **test** mode - you give it a hostname and servicename as input, and it will go through the configuration and tell you which rules match and who gets an alert.

```
osiris:~ $ cd server/
osiris:/server $ ./bin/bbcmcmd xymond_alert --test osiris.hswndk.cpu
Matching host:service:page 'osiris.hswndk:cpu:' against rule line 109:Matched
    *** Match with 'HOST=' ***
Matching host:service:page 'osiris.hswndk:cpu:' against rule line 110:Matched
    *** Match with 'MAIL henrik@example.com REPEAT=2 RECOVERED COLOR=red' ***
Mail alert with command 'mail -s "BB [12345] osiris.hswndk:cpu is RED" henrik@example.com'
```

If e-mail is not enough

The *MAIL* keyword means that the alert is sent in an e-mail. Sometimes this ends up being an SMS to your cell-phone - there are several “e-mail to SMS” gateways that perform this service - but that may not be what you want to do. And also, for an e-mail to actually be delivered requires that the mail-server is working. So if you need full control over how alerts are handled, you can use the *SCRIPT* method instead. Here’s how:

```
HOST=%(www|intranet|support|mail).foo.com SERVICE=http
SCRIPT /usr/local/bin/smsalert 4538761925 FORMAT=sms
```

This alert doesn’t go out as e-mail. Instead, when an alert needs to be delivered, xymon will run the script `/usr/local/bin/smsalert`. The script can use data from a series of environment variables to build the information it sends in the alert, depending on what the recipient can handle. E.g. for pagers you will typically just send a sequence of numbers - Xymon provides things like the IP-address of the server that has a problem and a numeric code for the service to the script. So a simple script to send an SMS alert with the “`sendsms`” tool could look like this:

```
#!/bin/sh

/usr/local/bin/sendsms $RCPT "$BBALPHAMSG"
```

Here you can see the script use two environment variables that xymon sets up for the script: The `$RCPT` is the recipient, i.e. the phone-number “4538761925” that is in the `xymonalerts.cfg` file. The `$BBALPHAMSG` is text of the status that triggers the alert.

Although `$BBALPHAMSG` is nice to have, not all recipients can handle the large messages that may be sent in the status message. The `FORMAT=sms` tells xymon to change the `BBALPHAMSG` into a form that is suitable for an SMS message - which has a maximum size of 160 bytes. So Xymon picks out the most important bits of the status message, and puts as much of that as possible into the `BBALPHMSG` variable for the script.

The full list of environment variables provided to scripts are as follows:

This set of environment variables are the same as those provided by Big Brother to custom paging scripts, so you should be able to re-use any paging scripts written for Big Brother with xymon.

Save on the typing - use macros

Say you have a long list of hosts or e-mail addresses that you want to use several times throughout the `xymon-alerts.cfg` file. Do you have to write the full list every time ? No:

```
$WEBHOSTS=%(www|intranet|support|mail).foo.com

HOST=$WEBHOSTS SERVICE=http
SCRIPT /usr/local/bin/smsalert 4538761925 FORMAT=sms

HOST=$WEBHOSTS SERVICE(cpu,disk,memory)
MAIL unixsupport@foo.com
```

The first line defines `$WEBHOSTS` as a *macro*. So everywhere else in the file, “`$WEBHOSTS`” is automatically replaced with “`%(www—inet—support—mail).foo.com`” before the rule is processed. The same method can be used for recipients, e.g. e-mail addresses. In fact, you can put an entire line into a macro:

```
$UNIXSUPPORT=MAIL unixsupport@foo.com TIME=*:0800:1600 SERVICE=cpu,disk,memory
```

```
HOST=%(www|intranet|support|mail).foo.com  
$UNIXSUPPORT
```

```
HOST=dns.bar.com  
$UNIXSUPPORT
```

would be a perfectly valid way of specifying that unixsupport@foo.com gets e-mailed about cpu-, disk- or memory-problems on the foo.com web-servers, and the bar.com dns-servers.

Note: Nesting macros is possible, except that you must define a macro before you use it in a subsequent macro definition.

There are rules ... and exceptions: IGNORE

A common scenario is where you handle most of the alerts with a wildcard rule, but there is *just* that one exception where you dont want any cpu alerts from the marketing server on Thursday afternoon. Then it is time for the IGNORE recipient:

```
HOST=* COLOR=red  
IGNORE HOST=marketing.foo.com SERVICE=cpu TIME=4:1500:1800  
MAIL admin@foo.com
```

What this does is it defines a general catch-all alert: All red alerts go off to the admin@foo.com mailbox. There is just one exception: When the marketing.foo.com alerts on the “cpu” status on Thursdays between 3PM and 6PM, that alert is ignored. The IGNORE recipient implicitly has a STOP flag associated, so when the IGNORE recipient is matched, xymon will stop looking for more recipients - so the next line with the MAIL recipient is never looked at when handling that busy marketing server on Thursdays.

Chapter 7

Configuring xymon Monitoring

7.1 Configuring xymon Monitoring

The xymon configuration is kept in the files in the `/server/etc/` directory. If you look at this directory, you will see these files:

- “`/etc/opt/xymonserver42/bb-hosts`” is the one you will change the most. This file contains a list of all the hosts you are monitoring, including information such as their IP-address, what network services you are monitoring on the host, what URL’s you are checking, what subpage in the xymon web-pages this host is shown on etc. The name of the file - “bb-hosts” - was chosen so it is compatible with the Big Brother system which uses the same filename and file format.
- “`/etc/opt/xymonserver42/xymon-clients.cfg`” is the configuration file for data reported by the xymon clients installed on the hosts you are monitoring. This defines the color of the cpu-, disk-, memory- and procs- columns, based on the information that is sent to xymon by the clients.
- “`/etc/opt/xymonserver42/xymon-alerts.cfg`” holds the alerting configuration. In this file, you setup the rules for sending out alerts about services going down: Who gets the alert, how is it sent, how often, whether to send alerts 24x7 or only between 10 AM and 4 PM on weekdays etc.
- “`/etc/opt/xymonserver42/xymonserver.cfg`” is the configuration file for the xymon server. This file defines a lot of environment variables that are made available to all of the xymon programs when they run. Some environment variables that are defined in the Big Brother system are also setup by xymon, so that Big Brother extension scripts will work.

The initial configuration of xymonserver.cfg is setup by the configure script when you install xymon, and in most cases you will not need to change it.

- “`/etc/opt/xymonserver42/xymonlaunch.cfg`” is the configuration file for the hobbitlaunch tool. xymonlaunch is the master program in xymon, it is the only program you start to run the xymon server. xymonlaunch reads the xymonlaunch.cfg file, and starts the programs listed here to run the server. Some of the programs may run as daemons, some of the programs may run at regular intervals. If you want to use some of the advanced options for the bbgen or bbttest-net programs, you change the xymonlaunch.cfg file to add these options to the commandline.
- “`/etc/opt/xymonserver42/xymongraph.cfg`” is a configuration file for the xymongraph CGI. It defines how the graphs are generated from the data in the RRD files.
- bb-services is a configuration file for the bbttest-net program. It defines how network services are checked.

7.2 Setting up monitoring of hosts

The bb-hosts file defines which hosts xymon monitors. When you install xymon, a simple configuration is setup that just lists the xymon server:

```
-bash-2.05b$ cd ~/server/etc/
-bash-2.05b$ cat bb-hosts
#
# Master configuration file for Hobbit and Big Brother
#
# This file defines several things:
#
# 1) By adding hosts to this file, you define hosts that are monitored
#    by Hobbit / Big Brother.
# 2) By adding "page", "subpage", "group" definitions, you define the layout
#    of the Hobbit / Big Brother webpages, and how hosts are divided among
#    the various webpages that Hobbit / Big Brother generates.
# 3) Several other definitions can be done for each host, see the bb-hosts(5)
#    man-page.
#
# You need to define at least the Hobbit server itself here.

80.62.63.88 www.hswn.dk      # BBDISPLAY BBPAGER BBNET bbd http://www.hswn.dk/

-bash-2.05b$ .
```

There are a few things to notice here:

- Lines that begin with a # are comments.
- Each host you monitor is on a line by itself, with the IP-address and the hostname of the host.
- You can add extra tags to each host definition, by putting in a #-mark and then some keywords. These keywords define how xymon handles the host.

The bb-hosts file shown in the example has only one host defined: `www.hswn.dk` which is the server running xymon. There are a few extra keywords thrown in:

- `BBDISPLAY`, `BBPAGER`, `BBNET` are compatibility settings for extensions written for Big Brother. xymon doesn't use these, but puts them in the bb-hosts file to avoid problems if you mix xymon and Big Brother modules.
- `bbd` is the name of a *network test*. This keyword causes the xymon network-tester `bbtest-net` to check if the `bbd` network service is running on this host, and send a status report to the xymon server with the information about this service. So you'll get a (hopefully!) green icon on the xymon webpage for this host, showing the status of the `bbd` network service.

Network services are defined in the `bb-services` file, so this file must have an entry for `bbd` defining what TCP port to check, and possibly also what data to send to the service and what to expect as a response.

- `http://www.hswn.dk/` is a URL, obviously. This also triggers a network test, the xymon network tester will try to request this URL, and send in a status report showing if the URL was accessible or not.

By default, xymon will always check if the host is up and running by trying to “ping” it. This results in a `conn` column on the xymon webpage for this host, showing if the ping-test succeeded. If you have a host that does not respond to ping - e.g. because there is a firewall that filters out such requests - then you can disable the ping-test by putting a “noconn” keyword on the line in bb-hosts.

As you can see, the syntax is pretty straight-forward. Need to monitor an extra URL for this server ? Just add the URL to the line. Need to check if ssh (Secure Shell) is running ? Just add `ssh`

Table 7.1: Monitored Network Services

conn	Simple ping test. Enabled by default, you can disable it by putting “noconn” into bb-hosts.
http	Web-server test. Enter the URL to request from the webserver.
ftp	FTP server test.
ssh	SSH (Secure Shell) server test. Supports ssh1 and ssh2.
telnet	Telnet server test.
smtp	SMTP (Mail server) test.
pop3	POP-3 test.
imap	IMAP test. IMAP version 2 and 4 are supported, for version 3 use “imap3”.
nntp	NNTP (News) server test.
ldap	LDAP (Directory server) test. Enter the full LDAP URI if xymon is configured with LDAP support.
rsync	rsync server test.
bdb	Big Brother daemon test. Also works with the xymon network daemon.
clamd	CLAM anti-virus daemon test.
spamd	SpamAssassin anti-spam daemon test.
oratns	Oracle TNS listener test. Will attempt to do an oratns “ping”.
qsmtp	QSMTP server test. For qmail’s qmtpd service.
qmqp	QMQP server test. For qmail’s mqpd service.

Table 7.2: Monitored Network Services with SSL enabled

https	Web-server test. Enter the URL to request from the webserver.
ftps	Secure FTP server test.
telnets	Secure Telnet server test.
smtps	Secure SMTP server test.
pop3s	Secure POP-3 server test.
imaps	Secure IMAP server test.
nntps	Secure NNTP (News) server test.
ldaps	Secure LDAP (Directory) server test.

to the line. The full set of keywords you can use is described in the bb-hosts man-page. Many of the keywords relate to the way xymon displays the information about the host on the web-pages, other keywords deal with how the uptime percentage is calculated for availability reports, and some keywords - like the *bdb* and *http://...* mentioned above - describe the network services that are tested for this host.

Monitoring network services

As shown in the example above, adding a network test for a host is as simple as putting the right keyword into the bb-hosts file. The default set of network tests configured in xymon 4.0 is as follows:

If xymon is built with OpenSSL support, the following SSL-enabled services can also be checked:

Enter the full LDAP URI if xymon is configured with LDAP support. Note that this is only possible when xymon is built with the OpenLDAP v2.x client library, and only for LDAP servers that support LDAP version 3 and the “starttls” command. LDAP server that use the older non-standard method of tunnelling LDAP through SSL on port 636 will not work.

There are a few network tests that xymon can run for you, by using external programs. This is not a very effective way of testing, so it is only done this way for a few very specialised tests:

Table 7.3: Other Monitored Network Services

ntp	NTP (Network Time protocol) server test, using the “ntpd” command.
rpc	RPC service test. This queries the <i>portmapper</i> service on the server, using the “rpcinfo” command. See the bb-hosts(5) man-page.

Monitoring host-specific data with clients

You can install a client on each of the hosts you monitor, to check host-specific data such as CPU utilisation, disk usage, if certain processes and services are running etc. xymon includes clients for most Unix-like operating systems. A client for Windows is planned but the programming has not yet started.

First, make sure you have installed the xymon client on all of the hosts you want to monitor, and you have these hosts listed in your bb-hosts file. The Xymon client will pick up the hostname of the box it is running on automatically, but it is not uncommon for the name it finds to be different from what you've put into bb-hosts. So if you know that the client is running but no data appears, check that the hostname used by the Hobbit client is the one you expect. See this FAQ item for details.

With the xymon client running and reporting data into Xymon, you should see the cpu-, disk-, memory- and procs- columns appear. The color of these status columns is determined by settings in the xymon-clients.cfg configuration file. Here is an example of how to setup a host:



The screenshot shows a window titled 'Session' with a menu bar including 'Redigér', 'Vis', 'Bogmærker', 'Opsætning', and 'Hjælp'. The main area displays host-specific data for 'voodoo.hswn.dk'. The data is organized into sections: 'HOST=voodoo.hswn.dk', 'UP 30m', 'LOAD 6.0 12.0', 'DISK /export/netdisk 93 98', 'DISK /export/home 85 95', 'DISK * 70 85', and 'PROC' entries for cron, apache2, svscan, hobbittd_history, hobbittd_rrd, sshd, dnscache, dhcpcd, hobbittd, and hobbittaunch. Below the data, there are icons for a sun and a square, followed by the word 'Skal'.

```

HOST=voodoo.hswn.dk
UP 30m
LOAD 6.0 12.0
DISK /export/netdisk 93 98
DISK /export/home 85 95
DISK * 70 85
PROC cron 1 -1 yellow
PROC apache2 1 -1 yellow
PROC svscan 1 -1 yellow
PROC *^hobbittd_history 1 -1 yellow
PROC *^hobbittd_rrd 2 -1 yellow
PROC sshd
PROC dnscache
PROC dhcpcd
PROC hobbittd
PROC hobbittaunch

```

As you can see, there's first a definition of what hosts the following criteria applies to. Here, it is only a single host: voodoo.hswn.dk - but you can use various filters on hostnames, pagenames and time of day to determine what the thresholds should be for each of the criteria monitored with the client data. The xymon-clients.cfg man-page describes this in detail.

After the host filter comes the criteria used to determine the color of each of the status columns.

UP

LOAD

column color, based on how much load is on the system. After the LOAD keyword you put two limits: The first number is the limit

DISK

The name of the filesystem is the mount point. You can specify this either with the full path, or you can use * meaning "all filesystems"

PROC

MEMSWAP

LOG

More about logfile monitoring

Configuring the LOG entries in the xymon-clients.cfg file is only one half of the configuration - you also need to tell the xymon client running on the monitored system that it must send in some data from that logfile in the first place. For that, you must configure the client-local.cfg file with the name of the logfile.

Upgrading from Big Brother to xymon

7.3 Upgrading from Big Brother to xymon

First, you should realize that this is not a fully automated process. You will need to do some work yourself - especially with the handling of alerts.

First step: Install xymon

To begin, install xymon as described in the Xymon installation guide. I recommend that you configure xymon to use the same user-ID as your current Big Brother installation, but have it use a different directory for the server- and data-files. The default is to use `/server` and `/data` respectively, which is unlikely to clash with the directories where you have Big Brother installed. If you do need to change the directories, you must edit the top-level Makefile and change the `BBHOME` and `BBVAR` settings near the top of the file.

Step two: Move the configuration files

A couple of configuration files can be copied directly from Big Brother to xymon:

- The **bb-hosts** file
- The **bb-services** file. You need only copy this if you have used bbgen before, and added custom network tests to the bb-services file.
- The **cookies** file. You may not have this file - it is only present if you have used bbgen before and have setup HTTP tests that require cookies.
- The **bbcombotests.cfg** file. You may not have this file - it is only present if you have used bbgen before and have setup the bbcombotest tool.

The `bbwarnrules.cfg` and `bbwarnsetup.cfg` files cannot be copied over. xymon uses a very different configuration file for the alert configuration, so you will have to re-write your alert configuration for xymon. See the xymon alert configuration to learn how Xymon alerts are configured.

Any server-side extension-scripts can be copied from the `$BBHOME/ext/` directory to the `/server/ext/` directory. You must also add entries for them to the xymon `xymonlaunch.cfg` file. Beware that many scripts rely on environment variables that Big Brother defines, but which xymon does not define - in that case, you need to setup those environment variables in the `xymonserver.cfg` file. It is probably easiest to save this until you start running xymon, and can look at any error-output from the scripts.

If you have modified the webpage header- and footer-files in `$BBHOME/web/` then you can copy the modified files over directly to the `/server/web/` directory. Note that xymon has a number of header- and footer-files for the various CGI scripts that are not present in Big Brother, so you may need to setup a few extra files to get a consistent look across your new xymon installation.

Step three: Stop Big Brother

You are now going to move over the data files. To avoid confusion about files being updated by Big Brother while they are being moved over to xymon, I recommend that you stop Big Brother now.

Step four: Move the history logs

You may want to save the historical logfiles and the history of your status changes. To do that, move all of the files or directories in the `$BBVAR/hist/` to the `/data/hist/` directory, and all of the files or directories in `$BBVAR/histlogs/` to the `/data/histlogs/` directory. If you prefer to keep them in the Big Brother directory, you can copy them over with “`cp -r`” or “`tar`” instead of moving them.

Step five: Move the RRD files

The RRD files are used to generate the graphs, if you have installed the LARRD add-on to Big Brother. xymon has RRD support built-in, and it is obviously nice to keep the historical data that has been collected over time.

The filesystem layout of the RRD files is different from Big Brother+LARRD to xymon. Instead of having all of the RRD files in one big directory, there is a subdirectory for each host holding only the RRD files for data from that host. This is easier to manage, and also speeds up the graph generation when you have many hosts. Unfortunately, it makes migrating from Big Brother to xymon slightly more complicated.

In the xymon source-tree, you will find a script *xymond/moverrd.sh*. This script moves or copies the RRD files from the Big Brother+LARRD structure into the xymon structure. You must edit a couple of settings at the beginning of the file, especially to set the correct directory where Big Brother stores your current RRD files (the SRCDIR setting). By default the script *copies* the files over to the new structure, if you would rather just move them then change to “OP” setting to “mv”.

After setting up the script, run it and it should copy all of the RRD-files that relate to a host currently in the bb-hosts file to the new directory structure.

Step 6: Start xymon

Start xymon with the */server/xymon.sh start* command. Look at the logfiles in the */var/log/xymon* directory (or elsewhere, if you did not choose the default logfile directory when configuring xymon) and fix any problems that show up.

Look at the webpages generated. For the first few minutes, there will be some missing columns and icons for each host, since it takes some time for all of the tests to report a status to the new xymon daemon. After 5-10 minutes all of the standard tests should appear.

Chapter 8

Integrating MRTG data into xymon

8.1 Integrating MRTG data into xymon

This document describes one way of integrating MRTG graphs into xymon. It's simple, doesn't require any additional scripts, and provides all of your MRTG graphs as part of the "trends" column that is already present for all hosts in Xymon.

Another way of doing this is the bb-mrtg.pl script. This is an extension script that gives you some more options for controlling where the graphs show up, and also lets you generate alerts based on data collected by MRTG.

Simple xymon-MRTG support

MRTG by default uses its own fileformat for the data files, and continuously generates PNG- or GIF-images of the data. This is a waste of ressources - most of the time, these images are never seen. This was in fact one of the reasons that RRDtool was developed, to separate the data-collection from the graph generation.

xymon uses the RRDtool format for all of its data. You can configure MRTG to save data using the RRDtool data format, instead of the default MRTG log-file format. This lets your MRTG save the data directly into the Xymon RRD directory, in the same format that all of the other Hobbit RRD files use. You can then use the normal Hobbit graph tools to view the graphs.

To configure MRTG to use the RRDtool format, you must setup the mrtg.cfg file like this at the top of the file:

```
# For xymon integration
WorkDir: /usr/local/xymon/data/rrd
LogFormat: rrdtool
```

Note that the WorkDir setting points to the top-level RRD directory, i.e. the one defined via the BBRRDS setting in xymonserver.cfg. The Logformat:rrdtool makes MRTG save data using the RRDtool data format.

Each of the network interfaces you monitor have a target-definition in the mrtg.cfg file. You need to modify this slightly, to make it save the RRD data file in a subdirectory matching the hostname you have in the bb-hosts file, and with a filename that begins with "mrtg.". Like this:

```
Target[mrtg.myrouter.eth0]: /10.0.0.1:public@myrouter.sample.com:
Directory[mrtg.myrouter.eth0]: myrouter.sample.com
```

This defines an MRTG target, where it monitors the interface on **myrouter.sample.com** that has the IP-address **10.0.0.1**. It uses the community name **public** to query the SNMP daemon on the router.

The **Directory[mrtg.myrouter.eth0]:myrouter.sample.com** instructs MRTG to save the data file in this directory relative to the **WorkDir** directory, i.e. the final directory for the RRD datafile will be /usr/local/xymon/data/rrd/myrouter.sample.com which is where xymon expects all of the RRD-files for the myrouter.sample.com host to be. The name of the RRD data-file will be **mrtg.myrouter.eth0.rrd** - i.e. the name of the target.

The reason for naming the data file **mrtg.*.rrd** is that the xymonograph tool has a built-in definition for generating graphs from this type of files. So if you stick to this naming convention, the graphs will automatically show up on the xymon “trends” page. If you have more than one device that you collect data from, you’ll need to modify this; you can use any name for the target as long as it is of the form **mrtg.*.DEVICE** - i.e. first “mrtg.”, then some random text (e.g. the hostname), then a dot and the device-name. The device-name is used as a legend on the graphs, so you probably want to make this something recognizable, like the name of the network interface, or some sensible description like “DSL”, “LAN”, “T1” or whatever you know your devices as. Note the MRTG converts this to lower-case.

Here is the full mrtg.cfg configuration used to track traffic on my Internet gateway (currently a 4 Mbit/512 Kbit ADSL). Note that even though MRTG does not use the Title and MaxBytes settings, they are required - MRTG will not run without them:

```
# For xymon integration
WorkDir: /var/lib/xymon/rrd
LogFormat: rrdtool

# The external interface on my router
Directory[mrtg.fenris.dsl]: fenris.hswndk
Target[mrtg.fenris.dsl]: /80.62.63.88:public@fenris:
Title[mrtg.fenris.dsl]: Traffic Analysis for External DSL
MaxBytes1[mrtg.fenris.dsl]: 500000
MaxBytes2[mrtg.fenris.dsl]: 62500

# The internal interface on my router
Directory[mrtg.fenris.lan]: fenris.hswndk
Target[mrtg.fenris.lan]: /10.0.0.1:public@fenris:
Title[mrtg.fenris.lan]: Traffic Analysis for internal LAN
MaxBytes[mrtg.fenris.lan]: 1250000
```

With this setup, I have the MRTG graphs readily available on the “trends” page, together with all of the other xymon graphs.

Running the MRTG data collector from xymonlaunch

Normally there is a cron job that runs the mrtg command every 5 minutes to collect the MRTG data. But you can run it from xymonlaunch - this also has the benefit that the RRD files will be owned by the hobbit user.

All that is needed is to add a section for MRTG to xymon’s xymonlaunch.cfg file. Mine looks like this:

```
[mrtg]
CMD /usr/bin/mrtg --lock-file $BB SERVERLOGS/mrtg.lock /etc/mrtg.cfg
INTERVAL 5m
LOGFILE $BB SERVERLOGS/mrtg.log
```

Some Linux distributions setup MRTG with the expectation that it will always be run by the root user. So you may have to change permissions on some files and directories e.g. to permit the xymon user to read the mrtg.cfg file. Check the mrtg.log file for errors.

Chapter 9

Critical Systems

9.1 Critical Systems

If you are monitoring lots of hosts, getting an overview of which hosts need attention can be difficult. Most likely you've split the hosts among several pages, and the "All non-green" view is just cramped full with systems where a logfile is showing some errors, a filesystem needs cleaning up etc.

The "Critical Systems" view lets you define exactly which tests on what hosts need attention. In other words, this is the view your Operations Center will be using to decide whether to call out people in the middle of the night. It might look like this:

This document describes how you configure the Critical Systems view, and how it works for your operators. By "operators" I mean the people who are doing the 24x7 monitoring. Where I work, these people normally do not resolve the issues - they just raise the trouble-tickets and assign them to the "engineer" on duty. It may be different in your organization.

9.2 The Critical Systems editor

To configure what goes on the Critical Systems view, you use a dedicated editor.

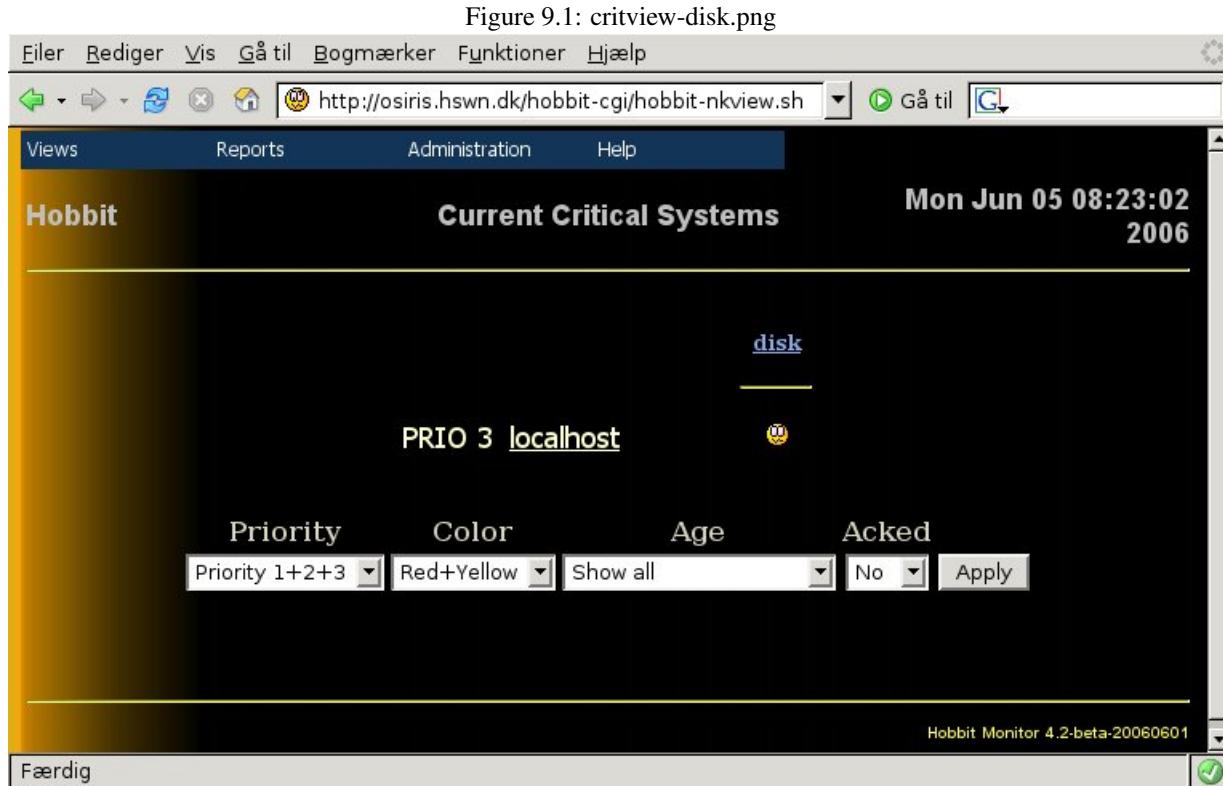
The default xymon setup has nothing on the critical systems view. So to use it, you must configure some of your systems and tests to be included on this view. From the **Administration** menu, pick the **Edit Critical Systems** item. This is usually in the password-protected area of xymon, so you will need to authenticate yourself before you are allowed access. If you haven't set this up yet, look at the installation guide to see how you do that.

After authenticating, you are presented with the editor page.

The editor form

Let me explain what the various fields are for:

- The **Host** and **Test** fields are text entry fields. This is where you enter the name of the host and test you want to configure. If you would rather not type too much, you can enter just the beginning of the hostname and use the **Search** and **Next** buttons to walk through the currently configured tests.
- The **Priority** field defines how important this test is. By default you have three priorities: 1, 2 and 3. Priority 3 is the lowest - things you must fix, but it can wait until you've had lunch or finished the department meeting. Priority 2 is for more important things, like one of your RAID systems running in degraded mode. Priority 1 is the highest priority - the kind of problem where you want to get a phonecall at 3 AM in the morning.
- Then there is a group of time-related settings. The **Monitoring time** defines when this test should show up on the Critical Systems view. By default, that will be 24 hours a day, 7 days a week. But you probably have some



systems that don't need attention during week-ends, or perhaps you only want to support a server during normal work-hours. Then you can use this setting to make sure it will only show up on the Critical Systems view during those periods. If you are migrating from the old "NK" settings in the bb-hosts file, this is the equivalent of the "NKTIME" setting.

The **Start monitoring** and **Stop monitoring** settings are used if you have systems that go into production at a certain date, or which are de-commisioned at a certain date. Instead of having to update your Critical Systems configuration exactly when that happens, you can configure the dates when monitoring of the systems should begin or end.

- The **Resolver group** is a text field. You can use it for your operations people to see which group of engineers they should call about this problem. If you have multiple groups handling different parts of your IT systems, use this to let the operations staff know whether to call the Unix admins, the DBA's or one of your Webmasters.
- The **Instruction** is a text entry field, where you can place a brief instruction to the operators handling the problem: If there is a simple thing that the operations people can try to fix the problem before calling the on-duty engineer, then you can place instructions here - e.g. perhaps the issue is with an external partner, so they just need to call them and let them know there is an issue. You can use HTML tags in this field, so if it's a long story then just put in an HTML link to another document.
- The **Clone** fields at the bottom of the form (not visible in the screenshot) are described later

Setting up a disk status

Right now, there is a yellow disk status on my system.

But it is not on the Critical Systems view, and I want it to be. It is a priority 3 event, and I only want it monitored between 7AM and 8PM on weekdays. Most likely it is just some logfiles that are filling up, so the operators can try

Figure 9.2: editor-main.png

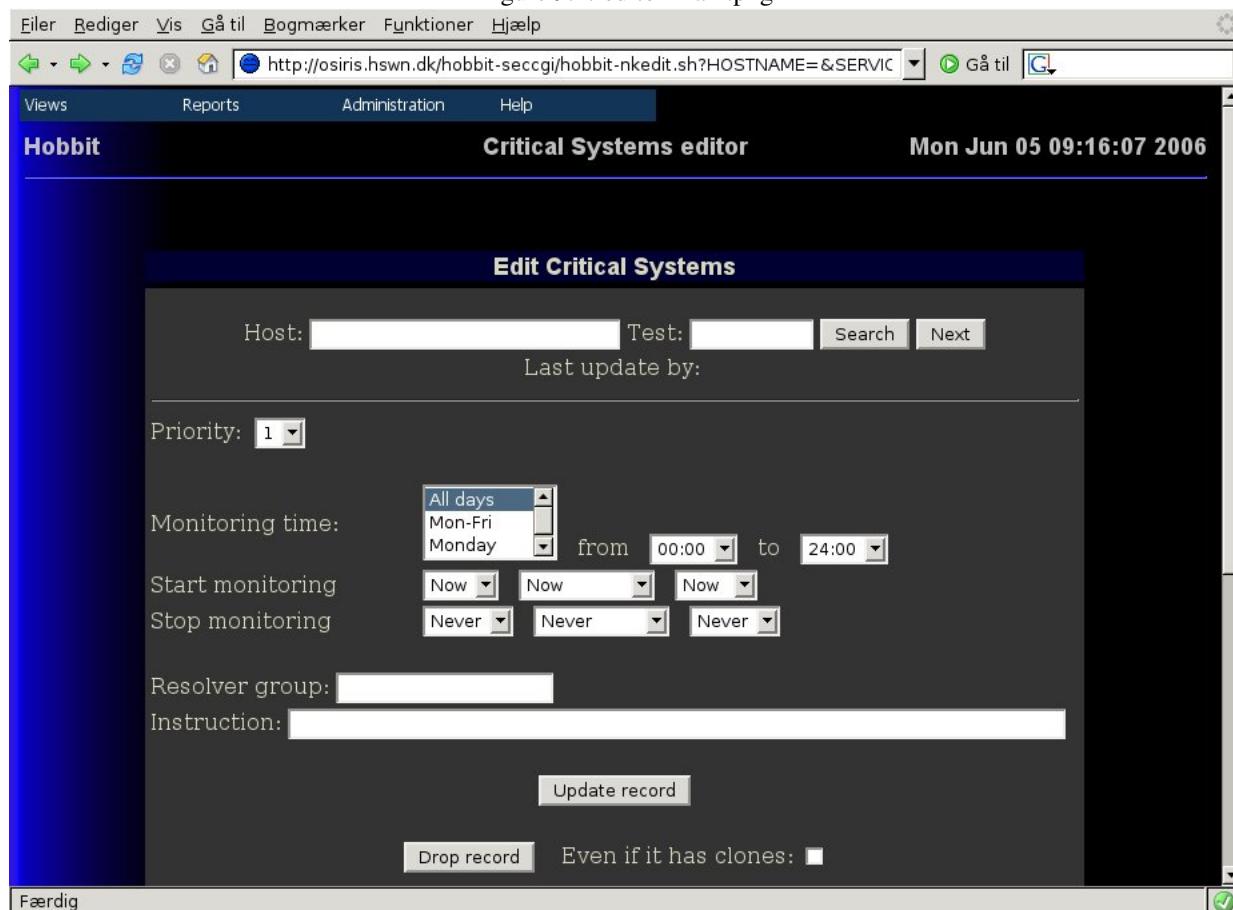
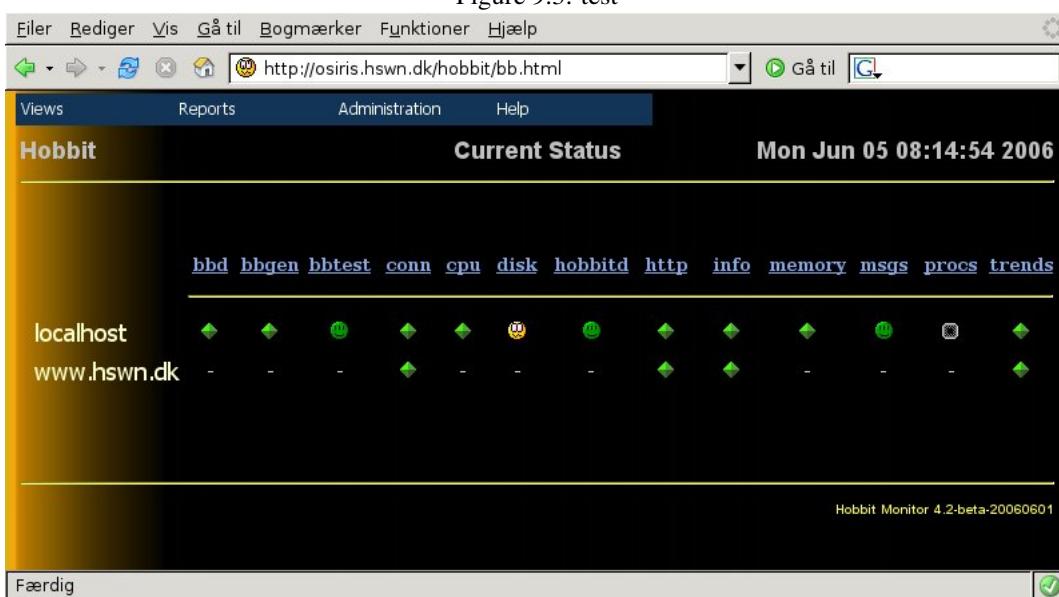
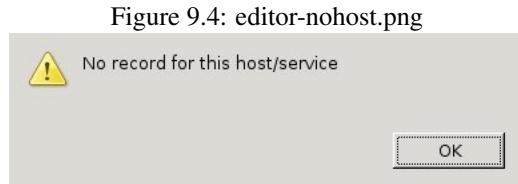


Figure 9.3: test





and clean out the `/var/log/` directory - if that doesn't solve the problem, then they must escalate it to the Unix admins.

So on the Critical Systems editor, I enter the hostname **localhost** and the test **disk**, then hit the **Search** button. I get this warning:

telling me that there is nothing configured yet for this host+test combination. If there had been any previous configuration, it would have shown up on the form.

So I fill out the fields of the form and hit the **Update** button. The form changes to look like this:

As you can see, there is now a **Last update** text showing who has changed this configuration, and when it was last done.

If I now go back to the Critical Systems view - from the menu, pick **Views** and **Critical Systems view** - you will see that the status is now showing up:

Template definitions - cloning records

If you have many hosts that share a common setup on the Critical Systems view, then editing all of them can be tiresome. Instead, you should define a template and then **clone** it to all of the hosts.

NOTE: A cloned definition is not a copy of the original definition. It is in fact a pointer back to the original definition, so if you change the original definition *after* you performed the cloning, then the clone definition will *also* change.

Defining a template is just like defining the Critical Systems view for a host. Just call the host something that looks like a template - "**Standard Unix**", for instance. So here is a definition for a **Unix cpu** template.

Now we have created the template (if you haven't pushed the **Update**

button to save the template, do it now). To apply this template to a host, scroll down to the bottom of the editor form, and enter the hostname that you want to apply the template to, then hit the **Add/remove clones** button:

After it has updated, you can see that "localhost" is now listed in the scrollbox showing the clones.

NOTE: Cloning happens at the **host** level, so even though we did the cloning from a **cpu** test definition, it will also affect all the other definitions we have for the **Standard Unix** host.

9.3 The Critical Systems view

The critical systems view lets the operators filter active alerts in several ways. It might look like this:

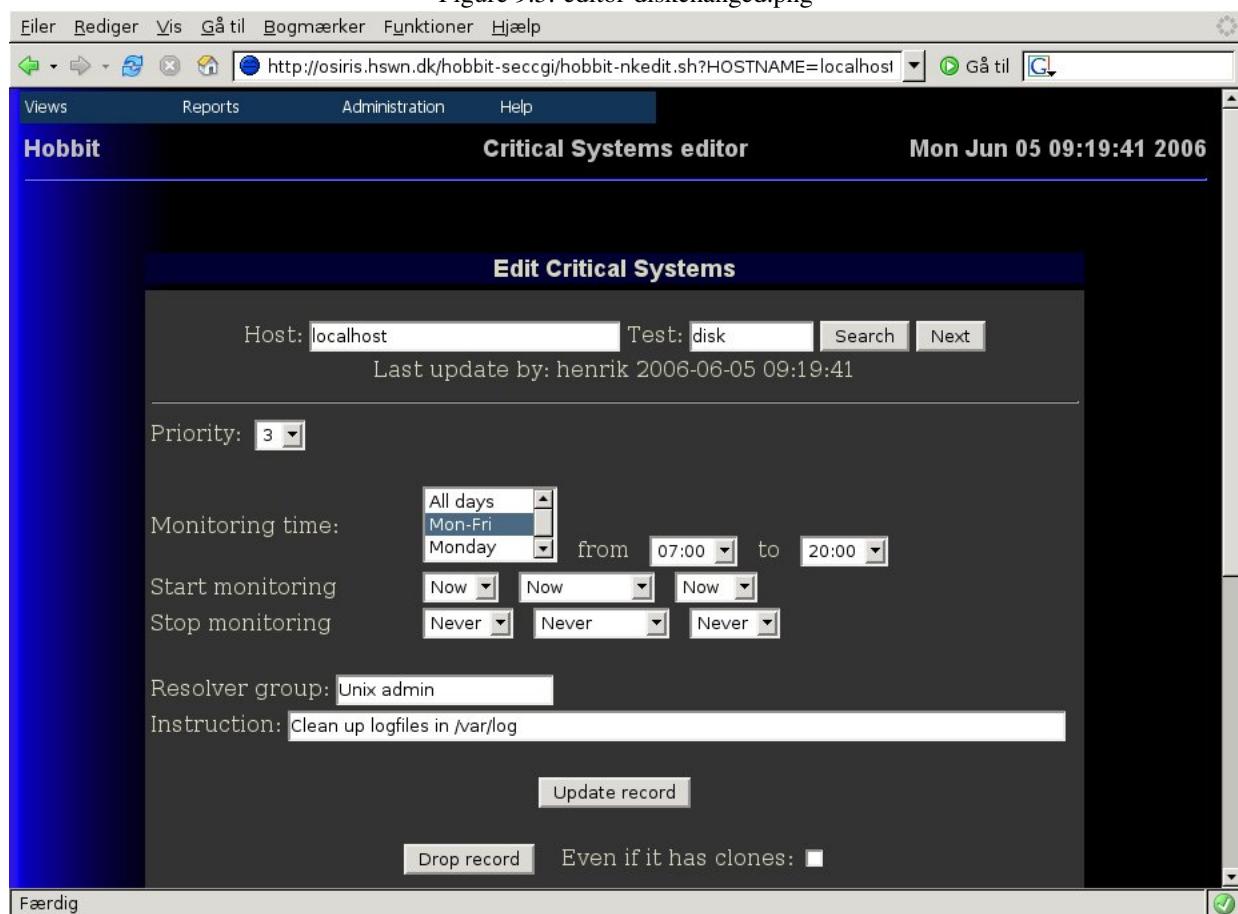
Filtering the Critical Systems view

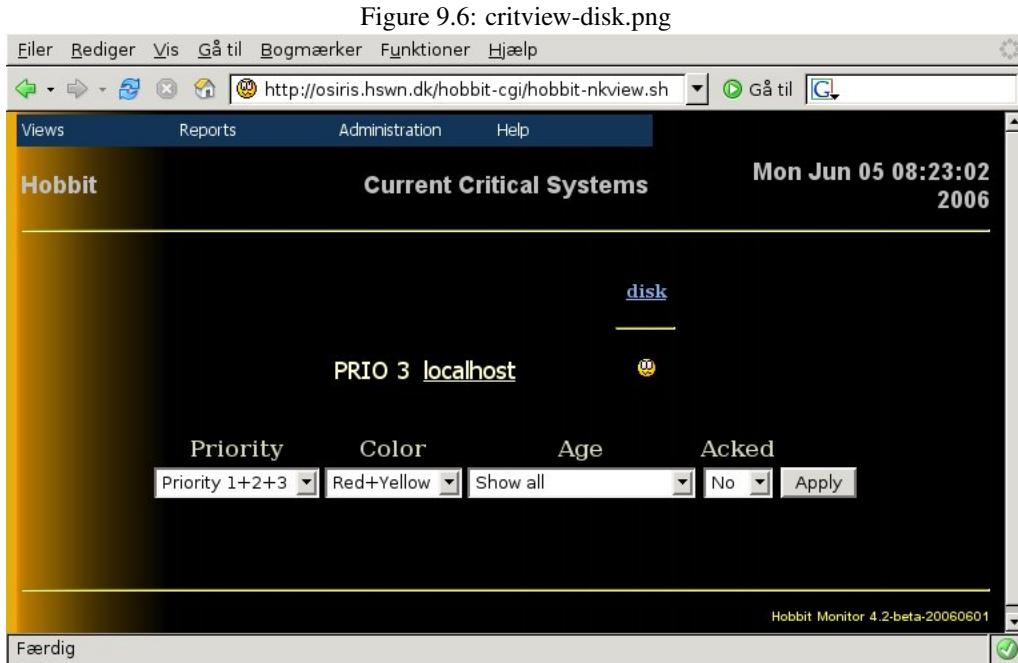
The drop-down boxes lets the operators filter the alerts that show up on the page.

- The **Priority** limits alerts so that only those with a matching priority get displayed.
- The **Color** removes those alerts that have an unwanted color
- The **Age** limit can be used to only see the most recent events.
- The **Acked** selection can be used to toggle the view of events that have been acknowledged by the operators.

Tip: If you have a preferred default setting for these, then you can bookmark it in your browser - the settings are part of the URL, so your bookmark will include the current settings.

Figure 9.5: editor-diskchanged.png





The detailed status view

When looking at the status of one of the items shown on the Critical Systems view, a number of additional items show up. On the example Critical Systems view above, you will notice that the instructions we entered about what to do with the disk status is shown here, so they are available to the operators. There are links to the host documentation and host information. There is also an acknowledge function, so that the operators can acknowledge an alert right away.

Critical Systems acknowledgment

From the detailed status view, the operator can **acknowledge** an alert, after he has assigned the problem to an engineer or has handled it in some other way. This serves two purposes: First, it removes the status from the Critical Systems view, so the operator can concentrate on the new problems that appear. And second, it lets everyone else see that the problem has been noticed and is being handled by someone.

When acknowledging an alert, the operator can add information about what the problem is, or who is handling it, and when it is expected to be resolved. E.g. like this:

The **Host-ack** checkbox lets the operator acknowledge all current alerts for a given host, e.g. a full disk could easily trigger alerts for both the disk-, msgs- and procs-statuses - a Host ack lets him handle all of those.

After the operator has acknowledged the status, the acknowledgment will be visible on the Critical systems status view:

(If you are wondering why this image says it is a “Level 1” acknowledgement, then the answer is that a future release of xymon will allow multiple acknowledgments by different groups. Level 1 is the operator who sees the alert on the Critical Systems view. Level 2 could be the engineer who gets paged by a xymon alert going out).

How acknowledgements are visible to everyone

The acknowledgments that the operator enters from the status page will show up on the status visible to everyone. E.g. here is how the overview page will appear to a normal user: Note that the “disk” status has a yellow checkmark, indicating that it has been acknowledged:

And the detailed status page also includes the acknowledgment information:

Figure 9.7: editor-clonemaster.png

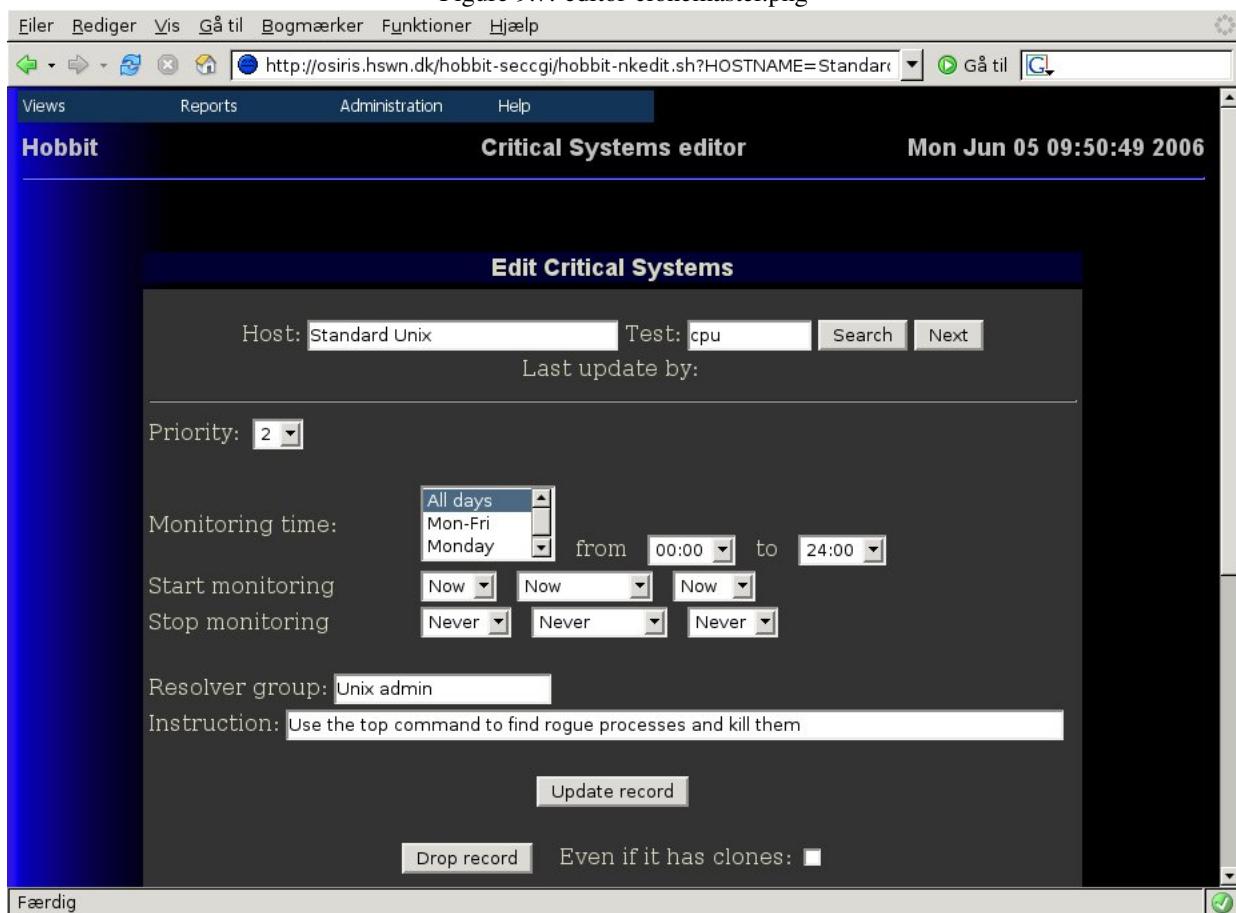


Figure 9.8: editor-makeclone.png

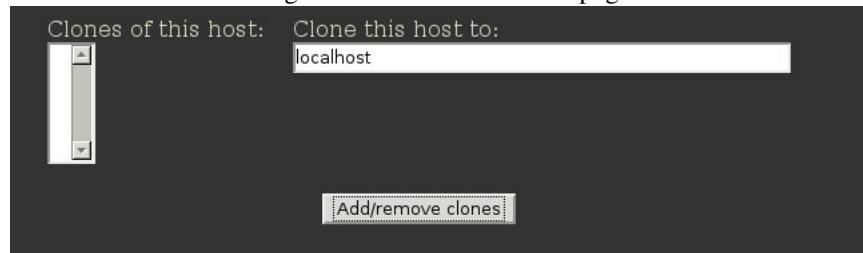


Figure 9.9: editor-showclone.png

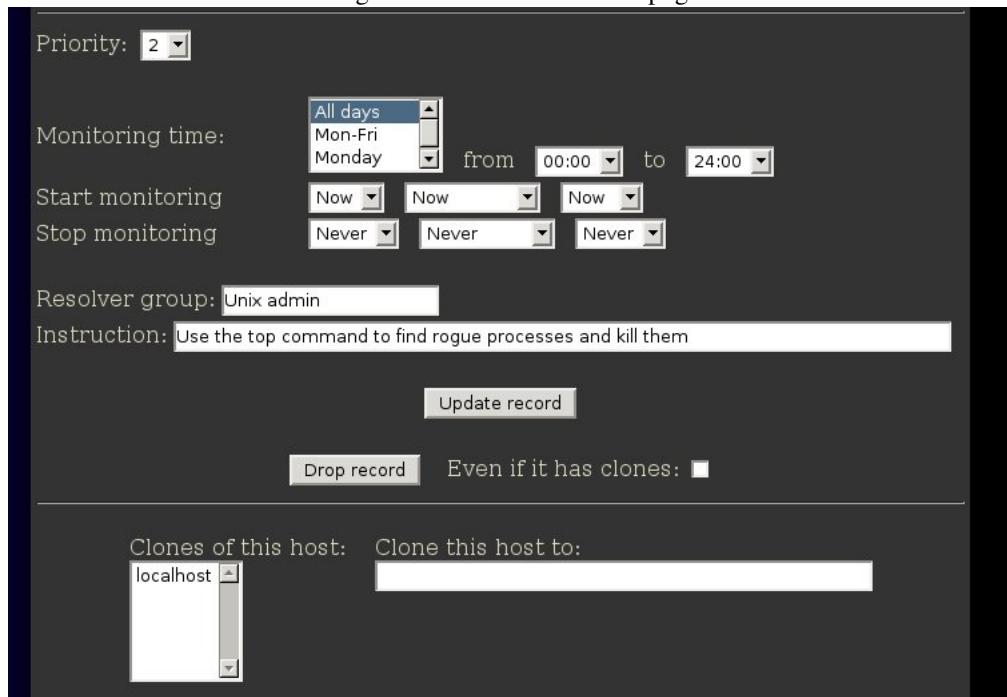


Figure 9.10: critview-disk.png

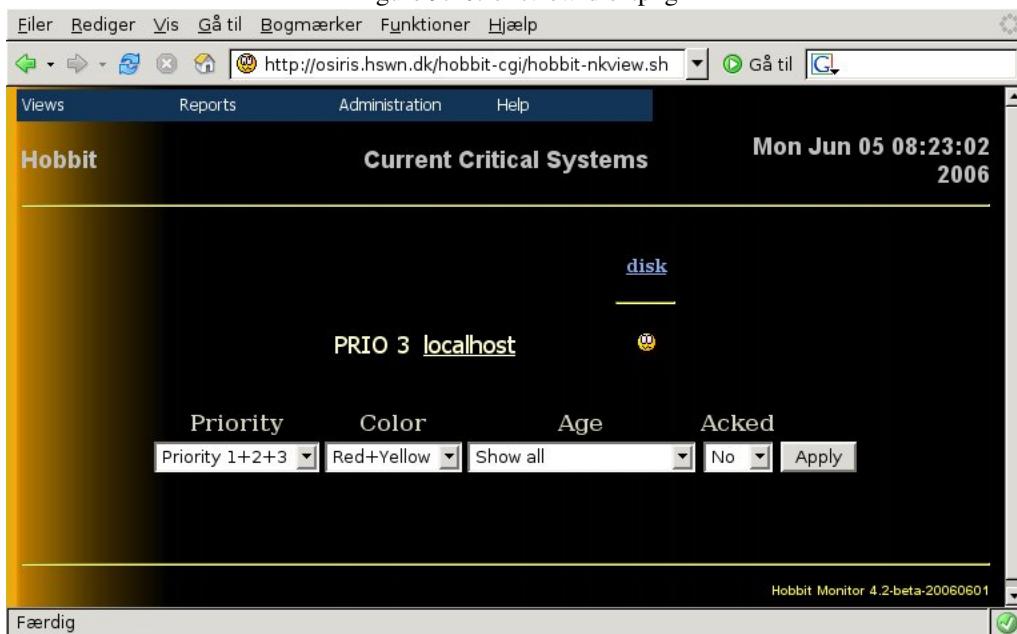


Figure 9.11: critview-detail-ackform.png

Hobbit		localhost - disk	Mon Jun 05 08:30:17 2006
Priority	Resolver group	Documentation	
3	Unix admin	Host info Host docs	
Clean up logfiles in /var/log			
Joe promised to take care of this right after lunch		Host-ack <input type="checkbox"/>	6 hours <input type="button" value="▼"/>
		<input type="button" value="Acknowledge"/>	

Figure 9.12: critview-detail-acked.png

Efiler Rediger Vis Gå til Bogmærker Funktioner Hjælp

Gå til http://osiris.hswn.dk/hobbit-cgi/bb-hostsvc.sh?HOST=localhost

Views Reports Administration Help

Hobbit localhost - disk Mon Jun 05 08:30:17 2006

Priority	Resolver group	Documentation
3	Unix admin	Host info Host docs
Clean up logfiles in /var/log		
Host-ack <input type="checkbox"/>		24 hours <input type="button" value="▼"/>
		<input type="button" value="Acknowledge"/>

Acknowledgments

Level From Validity	Message
1 henrik 2006-06-05 08:33 - 2006-06-05 14:33	Joe promised to take care of this right after lunch

Mon Jun 5 08:30:11 CEST 2006 - Filesystems NOT ok

! /work (68% used) has reached the WARNING level (50%)

Filesystem	1024-blocks	Used	Available	Capacity	Mounted on
/dev/hda2	99511580	10226564	89285016	11%	/
/dev/hda3	47856164	32313728	15542436	68%	/work

Færdig

Figure 9.13: mainview-acked.png

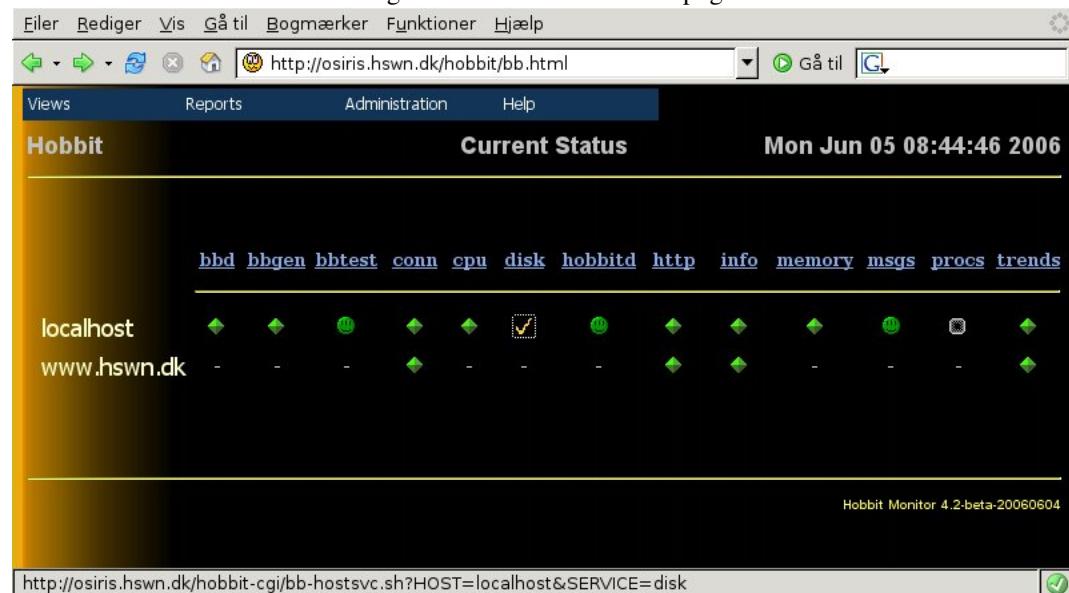
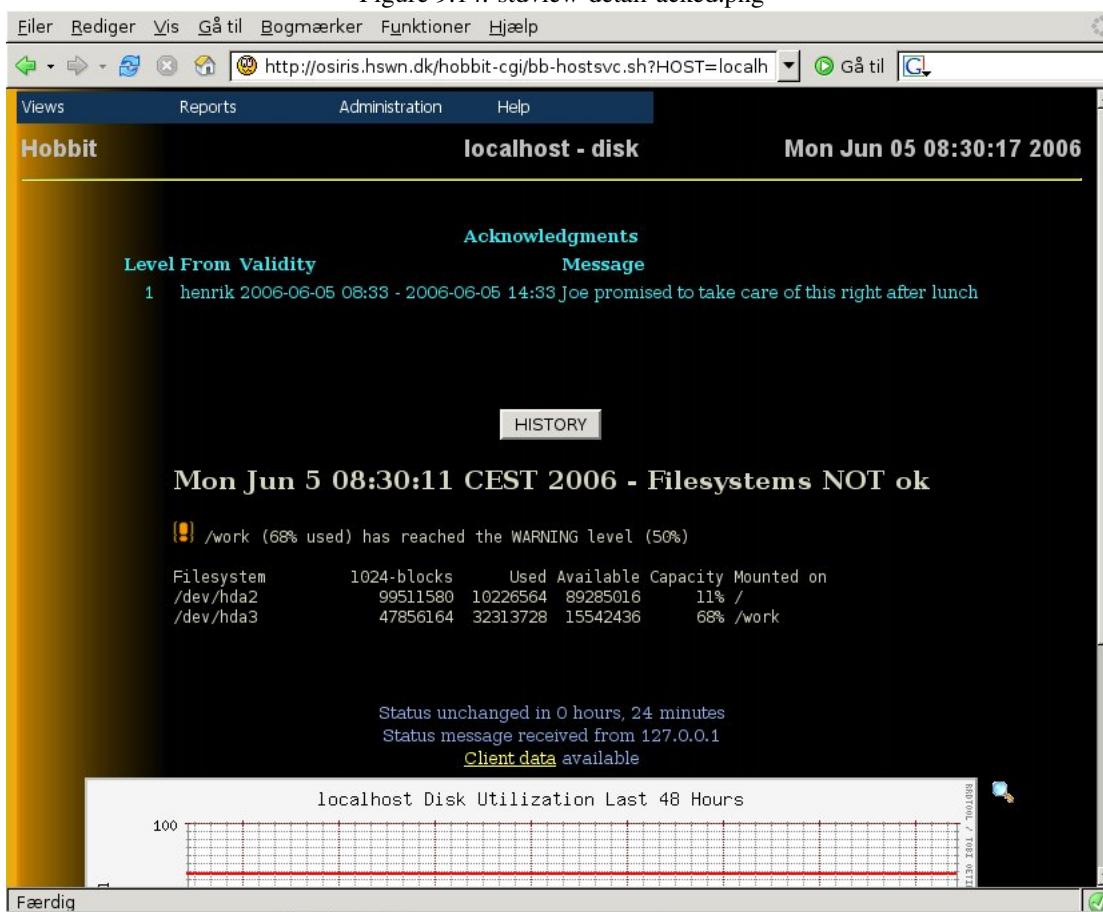


Figure 9.14: stdview-detail-acked.png



Chapter 10

Known issues in xymon

10.1 Known issues in xymon

This describes some known problems you may encounter when using xymon to monitor servers.

- How to report bugs
- JavaScript errors when using enable/disable function
- DNS error reported for network tests
- Network tests fail sporadically
- “Failed to find enough entropy” on Solaris
 - xymon fails on FreeBSD with “Could not get sem: No space left on device”
 - xymon will not run inside a FreeBSD jail
 - xymon on Solaris compiles but aborts with some “ld.so” error
 - xymon on Solaris compiles but aborts with some “ld.so” error
 - xymon on Solaris 10 “xymond status-board not available”

How to report bugs

If you think you have found a bug in xymon, please report it to the Xymon mailing list at xymon@hswn.dk. You can do a lot to help getting bugs fixed by providing detailed information about the bug:

- Always include the version number of xymon you’re using
- If one of the xymon tools crashes and leaves a core-file (usually in the `xymon/server/tmp/` directory), please use the **gdb** tool to pinpoint where the crash occurred:
 - Login as the xymon user
 - `$ cd /server`
 - `$ gdb bin/PROGRAMFILE tmp/COREFILE`
 - then at the `gdb>` prompt, execute the command
 - `gdb> bt`

Internet Explorer complains about Javascript errors in Enable/Disable

This happens for some, but works for most people. One workaround is to disable the Javascript validation code in the enable/disable function: Edit `xymon/cgi-bin/hobbit-enadis.sh` script and add the option “`-no-jsvalidation`” to the `hobbisvc.cgi` command - this disables Javascript validation on the “info” page - and edit the file `hobbit/server/web/maint_form` so you remove the text `'onClick="validateDisable(this.form)'` from the input-tag near the end of that file.

DNS error reported for network tests

The bbtest-net network tester uses the built-in ARES library for DNS lookups. There have been reports that this library fails on some systems; one confirmed case is on “OSF1 V5.1”. So if you suddenly get a lot of failed network tests that report “DNS error”, try running bbtest-net with the “`-no-ares`” option to use the standard DNS lookups instead.

Network tests fail sporadically, or report long reponsetimes

The bbtest-net network tester runs many tests in parallel; by default it will typically run up to 256 tests concurrently. This may be more than your network test server or network infrastructure can handle; if you see sporadic timeouts of network tests or the graphs show increased responsetimes, you can lower the number of concurrent tests by adding the “`-concurrency=N`” option to bbtest-net in the `/server/etc/xymonlaunch.cfg` file. This has been especially important for sites doing many http checks, since these typically have much more traffic going on while testing than simple TCP services such as smtp.

Network tests fail on Solaris with “Failed to find enough entropy”

OpenSSL uses random data to initialise a key that is negotiated when a new SSL-encrypted connection is being setup. Solaris 8 and earlier doesn’t have a standard way of getting random data, so OpenSSL cannot get all of the randomness it needs. Solaris **patch 112438** solves this by adding a `/dev/random` device that provides random data to applications. Thanks to Scott Kelley for the pointer to the Solaris patch.

Asif Iqbal notes: Patch 112438 only works for Solaris 8. For Solaris 6 and 7 you need to either install SUNWski pkg or ANDIrand pkg. See <http://www.cosy.sbg.ac.at/~andi/SUNrand/>. I have been using ANDIrand since that did not require a reboot and easily available.

xymon fails on FreeBSD with “Could not get sem: No space left on device”

xymon uses some kernel resources - semaphores and shared memory. If you get the following error message in `xymonlaunch.log` when trying to start Xymon:

```
2005-05-29 20:25:14 Setting up xymond channels
2005-05-29 20:25:14 Could not get sem: No space left on device
2005-05-29 20:25:14 Cannot setup status channel
2005-05-29 20:25:14 Task xymond terminated, status 1
```

then you need to increase the number of semaphore sets and individual semaphores available to applications. The current settings for your kernel can be found with “`sysctl kern.ipc.semmin`” (semaphore sets) and “`sysctl kern.ipc.semmaxns`” (total number of semaphores). xymon uses 6 semaphore sets, with a total of 18 semaphores.

To increase this, put these two lines in `/boot/loader.conf` on your system:

```
kern.ipc.semmin="40"
kern.ipc.semmaxns="100"
```

Adjust the values to something reasonable for your system - considering the current settings (from the sysctl output), and xymon's needs (6 sets with 18 semaphores).

More information about tuning the FreeBSD kernel parameters is available in the FreeBSD Handbook

xymon will not run inside a FreeBSD jail

FreeBSD jails have System V IPC support disabled by default. See the FreeBSD architecture Handbook section on jail restrictions. Essentially, there is a **security.jail.jail.sysvipc.allowed** sysctl that must be enabled. If it isn't then xymon will refuse to start with the error Could not get shm of size 262144: Function not implemented.

xymon on Solaris compiles but aborts with some “ld.so” error

This info was contributed by sladewig, with a few modifications:

The system loader/linker can't find your lib. Assuming you have the .so lib in /usr/local/lib, You can add -R to the Makefile PCRELIB

Note: This information only applies if you are using the Solaris linker. The GNU linker uses the “-rpath” option which is defined differently: Add

RPATH = -Wl,-rpath=

at the bottom of the top-level Makefile.

xymon on Solaris 10 “xymond status-board not available”

It seems that xymon triggers a bug in the Solaris 10 “TCP fusion” loopback interface, resulting in no data being returned from xymond when queried for the current status of all monitored systems. The recommended solution is to disable the use of TCP fusion. Colin Spargo writes:

If you disable TCP fusion be adding the following kernel parameter to /etc/system and reboot, hopefully you will find that the problem goes away.

```
set ip:do_tcp_fusion = 0
```

Apparently this can be done on a live system as well (without rebooting), but will require xymon to be restarted. To do this:

```
echo do_tcp_fusion/W0 | mdb -kw
```

More in his mail

Chapter 11

Server Configuration files

11.1 xymonserver.cfg - xymon Environment Variables

11.1.1 DESCRIPTION

xymon programs use multiple environment variables beside the normal set of variables. The environment definitions are stored in the xymon/server/etc/hobbitserver.cfg file. Each line in this file is of the form **NAME=VALUE** and defines one environment variable NAME with the value VALUE.

11.1.2 ENVIRONMENT AREAS

In some cases it may be useful to have different values for an environment variable, depending on where it is used. This is possible by defining variables with an associated “area”. Such definitions have the form **AREA/NAME=VALUE**.

E.g. to define a special setup of the BBDISP variable when it is used by an application in the “management” area, you would do this:

```
BBDISP="127.0.0.1"          # Default definition
management/BBDISP="10.1.0.5"  # Definition in the "management" area
```

Areas are invoked by using the “–area” option for all tools, or via the ENVAREA setting in the *xymonlaunch.cfg(5)* file.

11.1.3 GENERAL SETTINGS

The fully-qualified hostname of the server that is running xymon.

The hostname used to access this servers’ web-pages, used to construct URL’s in the xymon webpages. Default is the BBSERVERHOSTNAME.

The public IP-address of the server that is running xymon.

A name identifying the operating system of the xymon server. The known names are currently “linux”, “freebsd”, “solaris”, “hpx”, “aix” and “osf”.

If set to TRUE, xymon will use fully-qualified hostnames throughout. If set to FALSE, hostnames are stripped of their domain-part before being processed. It is **highly recommended** that you keep this set to TRUE. Default: TRUE.

Controls how the HTML page for a status log is generated. If set to DYNAMIC, the HTML logs are generated on-demand by the *bb-hostsvc.cgi(1)* script. If set to STATIC, you must activate the *xymond_filestore(8)* module (through an entry in the *xymonlaunch.cfg(5)* file) to create and store the HTML logs whenever a status update is received. Setting “BBLOGSTATUS=STATIC” is **discouraged** since the I/O load on the xymon server will increase significantly.

Defines the name of the column for “ping” test status. The data from the “ping” test is used internally by xymon, so it must be defined here so all of the Xymon tools know which column to watch for this data. The default setting is PINGCOLUMN=conn.

Defines the name of the column for the “info” pages.

Defines the name of the column for the RRD graph pages.

The default height (in pixels) of the RRD graph images. Default: 120 pixels.

The default width (in pixels) of the RRD graph images. Default: 576 pixels.

The graphs on the “trends” page show data for the past TRENDSECONDS seconds. Default: 172800 seconds, i.e. 48 hours.

The Content-type reported by the CGI scripts that generate web pages. By default, this is “text/html”. If you have on-line help texts in character sets other than the ISO-8859-1 (western european) character set, it may be necessary to modify this to include a character set. E.g.

HTMLCONTENTTYPE=”text/html;charset=euc-jp” for a Japanese character sets. Note: Some webservers will automatically add this, if configured to do so.

Defines the default set of holidays used if there is no “holidays” tag for a host in the bb-hosts file. Holiday sets are defined in the *xymon-holidays.cfg(5)* file. If not defined, only the default holidays (those defined outside a named holiday set) will be considered as holidays.

Defines which day is the first day of the week. Set to “0” for Sunday, “1” for Monday. Default: 1 (Monday).

11.1.4 DIRECTORIES

The top-level directory for the xymon installation. The default is the home-directory for the user running xymon.

The directory for the xymon’s own logfiles (NOT the status-logs from the monitored hosts).

The xymon server directory, where programs and configurations are kept. Default: \$BBSERVERROOT/server/ .

Directory used for temporary files. Default: \$BBHOME/tmp/

Directory for xymon webfiles. The \$BBWEB URL must map to this directory. Default: \$BBHOME/www/

Directory for xymon notes-files. The \$BBNOTESSKIN URL must map to this directory. Default: \$BBHOME/www/notes/

Directory for xymon availability reports. The \$BBREPURL URL must map to this directory. Note also that your webserver must have write-access to this directory, if you want to use the *bb-rep.cgi(1)* CGI script to generate reports on-demand. Default: \$BBHOME/www/rep/

Directory for xymon snapshots. The \$BBSNAPURL URL must map to this directory. Note also that your web-server must have write-access to this directory, if you want to use the *bb-snapshot.cgi(1)* CGI script to generate snapshots on-demand. Default: \$BBHOME/www/snap/

Directory for all data stored about the monitored items. Default: \$BBSERVERROOT/data/

Directory for storing the raw status-logs. Not used unless “xymond_filestore –status” is running, which is **discouraged** since it increases the load on the xymon server significantly. Default: \$BBVAR/logs/

Directory for storing HTML status-logs. Not used unless “xymond_filestore –status –html” is running, which is **discouraged** since it increases the load on the xymon server significantly. Default: \$BBHOME/www/html/

Directory for storing the history of monitored items. Default: \$BBVAR/hist/

Directory for storing the detailed status-log of historical events. Default: \$BBVAR/histlogs/

Directory for storing information about alerts that have been acknowledged. Default: \$BBVAR/acks/

Directory for storing information about tests that have been disabled. Default: \$BBVAR/disabled/

Directory for storing incoming “data” messages. Default: \$BBVAR/data/

Top-level directory for storing RRD files (the databases with trend-information used to generate graphs). Default: \$BBVAR/rrd/

Directory for storing the data sent by a xymon client around the time a status changes to a warning (yellow) or critical (red) state. Used by the *xymond_hostdata(8)* module. Default: \$BBVAR/hostdata/

11.1.5 SYSTEM FILES

Full path to the xymon *bb-hosts(5)* configuration file. Default: \$BBHOME/etc/bb-hosts.

Full path to the *bb(1)* client program. Default: \$BBHOME/bin/bb.

Full path to the *bbgen(1)* webpage generator program. Default: \$BBHOME/bin/bbgen.

11.1.6 URLs

The root URL for the xymon webpages, without the hostname. This URL must be mapped to the /server/www/ directory in your webserver configuration. See the sample Apache configuration in /server/etc/xymon-apache.conf.

The root URL for the xymon CGI-scripts, without the hostname. This directory must be mapped to the /cgi-bin/ directory in your webserver configuration, and must be flagged as holding executable scripts. See the sample Apache configuration in /server/etc/xymon-apache.conf.

Initial part of the xymon URL, including just the protocol and the hostname, e.g. “<http://www.foo.com>“

Prefix for all of the static xymon webpages, e.g. “<http://www.foo.com/xymon>“

URL prefix for the static HTML status-logs generated when BBLOGSTATUS=STATIC. Note that this setting is **discouraged** so this setting should not be used.

URL prefix (without hostname) of the xymon webpages. E.g. “/xymon”.

URL prefix (without hostname) of the xymon graphics. E.g. “/xymon/gifs”.

URL prefix (without hostname) of the xymon on-line help files. E.g “/xymon/help”.

URL prefix (without hostname) of the xymon menu files. E.g “/xymon/menu”.

URL prefix (without hostname) of the xymon on-line notes files. E.g “/xymon/notes”.

URL prefix (without hostname) of the xymon availability reports. E.g. “/xymon/rep”.

URL prefix (without hostname) of the xymon snapshots. E.g. “/xymon/snap”.

URL prefix (without hostname) of the xymon WAP/WML files. E.g. “/xymon/wml”.

URL prefix (without hostname) of the xymon CGI-scripts. Default: \$BBSERVERCGIURL .

Format string used to build a link to the documentation for a column heading. Default: “\$CGIBINURL/xymoncolumn.sh?%s”, which causes links to use the *xymoncolumn.sh(1)* script to document a column.

11.1.7 SETTINGS FOR SENDING MESSAGES TO HOBBIT

The IP-address used to contact the *xymond(8)* service. Used by clients and the tools that perform network tests. Default: \$BBSERVERIP

List of IP-adresses. Clients and network test tools will try to send status reports to a xymon server running on each of these adresses. This setting is only used if BBDISP=0.0.0.0.

Compatibility setting for Big Brother: List of colors that are considered “critical” and therefore will trigger an alert. Not used by xymon.

Compatibility setting for Big Brother: This is the IP-address of the server where a BBPAGER service is running. It is not used by xymon.

Compatibility setting for Big Brother: List of servers running the BBPAGER service, used if BBPAGE=0.0.0.0. It is not used by xymon.

The portnumber for used to contact the *xymond(8)* service. Used by clients and the tools that perform network tests. Default: 1984.

Compatibility setting for Big Brother. Controls whether so send combo-messages or not. Ignored by xymon.

The maximum number of status messages to combine into one combo message. You may need to lower this number if your BBDISPLAY server has trouble keeping up with the incoming status messages from bbtest-net. Default: 100.

Length of a pause introduced between each successive transmission of a combo-message by bbtest-net. You may have to increase this value to give your BBDISPLAY server time to process one combo message before the next one arrives. This number defines how many microseconds to wait between the messages. Default: 0 (send messages as quickly as possible).

11.1.8 HOBBITD SETTINGS

Comma-separated list of the colors that may trigger an alert-message. The default is “red,yellow,purple”. Note that alerts may further be generated or suppressed based on the configuration in the *xymon-alerts.cfg(5)* file.

Comma-separated list of the colors that may trigger a recovery-message. The default is “green,clear,blue”.

How often alerts get repeated while a status is in an alert state. This is the default setting, which may be changed in the *xymon-alerts.cfg(5)* file.

Controls how status messages from unknown hosts (i.e. hosts not listed in the bb-hosts file) are handled.

BBGHOSTS=1: Causes the status report to be silently discarded. This is the default behaviour in xymon.

BBGHOSTS=2: Discards the status report, but keep track of the hostname and report it on the xymond status page.

When BBGHOSTS is set to 1 or 2, the hostnames in incoming status-messages is matched without any case-sensitivity, unlike normal Big Brother which is case-sensitive in hostnames. So with BBGHOSTS set to 1 or 2, “WWW.FOO.COM” and “www.foo.com“ are considered to be the same host. If necessary, the incoming hostname will be changed to match the way it is written in the bb-hosts file, changing case as needed.

The maximum size of a “status” message in kB, default: 256. Status messages are the ones that end up as columns on the web display. The default size should be adequate in most cases, but some extension scripts can generate very large status messages - close to 1024 kB. You should only change this if you see messages in the xymond log file about status messages being truncated.

The maximum size of a “client” message in kB, default: 512. “client” messages are generated by the xymon client, and often include large process-listings. You should only change this if you see messages in the xymond log file about client messages being truncated.

The maximum size of a “data” message in kB, default: 256. “data” messages are typically used for client reports of e.g. netstat or vmstat data. You should only change this setting if you see messages in the xymond log file about data messages being truncated.

The maximum size of a “notes” message in kB, default: 256. “notes” messages provide a way for uploading documentation about a host to xymon; it is not enabled by default. If you want to upload large documents, you may need to change this setting.

The maximum size of a “status change” message in kB, default: Current value of the MAXMSG_STATUS setting. Status-change messages occur when a status changes color. There is no reason to change this setting.

The maximum size of a “page” message in kB, default: Current value of the MAXMSG_STATUS setting. “page” messages are alerts, and include the status message that triggers the alert. There is no reason to change this setting.

The maximum size of an “enadis” message in kB, default: 32. “enadis” are small messages used when enabling or disabling hosts and tests, so the default size should be adequate.

The maximum size of a “client change” message in kB, default: Current value of the MAXMSG_CLIENT setting. Client-change messages occur when a status changes color to one of the alert-colors, usually red, yellow and purple. There is no reason to change this setting.

11.1.9 HOBBITD_HISTORY SETTINGS

If set to TRUE, *xymond_history(8)* will update the \$BBHIST/allevents file logging all changes to a status. The allevents file is used by the *bb-eventlog.cgi(1)* tool to show the list of recent events on the BB2 webpage.

If set to TRUE, *xymond_history(8)* will update the host-specific eventlog that keeps record of all status changes for a host. This logfile is not used by any xymon tool.

If set to TRUE, *xymond_history(8)* will save historical detailed status-logs to the \$BBHISTLOGS directory.

11.1.10 HOBBITD_ALERT SETTINGS

Command used to send alerts via e-mail, including a “Subject:” header in the mail. Default: “mail -s”

Command used to send alerts via e-mail in a form that does not have a “Subject” in the mail. Default: “mail”

Maps status-columns to numeric service-codes. The numeric codes are used when sending an alert using a script, where the numeric code of the service is provided in the BBSVCNUM variable.

11.1.11 HOBBITD_RRD SETTINGS

List of “COLUMNNAME[=RRDSERVICE]” settings, that define which status- and data-messages have a corresponding RRD graph. You will normally not need to modify this, unless you have added a custom TCP-based test to the bb-services file, and want to collect data about the response-time, OR if you are using the *xymond_rrd(8)* external script mechanism to collect data from custom tests. Note: All TCP tests are automatically added.

This is also used by the *bb-hostsvc.cgi(1)* script to determine if the detailed status view of a test should include a graph.

List of the RRD databases, that should be shown as a graph on the “trends” column.

This is used to disable the tracking of certain filesystems. By default all filesystems reported by a client are tracked. In some cases you may want to disable this for certain filesystems, e.g. database filesystems since they are always completely full. This setting is a regular expression that is matched against the filesystem name (the Unix mount-point, or the Windows disk-letter) - if the filesystem name matches this expression, then it will not be tracked by xymon.

Note: Setting this does not affect filesystems that are already being tracked by xymon - to remove them, you must remove the RRD files for the unwanted filesystems from the *xymon/data/rrd/HOSTNAME/* directory.

This is used to enable tracking of only selected filesystems (see the NORRDDISKS setting above). By default all filesystems are being tracked, setting this changes that default so that only those filesystems that match this pattern will be tracked.

11.1.12 BBTEST-NET NETWORK TEST SETTINGS

If this variable is defined, then only the hosts that have been tagged with “NET:\$BBLOCATION” will be tested by the bbtest-net tool.

If set to TRUE, the connectivity (ping) test will be performed.

If set to TRUE, then failing network tests go CLEAR if the conn-test fails.

List of network services (separated with <space>) that should go yellow upon failure instead of red.

When using the “router” or “depends” tags for a host, a failure status will include text that an “Intermediate router is down”. With todays network topologies, the router could be a switch or another network device; if you define this environment variable the word “router” will be replaced with whatever you put into the variable. So to inform the users that an intermediate switch or router is down, use BBROUTERTEXT=“switch or router”. This can also be set on a per-host basis using the “DESCR:hosttype:description” tag in the *bb-hosts(5)* file.

When a network test fails, the status message reports “SERVICENAME not OK”. The “not OK” message can be changed via this variable, e.g. you can change it to “FAILED” or customize it as you like.

The command used to run the *xymonping(1)* tool for the connectivity test. (The name FPING is due to the fact that the “fping” utility was used until xymon version 4.2). This may include suid-root wrappers and xymonping options. Default: “xymonping”

Defines the location of the “traceroute” tool and any options needed to run it. traceroute is used by the connectivity test when the ping test fails; if requested via the “trace” tag, the TRACEROUTE command is executed to try to determine the point in the network that is causing the problem. By default the command executed is “traceroute -n -q 2 -w 2 -m 15” (no DNS lookup, max. 2 probes, wait 2 seconds per hop, max 15 hops).

If you have the *mtr(8)* tool installed - available from <http://www.bitwizard.nl/mtr/> - I strongly recommend using this instead. The recommended setting for mtr is “/usr/sbin/mtr -c 2 -n –report” (the exact path to the mtr utility may be different on your system). Note that mtr needs to be installed suid-root on most systems.

Defines the *ntpdate(1)* program used for the “ntp” test. Default: “ntpdate”

Defines the *rpcinfo(8)* program used for “rpc” tests. Default: “rpcinfo”

11.1.13 BBGEN WEBPAGE GENERATOR SETTINGS

HTML code that is inserted on all standard headers. The default is to add the text “xymon” in the upper-left corner of the page, but you can easily replace this with e.g. a company logo. If you do, I suggest that you keep it at about 30-35 pixels high, and 100-150 pixels wide.

The string “Pages hosted locally” that appears above all of the pages linked from the main xymon webpage.

The string “Subpages hosted locally” that appears above all of the sub-pages linked from pages below the main xymon webpage.

The string “Remote status display” that appears about the summary statuses displayed on the min xymon webpage.

HTML tags designed to go in a tag, to choose the font for titles of the webpages.

HTML tags designed to go in a tag, to choose the font for row headings (hostnames) on the webpages.

HTML tags designed to go in a tag, to chose the font for column headings (test names) on the webpages.

HTML tags designed to go in a tag, to chose the font for the acknowledgement text displayed on the status-log HTML page for an acknowledged status.

When displaying the detailed status of an acknowledged test, xymon will include the time that the acknowledgement expires using the print-format defined in this setting. You can define the timeformat using the controls in your systems *strftime(3)* routine, and add the text suitable for your setup.

On webpages generated by bbgen, the default header includes the current date and time. Normally this looks like “Tue Aug 24 21:59:47 2004”. The BBDATEFORMAT controls the format of this timestamp - you can define the format using the controls in the *strftime(3)* routine. E.g. to have it show up as “2004-08-24 21:59:47 +0200” you would set BBDATEFORMAT=%Y-%m-%d %H:%M:%S %z”

How holiday dates are displayed. The default is “%d/%m” which show the day and month. American users may want to change this to “%m/%d” to suit their preferred date-display style. This is a formatting string for the system *strftime(3)* routine, so any controls available for this routine may be used.

Inspired by Jeff Stoner’s col_repeat_patch.tgz patch, this defines the maximum number of rows before repeating the column headings on a webpage. This sets the default value for the *bbgen(1)* “–maxrows” option; if the command-line option is also specified, then it overrides this environment variable. Note that unlike Jeff’s patch, bbgen implements this for both the bb2.html page and all other pages (bb.html, subpages, bbnk.html).

If set to TRUE, then summaries will affect the color of the main xymon webpage. Default: FALSE.

The height (in pixels) of the icons showing the color of a status. Default: 16, which matches the default icons.

The width (in pixels) of the icons showing the color of a status. Default: 16, which matches the default icons.

List of the status logs fed by data from the xymon client. These status logs will - if there are Xymon client data available for the host - include a link to the raw data sent by the client. Default: cpu,disk,memory,procs,svcs.

If defined, this is the title of the RSS/RDF documents generated when *bbgen(1)* is invoked with the “–rss” option. The default value is “xymon Alerts”.

Maximum size of a WAP/WML output “card” when generating these. Default: 1500.

List of scripts to run as extensions to the BB2 page. Note that two scripts, “eventlog.sh” and “acklog.sh” are handled specially: They are handled internally by bbgen, but the script names must be listed in this variable for this function to be enabled.

List of scripts to run as extensions to a history page.

Default threshold for listing the availability as “critical” (red) when generating the availability report. This can be set on a per-host basis with the WARNPCT setting in *bb-hosts(5)*.

Default: 97 (percent)

Default bbgen options used for reports. This will typically include such options as “–subpagecolumns”, and also “–ignorecolumns” if you wish to exclude certain tests from reports by default.

Default bbgen options used by snapshots. This should be identical to the options you normally used when building xymon webpages.

11.1.14 FILES

xymon/server/etc/hobbitserver.cfg

11.1.15 SEE ALSO

xymon(7)

11.2 bb-hosts - Main xymon configuration file

11.2.1 SYNOPSIS

11.2.2 DESCRIPTION

The *bb-hosts(5)* file is the most important configuration file for all of the xymon programs. This file contains the full list of all the systems monitored by xymon, including the set of tests and other configuration items stored for each host.

11.2.3 FILE FORMAT

Each line of the file defines a host. Blank lines and lines starting with a hash mark (#) are treated as comments and ignored. Long lines can be broken up by putting a backslash at the end of the line and continuing the entry on the next line.

The format of an entry in the bb-hosts file is as follows:

IP-addresshostname#tag1tag2...

The IP-address and hostname are mandatory; all of the tags are optional. Listing a host with only IP-address and hostname will cause a network test to be executed for the host - the connectivity test is enabled by default, but no other tests.

The optional tags are then used to define which tests are relevant for the host, and also to set e.g. the time-interval used for availability reporting by *bbgen(1)*

An example of setting up the bb-hosts file is in the xymon on-line documentation (from the Help menu, choose “Configuring Monitoring”). The following describes the possible settings in a bb-hosts file supported by xymon.

11.2.4 TAGS RECOGNIZED BY ALL TOOLS

This tag is used to include another file into the bb-hosts file at run-time, allowing for a large bb-hosts file to be split up into more manageable pieces.

The “filename” argument should point to a file that uses the same syntax as bb-hosts. The filename can be an absolute filename (if it begins with a ‘/’), or a relative filename - relative filenames are prefixed with the directory where the main bb-hosts file is located (usually \$BBHOME/etc/).

You can nest include tags, i.e. a file that is included from the main bb-hosts file can itself include other files.

Acts like the “include” tag, but only on the BBDISPLAY server. Can be used e.g. to put a group of hosts on multiple subpages, without having to repeat the host definitions.

Acts like the “include” tag, but only on the BBNET server.

11.2.5 GENERAL PER-HOST OPTIONS

Controls whether stale status messages go purple or clear when a host is down. Normally, when a host is down the client statuses (“cpu”, “disk”, “memory” etc) will stop updating - this would usually make them go “purple” which can trigger alerts. To avoid that, xymon checks if the “conn” test has failed, and if that is true then the other tests will go “clear” instead of purple so you only get alerts for the “conn” test. If you do want the stale statuses to go purple, you can use the “noclear” tag to override this behaviour.

Note that “noclear” also affects the behaviour of network tests; see below.

When a single host is defined multiple time in the bb-hosts file, bbgen tries to guess which definition is the best to use for the information used on the “info” column, or for the NOPROPRED and other bbgen-specific settings. Host definitions that have a “noconn” tag or an IP of 0.0.0.0 get lower priority.

By using the “prefer” tag you tell bbgen that this host definition should be used.

Note: This only applies to hosts that are defined multiple times in the bb-hosts file, although it will not hurt to add it on other hosts as well.

11.2.6 BBGEN DISPLAY OPTIONS

These tags are processed by the *bbgen(1)* tool when generating the xymon webpages or reports.

page NAME [Page-title] This defines a page at the level below the entry page. All hosts following the “page” directive appear on this page, until a new “page”, “subpage” or “subparent” line is found.

subpage NAME [Page-title] This defines a subpage in the second level below the entry page. You must have a previous “page” line to hook this subpage to.

subparent parentpage newpage [Page-title] This is used to define subpages in whatever levels you may wish. Just like the standard “subpage” tag, “subparent” defines a new xymon webpage; however with “subparent” you explicitly list which page it should go as a subpage to. You can pick any page as the parent - pages, subpages or even other subparent pages. So this allows you to define any tree structure of pages that you like.

E.g. with this in bb-hosts:

```
pageUSAUnitedStates subpageNYNewYork subparentNYmanhattanManhattandatacenters subparentmanhattanwallstreetWallStreetcenter
```

you get this hierarchy of pages:

```
USA(UnitedStates) NY(NewYork) manhattan(Manhattandatacenters) wallstreet(WallStreetcenter)
```

Note: The parent page must be defined before you define the subparent. If not, the page will not be generated, and you get a message in the log file.

Note: bbgen is case-sensitive, when trying to match the name of the parent page.

The inspiration for this came from Craig Cook's mkbb.pl script, and I am grateful to Craig for suggesting that I implement it in bbgen. The idea to explicitly list the parent page in the "subparent" tag was what made it easy to implement.

group [group-title]

group-compress [group-title] Defines a group of hosts, that appear together on the webpage, with a single header-line listing all of the columns. Hosts following the "group" line appear inside the group, until a new "group" or page-line is found. The two group-directives are handled identically by xymon and bbgen, but both forms are allowed for backwards compatibility.

group-sorted [group-title] Same as the "group" line, but will sort the hosts inside the group so they appear in strict lexicographic order.

group-only COLUMN1—COLUMN2—COLUMN3 [group-title] Same as the "group" and "group-compress" lines, but includes only the columns explicitly listed in the group. Any columns not listed will be ignored for these hosts.

group-except COLUMN1—COLUMN2—COLUMN3 [group-title] Same as the "group-only" lines, but includes all columns EXCEPT those explicitly listed in the group. Any columns listed will be ignored for these hosts - all other columns are shown.

title Page, group or host title text The "title" tag is used to put custom headings into the pages generated by bbgen, in front of page/subpage links, groups or hosts.

The title tag operates on the next item in the bb-hosts file following the title tag.

If a title tag precedes a host entry, the title is shown just before the host is listed on the status page. The column headings present for the host will be repeated just after the heading.

If a title tag precedes a group entry, the title is show just before the group on the status page.

If a title tag precedes a page/subpage/subparent entry, the title text replaces the normal "Pages hosted locally" heading normally inserted by xymon. This appears on the page that links to the subpages, not on the subpage itself. To get a custom heading on the subpage, you may want to use the "-pagetext-heading" when running *bbgen(1)*

NAME:hostname Overrides the default hostname used on the overview web pages. If "hostname" contains spaces, it must be enclosed in double quotes, e.g. NAME:"R&D Oracle Server"

CLIENT:hostname Defines an alias for a host, which will be used when identifying status messages. This is typically used to accomodate a local client that sends in status reports with a different hostname, e.g. if you use hostnames with domains in your xymon configuration, but the client is a silly Window box that does not include the hostname. Or vice versa. Whatever the reason, this can be used to match status reports with the hosts you define in your bb-hosts file. It causes incoming status reports with the specified hostname to be filed using the hostname defined in bb-hosts.

NOCOLUMNS:column[,column] Used to drop certain of the status columns generated by the xymon client. **column**

is one of **cpu , disk , files , memory , msgs , ports , procs** . This setting stops these columns from being updated for the host. Note: If the columns already exist, you must use the *bb(1)* utility to **drop** them, or they will go purple.

COMMENT:Host comment Adds a small text after the hostname on the webpage. This can be used to describe the host, without completely changing its display-name as the NAME: tag does. If the comment includes whitespace, it must be in double-quotes, e.g. COMMENT:"Sun webserver"

DESCR:Hosttype:Description Define some informational text about the host. The “Hosttype” is a text describing the type of this device - “router”, “switch”, “hub”, “server” etc. The “Description” is an informational text that will be shown on the “Info” column page; this can e.g. be used to store information about the physical location of the device, contact persons etc. If the text contain whitespace, you must enclose it in double-quotes, e.g. DESCRIPTOR：“switch:4th floor Marketing switch”

CLASS:Classname Force the host to belong to a specific class. Class-names are used when configuring log-file monitoring (they can be used as references in *client-local.cfg(5)* and *xymon-clients.cfg(5)* to group logfile checks). Normally, class-names are controlled on the client by starting the xymon client with the “–class=Classname” option. If you specify it in the bb-hosts file on the xymon server, it overrides any classname that the client reports.

The keyword “dialup” for a host means that it is OK for it to be off-line - this should not trigger an alert. All network tests will go “clear” upon failure, and any missing reports from e.g. cpu- and disk-status will not go purple when they are not updated.

Ignore this host on the BB2 page. Even if it has an active alert, it will not be included in the BB2 page. This also removes the host from the event-log display.

Ignore this host completely when generating the xymon webpages. Can be useful for monitoring a host without having it show up on the webpages, e.g. because it is not yet in production use. Or for hiding a host that is shown only on a second pageset.

TRENDS:[*, [![graph,...]]] Defines the RRD graphs to include in the “trends” column generated by bbgen. This option syntax is complex.

If this option is not present, bbgen provides graphs matching the standard set of RRD files: la, disk, memory, users, vmstat, iostat, netstat, tcp, bind, apache, sendmail

If this option is specified, the list of graphs to include start out as being empty (no graphs). To include all default graphs, use an asterisk. E.g. “TRENDS:*” To exclude a certain graph, specify it prefixed with ‘!’. E.g. to see all graphs except users: “TRENDS:*,!users”

The netstat, vmstat and tcp graphs have many “subgraphs”. Which of these are shown can be specified like this: “TRENDS:*,netstat:netstat2—netstat3,tcp:http—smtp—conn” This will show all graphs, but instead of the normal netstat graph, there will be two: The netstat2 and netstat3 graphs.

Instead of the combined tcp graphs showing all services, there will be three: One for each of the http, conn and smtp services.

DOWNTIME=[columns: day:starttime:endtime:cause[,day:starttime:endtime:cause]] This tag can be used to ignore failed checks during specific times of the day - e.g. if you run services that are only monitored e.g. Mon-Fri 8am-5pm, or you always reboot a server every Monday between 5 and 6 pm.

The timespecification consists of

day: W means Mon-Fri (“weekdays”), * means all days, **0 .. 6** = Sunday .. Saturday. Listing multiple days is possible, e.g. “60” is valid meaning “Saturday and Sunday”.

starttime: Time to start showing errors, must be in 24-hour clock format as HHMM hours/minutes. E.g. for 8 am enter “0800”, for 9.30 pm enter “2130”

endtime:

Time to stop showing errors.

If necessary, multiple periods can be specified. E.g. to monitor a site 24x7, except between noon and 1 pm, use DOWNTIME=*:0000:1159,*:1300:2359

The interval between starttime and endtime may cross midnight, e.g. ***:2330:0200** would be valid and have the same effect as ***:2330:2400,*:0000:0200**.

What happens is that if a test fails during the specified time, it is reported with status BLUE instead of yellow or red. Thus you can still see when the service was unavailable, but alarms will not be triggered and the downtime is not counted in the availability calculations generated by the xmon reports.

The “columns” setting is optional - it may be a comma-separated list of status columns in which case the DOWNTIME setting only applies to these columns.

The “cause” string (optional) is a text that will be displayed on the status web page to explain thy the system is down.

SLA=day:starttime:endtime[,day:starttime:endtime] This tag is now deprecated. Use the DOWNTIME tag instead.

This tag works the opposite of the DOWNTIME tag - you use it to specify the periods of the day that the service should be green. Failures OUTSIDE the SLA interval are reported as blue.

holidays=HOLIDAYSETNAME Define which set of holidays to use for a host. Holiday sets are defined in the *xmon-holidays.cfg(5)* configuration file. If omitted, the set given in the HOLIDAYS setting in the *xmon-server.cfg(5)* file is used.

11.2.7 HOBBIT TAGS FOR THE CRITICAL SYSTEMS OVERVIEW PAGE

NOTE: The “NK” set of tags is deprecated. They will be supported for xmon 4.x, but will be dropped in version 5. It is recommended that you move your critical systems view to the *xmon-nkview.cgi(1)* viewer, which has a separate configuration tool, *xmon-nkedit.cgi(1)* with more facilities than the NK tags in bb-hosts.

bbgen will create three sets of pages: The main page bb.html, the all-non-green-statuses page (bb2.html), and a specially reduced version of bb2.html with only selected tests (bbnk.html). This page includes selected tests that currently have a red or yellow status.

NK:testname[,testname] Define the tests that you want included on the bbnk page. E.g. if you have a host where you only want to see the http tests on bbnk.html, you specify it as

12.34.56.78www.acme.com#<http://www.acme.com>/NK:http

If you want multiple tests for a host to show up on the bbnk.html page, specify all the tests separated by commas. The test names correspond to the column names (e.g. https tests are covered by an “NK:http” tag).

NKTIME=day:starttime:endtime[,day:starttime:endtime] This tag limits the time when an active alert is presented on the NK webpage.

By default, tests with a red or yellow status that are listed in the “NK:testname” tag will appear on the NK page. However, you may not want the test to be shown outside of normal working hours - if, for example, the host is not being serviced during week-ends.

You can then use the NKTIME tag to define the time periods where the alert will show up on the NK page. The NKTIME syntax is identical to the one used for DOWNTIME specification.

11.2.8 HOBBIT TAGS FOR THE WML (WAP) CARDS

If bbgen is run with the “–wml” option, it will generate a set of WAP-format output “cards” that can be viewed with a WAP-capable device, e.g. a PDA or cell-phone.

WML:[+--- testname[,[+---]testname]] This tag determines which tests for this hosts are included in the WML (WAP) page. Syntax is identical to the NK: tag.

The default set of WML tests are taken from the –wml commandline option. If no “WML:” tag is specified, the “NK:” tag is used if present.

11.2.9 HOBBIT STATUS PROPAGATION OPTIONS

These tags affect how a status propagates upwards from a single test to the page and higher. This can also be done with the command-line options –nopropyellow and –nopropred, but the tags apply to individual hosts, whereas the command line options are global.

NOPROPRED:[+— testname[,[+—]testname]] This tag is used to inhibit a yellow or red status from propagating upwards - i.e. from a test status color to the (sub)page status color, and further on to bb.html or bb2.html

If a host-specific tag begins with a ‘-’ or a ‘+’, the host-specific tags are removed/added to the default setting from the command-line option. If the host-specific tag does not begin with a ‘+’ or a ‘-’, the default setting is ignored for this host and the NOPROPRED applies to the tests given with this tag.

E.g.: bbgen runs with “–nopropred=ftp,smtp”. “NOPROPRED:+dns,-smtp” gives a NOPROPRED setting of “ftp,dns” (dns is added to the default, ftp is removed). “NOPROPRED:dns” gives a setting of “dns” only (the default is ignored).

Note: If you set use the “–nopropred=*” commandline option to disable propagation of all alerts, you cannot use the “+” and “-” methods to add or remove from the wildcard setting. In that case, do not use the “+” or “-” setting, but simply list the required tests that you want to keep from propagating.

NOPROPYELLOW:[+— testname[,[+—]testname]] Similar to NOPROPRED: tag, but applies to propagating a yellow status upwards.

NOPROPPURPLE:[+— testname[,[+—]testname]] Similar to NOPROPRED: tag, but applies to propagating a purple status upwards.

NOPROPACK:[+— testname[,[+—]testname]] Similar to NOPROPRED: tag, but applies to propagating an acknowledged status upwards.

11.2.10 HOBBIT AVAILABILITY REPORT OPTIONS

These options affect the way the xymon availability reports are processed (see *bb-rep.cgi(1)* for details about availability reports).

REPORTTIME=day:starttime:endtime[,day:starttime:endtime] This tag defines the time interval where you measure uptime of a service for reporting purposes.

When bbgen generates a report, it computes the availability of each service - i.e. the percentage of time that the service is reported as available (meaning: not red).

By default, this calculation is done on a 24x7 basis, so no matter when an outage occurs, it counts as downtime.

The REPORTTIME tag allows you to specify a period of time other than 24x7 for the service availability calculation. If you have systems where you only guarantee availability from e.g. 7 AM to 8 PM on weekdays, you can use

REPORTTIME=W:0700:2000 and the availability calculation will only be performed for the service with measurements from this time interval.

The syntax for REPORTTIME is the same as the one used by the DOWNTIME parameter.

When REPORTTIME is specified, the availability calculation happens like this:

* Only measurements done during the given time period is used for the calculation. * “blue” time reduces the length of the report interval, so if you are generating a report for a 10-hour period and there are 20 minutes of “blue” time, then the availability calculation will consider the reporting period to be 580 minutes (10 hours minus 20 minutes). This allows you to have scheduled downtime during the REPORTTIME interval without hurting your availability; this is (I believe) the whole idea of the downtime being “planned”.

* “red” and “clear” status counts as downtime; “yellow” and “green” count as uptime. “purple” time is ignored.

The availability calculation correctly handles status changes that cross into/out of a REPORTTIME interval.
If no REPORTTIME is given, the standard 24x7 calculation is used.

WARNPCT:percentage BB's reporting facility uses a computed availability threshold to color services green (100% available), yellow (above threshold, but less than 100%), or red (below threshold) in the reports.

This option allows you to set the threshold value on a host-by-host basis, instead of using a global setting for all hosts. The threshold is defined as the percentage of the time that the host must be available, e.g. "WARN-PCT:98.5" if you want the threshold to be at 98.5%

11.2.11 NETWORK TEST SETTINGS

By default, xymon will perform a name lookup of the hostname to get the IP address it will use for network tests.
This tag causes xymon to use the IP listed in the bb-hosts file.

NET:location This tag defines the host as being tested from a specific location. If bbtest-net sees that the environment variable BBLOCATION is set, it will only test the hosts that have a matching "NET:location" tag in the bb-hosts file. So this tag is useful if you have more than one BBNET system, but you still want to keep a consolidated bb-hosts file for all your systems.

Note: The "-test-untagged" option modifies this behaviour, see *bbtest-net(1)*

Some network tests depend on others. E.g. if the host does not respond to ping, then there's a good chance that the entire host is down and all network tests will fail. Or if the http server is down, then any web content checks are also likely to fail. To avoid floods of alerts, the default behaviour is for bbtest-net to change the status of these tests that fail because of another problem to "clear" instead of "red". The "noclear" tag disables this behaviour and causes all failing tests to be reported with their true color.

This behaviour can also be implemented on a per-test basis by putting the " " flag on any network test.

Note that "noclear" also affects whether stale status messages from e.g. a client on the host go purple or clear when the host is down; see the "noclear" description in the "GENERAL PER-HOST OPTIONS" section above.

Disables the standard check of any SSL certificates for this host. By default, if an SSL-enabled service is tested, a second test result is generated with information about the SSL certificate - this tag disables the SSL certificate checks for the host.

ssldays=WARNDAYS:ALARMDAYS Define the number of days before an SSL certificate expires, in which the sslcert status shows a warning (yellow) or alarm (red) status. These default to the values from the "-sslwarndays" and "-sslalarmdays" options for the *bbtest-net(1)* tool; the values specified in the "ssldays" tag overrides the default.

depends=(testA:host1/test1,host2/test2),(testB:host3/test3),[...] This tag allows you to define dependencies between tests. If "testA" for the current host depends on "test1" for host "host1" and test "test2" for "host2", this can be defined with

`depends=(testA:host1/test1,host2/test2)`

When deciding the color to report for testA, if either host1/test1 failed or host2/test2 failed, if testA has failed also then the color of testA will be "clear" instead of red or yellow.

Since all tests are actually run before the dependencies are evaluated, you can use any host/test in the dependency - regardless of the actual sequence that the hosts are listed, or the tests run. It is also valid to use tests from the same host that the dependency is for. E.g.

`1.2.3.4foo#http://foo/ webmindepends=(webmin:foo/http)`

is valid; if both the http and the webmin tests fail, then webmin will be reported as clear.

Note: The "depends" tag is evaluated on the BBNET server while running the network tests. It can therefore only refer to other network tests that are handled by the same BBNET server - there is currently no way to

use the e.g. the status of locally run tests (disk, cpu, msgs) or network tests from other BBNET servers in a dependency definition. Such dependencies are silently ignored.

badTEST[-weekdaysstarttime-endtime :x:y:z] Normally when a network test fails, the status changes to red immediately. With a “badTEST:x:y:z” tag this behaviour changes:

* While “z” or more successive tests fail, the column goes RED. * While “y” or more successive tests fail, but fewer than “z”, the column goes YELLOW. * While “x” or more successive tests fail, but fewer than “y”, the column goes CLEAR. * While fewer than “x” successive tests fail, the column stays GREEN.

The optional timespecification can be used to limit this “badTEST” setting to a particular time of day, e.g. to require a longer period of downtime before raising an alarm during out-of-office hours. The time-specification uses:

* Weekdays: The weekdays this badTEST tag applies, from 0 (Sunday) through 6 (Saturday). Putting “W” here counts as “12345”, i.e. all working days. Putting “*” here counts as all days of the week, equivalent to “0123456”.

* starttime and endtime are specified using 24-hour clocks, e.g. “badTEST-W-0900-2000” is valid for working days between 9 AM (09:00) and 8 PM (20:00).

When using multiple badTEST tags, the LAST one specified with a matching time-spec is used.

Note: The “TEST” is replaced by the name of the test, e.g.

12.34.56.78www.foo.com#<http://www.foo.com>/badhttp:1:2:4

defines a http test that goes “clear” after the first failure, “yellow” after two successive failures, and “red” after four successive failures.

For the other network tests, use “badftp”, “badssh” etc.

11.2.12 CONNECTIVITY (PING) TEST

These tags affect the behaviour of the bbtest-net connectivity test.

Disables the ping-test, but will keep the “conn” column on the web display with a notice that it has been disabled.

Disables the ping-test, and does not put a “conn” column on the web display.

The “conn” test (which does a ping of the host) is enabled for all hosts by default, and normally you just want to disable it using “noconn” or “noping”. However, on the rare occasion where you may want to check that a host is NOT up, you can specify it as an explicit test, and use the normal test modifiers, e.g. “!conn” will be green when the host is NOT up, and red if it does appear on the network.

The actual name of the tag - “conn” by default - depends on the “-ping=TESTNAME” option for bbtest-net, as that decides the testname for the connectivity test.

conn=best,—worst, IP1[,IP2...] This adds additional IP-adresses that are pinged during the normal “conn” test. So the normal “conn” test must be enabled (the default) before this tag has any effect. The IP-adresses listed here are pinged in addition to the main IP-address.

When multiple IP’s are pinged, you can choose if ALL IP’s must respond (the “worst” method), or AT LEAST one IP must respond (the “best” setting). All of the IP’s are reported in a single “conn” status, whose color is determined from the result of pinging the IP’s and the best/worst setting. The default method is “best” - so it will report green if just one of the IP’s respond to ping.

badconn[-weekdaysstarttime-endtime :x:y:z] This is taken directly from the “fping.sh” connectivity- testing script, and is used by bbtest-net when it runs with ping testing enabled (the default). See the description of the “badTEST” tag.

route:router1,router2,... This tag is taken from the “fping.sh” script, and is used by bbtest-net when run with the “–ping” option to enable ping testing.

The router1,router2,... is a comma-separated list of hosts elsewhere in the bb-hosts file. You cannot have any spaces in the list - separate hosts with commas.

This tag changes the color reported for a ping check that fails, when one or more of the hosts in the “route” list is also down. A “red” status becomes “yellow” - other colors are unchanged. The status message will include information about the hosts in the router-list that are down, to aid tracking down which router is the root cause of the problem.

Note: Internally, the ping test will still be handled as “failed”, and therefore any other tests run for this host will report a status of “clear”.

route LOCATION:router1,router2,... If the BBLOCATION environment variable is defined, a tag of “route_BBLOCATION:” is recognized by bbtest-net with the same effect as the normal “route:” tag (see above). This allows you to have different route: tags for each BBNET server. The actual text for the tag then must match the value you have for the BBLOCATION setting. E.g. with BBLOCATION=dmz, the tag becomes “route_dmz:”

If the connectivity test fails, run a “traceroute” and include the output from this in the status message from the failed connectivity test. Note: For this to work, you may have to define the TRACEROUTE environment variable, see *xmonserver.cfg(5)*

Similar to the “trace” option, this disables the running of a traceroute for the host after a failed connectivity test. It is only used if running traceroute is made the default via the –trace option.

11.2.13 SIMPLE NETWORK TESTS

These tests perform a simple network test of a service by connecting to the port and possibly checking that a banner is shown by the server.

How these tests operate are configured in the *bb-services(5)* configuration file, which controls which port to use for the service, whether to send any data to the service, whether to check for a response from the service etc.

You can modify the behaviour of these tests on a per-test basis by adding one or more modifiers to the test: **:NUMBER** changes the port number from the default to the one you specify for this test. E.g. to test ssh running on port 8022, specify the test as **ssh:8022**.

:s makes the test silent, i.e. it does not send any data to the service. E.g. to do a silent test of an smtp server, enter **smtp:s**.

You can combine these two: **ftp:8021:s** is valid.

If you must test a service from a multi-homed host (i.e. using a specific source IP-address instead of the one your operating system provides), you can use the modifier “@IPADDRESS” at the end of the test specification, **after** any other modifiers or port number. “IPADDRESS” must be a valid dotted IP-address (not hostname) which is assigned to the host running the network tests.

The name of the test also determines the columnname that the test result will appear with in the xmon webpages.

By prefixing a test with “!” it becomes a reverse test: xmon will expect the service NOT to be available, and send a green status if it does NOT respond. If a connection to the service succeeds, the status will go red.

By prefixing a test with “?” errors will be reported with a “clear” status instead of red. This is known as a test for a “dialup” service, and allows you to run tests of hosts that are not always online, without getting alarms while they are off-line.

ftp ssh telnet smtp pop3 imap nntp rsync clamd oratns qmtp qmqp These tags are for testing services offering the FTP, Secure Shell (ssh), SMTP, POP3, IMAP, NNTP, rsync, CLAM antivirus daemon (clamd), Oracle TNS listener (oratns), qmail QMTP and QMQP protocols.

ftps telnets smtps pop3s imaps nntps These tags are for testing of the SSL-tunneled versions of the standard ftp, telnet, smtp, pop3, imap and nntp protocols. If xmon was configured with support for SSL, you can test these services like any other network service - bbtest-net will setup an SSL-encrypted session while testing the

service. The server certificate is validated and information about it sent in the “sslcert” column. Note that smtps does not have a standard portnumber assignment, so you will need to enter this into the bb-services file or your /etc/services file.

Test that a Big Brother compatible daemon is running. This check works both for the xymon *xymond(8)* daemon, and the original Big Brother bbd daemon.

11.2.14 DNS SERVER TESTS

These tags are used to setup monitoring of DNS servers.

Simple DNS test. It will attempt to lookup the A record for the hostname of the DNS server.

This is an alias for the “dns” test. In bbtest-net, the “dns” and “dig” tests are handled identically, so all of the facilities for testing described for the “dns” test are also available for the “dig” test.

dns=hostname

dns=TYPE:lookup[,TYPE:lookup...] The default DNS tests will attempt a DNS lookup of the DNS’ servers own hostname. You can specify the hostname to lookup on a DNS server by listing it on each test.

The second form of the test allows you to perform multiple queries of the DNS server, requesting different types of DNS records. The TYPE defines the type of DNS data: A (IP-address), MX (Mail eXchanger), PTR (reverse), CNAME (alias), SOA (Start-Of-Authority), NS (Name Server) are among the more common ones used. The “lookup” is the query. E.g. to lookup the MX records for the “foo.com” domain, you would use “dns=mx:foo.com”. Or to lookup the nameservers for the “bar.org” domain, “dns=ns:bar.org”. You can list multiple lookups, separated by commas. For the test to end up with a green status, all lookups must succeed.

11.2.15 OTHER NETWORK TESTS

Check for a running NTP (Network Time Protocol) server on this host. This test uses the “ntpdate” utility to check for a NTP server - you should either have ntpdate in your PATH, or set the location of the ntpdate program in \$BBHOME/etc/bbsys.local

rpc[=rpcservice1,rpcservice2,...] Check for one or more available RPC services. This check is indirect in that it only queries the RPC Portmapper on the host, not the actual service.

If only “rpc” is given, the test only verifies that the portmapper is available on the remote host. If you want to check that one or more RPC services are registered with the portmapper, list the names of the desired RPC services after the equals-sign. E.g. for a working NFS server the “mount”, “nlockmgr” and “nfs” services must be available; this can be checked with “rpc=mount,nlockmgr,nfs”.

This test uses the rpcinfo tool for the actual test; if this tool is not available in the PATH of bbtest-net, you must define the RPCINFO environment variable to point at this tool. See *xymonserver:cfg(5)*

11.2.16 HTTP TESTS

Simple testing of a http URL is done simply by putting the URL into the bb-hosts file. Note that this only applies to URLs that begin with “http:” or “https:”.

The following items describe more advanced forms of http URL’s.

Basic Authentication with username/password If the URL requires authentication in the form of a username and password, it is most likely using the HTTP “Basic” authentication. bbtest-net support this, and you can provide the username and password either by embedding them in the URL e.g.

<http://USERNAME:PASSWORD@www.sample.com/> or by putting the username and password into the /.netrc file (see *ftp(1)* for details).

Authentication with SSL client certificates An SSL client certificate can be used for authentication. To use this, the client certificate must be stored in a PEM-formatted file together with the client certificate key, in the \$BBHOME/certs/ directory. The URL is then given as

`http://CERT:FILENAME@www.sample.com/` The “CERT:” part is literal - i.e. you write C-E-R-T-colon and then the filename of the PEM-formatted certificate.

A PEM-formatted certificate file can be generated based on certificates stored in Microsoft Internet Explorer and OpenSSL. Do as follows:

From the MSIE Tools-Options menu, pick the Content tab, click on Certificates, choose the Personal tab, select the certificate and click Export. Make sure you export the private key also. In the Export File Format, choose PKCS 12 (.PFX), check the “Include all certificates” checkbox and uncheck the “Enable strong protection”. Provide a temporary password for the exported file, and select a filename for the PFX-file.

Now run “openssl pkcs12 -in file.pfx -out file.pem”. When prompted for the “Import Password”, provide the temporary password you gave when exporting the certificate. Then provide a “PEM pass phrase” (twice) when prompted for one.

The file.pem file is the one you should use in the FILENAME field in the URL - this file must be kept in \$BBHOME/certs/. The PEM pass phrase must be put into a file named the same as the certificate, but with extension “.pass”. E.g. if you have the PEM certificate in \$BBHOME/certs/client.pem, you must put the pass phrase into the \$BBHOME/certs/client.pass file. Make sure to protect this file with Unix permissions, so that only the user running xymon can read it.

Forcing an HTTP or SSL version Some SSL sites will only allow you to connect, if you use specific “dialects” of HTTP or SSL. Normally this is auto-negotiated, but experience shows that this fails on some systems.

bbtest-net can be told to use specific dialects, by adding one or more “dialect names” to the URL scheme, i.e. the “http” or “https” in the URL:

* “2”, e.g. https2://www.sample.com/ : use only SSLv2 * “3”, e.g. https3://www.sample.com/ : use only SSLv3
* “m”, e.g. httpsm://www.sample.com/ : use only 128-bit ciphers * “h”, e.g. httpsh://www.sample.com/ : use only >128-bit ciphers * “10”, e.g. http10://www.sample.com/ : use HTTP 1.0 * “11”, e.g. http11://www.sample.com/ : use HTTP 1.1

These can be combined where it makes sense, e.g to force SSLv2 and HTTP 1.0 you would use “https210”.

Testing sites by IP-address bbtest-net ignores the “testip” tag normally used to force a test to use the IP-address from the bb-hosts file instead of the hostname, when it performs http and https tests.

The reason for this is that it interacts badly with virtual hosts, especially if these are IP-based as is common with https-websites.

Instead the IP-address to connect to can be overridden by specifying it as:

`http://www.sample.com=1.2.3.4/index.html`

The “=1.2.3.4” will cause bbtest-net to run the test against the IP-address “1.2.3.4”, but still trying to access a virtual website with the name “www.sample.com“.

The “=ip.address.of.host” must be the last part of the hostname, so if you need to combine this with e.g. an explicit portnumber, it should be done as

`http://www.sample.com:3128=1.2.3.4/index.html`

HTTP Testing via proxy bbtest-net supports the Big Brother syntax for specifying an HTTP proxy to use when performing http tests. This syntax just joins the proxy- and the target-URL into one, e.g.

`http://webproxy.sample.com:3128/http://www.foo.com/` would be the syntax for testing the www.foo.com website via the proxy running on “webproxy.sample.com” port 3128.

If the proxy portnumber is not specified, the default HTTP portnumber (80) is used.

If your proxy requires authentication, you can specify the username and password inside the proxy-part of the URL, e.g.

`http://fred:Wilma1@webproxy.sample.com:3128/http://www.foo.com/` will authenticate to the proxy using a username of “fred” and a password of “Wilma1”, before requesting the proxy to fetch the www.foo.com homepage.

Note that it is not possible to test https-sites via a proxy, nor is it possible to use https for connecting to the proxy itself.

cont[=COLUMN ;URL;[expected_data_regex—#digesttype:digest]] This tag is used to specify a http/https check, where it is also checked that specific content is present in the server response.

If the URL itself includes a semi-colon, this must be escaped as ‘%3B’ to avoid confusion over which semicolon is part of the URL, and which semicolon acts as a delimiter.

The data that must be returned can be specified either as a regular expression (except that <space> is not allowed) or as a message digest (typically using an MD5 sum or SHA-1 hash).

The regex is pre-processed for backslash “\” escape sequences. So you can really put any character in this string by escaping it first:

```
\nNewline(LF,ASCII10decimal)  
\rCarriagereturn(CR,ASCII13decimal)  
\tTAB(ASCII8decimal)  
\\\Backslash(ASCII92decimal)  
\XXThecharacterwithASCIIhex-valueXX
```

If you must have whitespace in the regex, use the `[[:space:]]` syntax, e.g. if you want to test for the string “All is OK”, use “All`[[:space:]]`is`[[:space:]]`OK”. Note that this may depend on your particular implementation of the regex functions found in your C library. Thanks to Charles Goyard for this tip.

Note: If you are migrating from the “cont2.sh” script, you must change the ‘_’ used as wildcards by cont2.sh into ‘.’ which is the regular-expression wildcard character.

Message digests can use different algorithms: “md5”, “sha1”, “sha256”, “sha512” or “rmd160”. The digest is calculated on the data portion of the response from the server, i.e. HTTP headers are not included in the digest (as they change from one request to the next). Note that the md5 and sha1 hash algorithms are no longer considered cryptographically safe.

The expected digest value can be computed with the *bbdigest(1)* utility.

“cont” tags in bb-hosts result in two status reports: One status with the “http” check, and another with the “content” check.

As with normal URL’s, the extended syntax described above can be used e.g. when testing SSL sites that require the use of SSLv2 or strong ciphers.

The column name for the result of the content check is by default called “content” - you can change the default with the “–content=NAME” option to *bbtest-net(1)*. See *bbtest-net(1)* for a description of this option.

If more than one content check is present for a host, the first content check is reported in the column “content”, the second is reported in the column “content1”, the third in “content2” etc.

You can also specify the columnname directly in the test specification, by writing it as “cont=COLUMN;`http://...``“. Column-names cannot include whitespace or semi-colon.

The content-check status by default includes the full URL that was requested, and the HTML data returned by the server. You can hide the HTML data on a per-host (not per-test) basis by adding the **HIDEHTTP** tag to the host entry.

content=URL This syntax is deprecated. You should use the “cont” tag instead, see above.

post[=COLUMN ;URL;form-data;[expected_data_regex—#digesttype:digest]] This tag can be used to test web pages, that use an input form. Data can be posted to the form by specifying them in the form-data field, and the result can be checked as if it was a normal content check (see above for a description of the cont-tag and the restrictions on how the URL must be written).

The form-data field by default is in the “application/x-www-form-urlencoded” format, which is the most commonly used format for web forms.

E.g. if you have a web form defined like this:

```
<form action="/cgi-bin/form.cgi" method="post">
<p>Givenname<input type="text" name="givenname"></p>
<p>Surname<input type="text" name="surname"></p>
<input type="submit" value="Send">
</form>
```

and you want to post the value “John” to the first field and “Doe Jr.” to the second field, then the formdata field would be

```
givenname=John&surname=Doe\20Jr.
```

Note that any spaces in the input value is replaced with ‘+’.

If your form-data requires a different content-type, you can specify it by beginning the form-data with (**content-type=TYPE**), e.g. “(content-type=text/xml)” followed by the POST data. Note that as with normal forms, the POST data should be specified using escape-sequences for reserved characters: “space” should be entered as “\x20”, double quote as “\x22”, newline as “\n”, carriage-return as “\r”, TAB as “\t”, backslash as “\\”. Any byte value can be entered using “\xNN” with NN being the hexdecimal value, e.g. “\x20” is the space character.

The [expected_data_regex—#digesttype:digest] is the expected data returned from the server in response to the POST. See the “cont;” tag above for details. If you are only interested in knowing if it is possible to submit the form (but don’t care about the data), this can be an empty string - but the ‘;’ at the end is required.

nocont[=COLUMN ;URL;forbidden_data_regex] This tag works just like “cont” tag, but reverses the test. It is green when the “forbidden_data_regex” is NOT found in the response, and red when it IS found. So it can be used to watch for data that should NOT be present in the response, e.g. a server error message.

nopost[=COLUMN ;URL;form-data;expected_data_regex] This tag works just like “post” tag, but reverses the test. It is green when the “forbidden_data_regex” is NOT found in the response, and red when it IS found. So it can be used to watch for data that should NOT be present in the response, e.g. a server error message.

type[=COLUMN ;URL;expected_content_type] This is a variant of the content check - instead of checking the content data, it checks the type of the data as given by the HTTP Content-Type: header. This can be used to check if a URL returns e.g. a PDF file, regardless of what is inside the PDF file.

httpstatus[=COLUMN ;URL;okstatusexpr;notokstatusexpr] This is used to explicitly test for certain HTTP status-codes returned when the URL is requested. The **okstatusexpr** and **notokstatusexpr** expressions are Perl-compatible regular expressions, e.g. “2..—302” will match all OK codes and the redirect (302) status code. If the URL cannot be retrieved, the status is “999”.

The status display for HTTP checks usually includes the URL, and for content checks also the actual data from the webpage. If you would like to hide these from view, then the HIDEHTTP tag will keep this information from showing up on the status webpages.

browser=BROWSERNAME By default, xmon sends an HTTP “User-Agent” header identifying it as “xmon”. Some websites require that you use a specific browser, typically Internet Explorer. To cater for testing of such sites, this tag can be used to modify the data sent in the User-Agent header. E.g. to perform an HTTP test with

xymon masquerading as an Internet Explorer 6.0 browser, use **browser="Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"**. If you do not know what the User-Agent header should be, open up the browser that works with this particular site, and open the URL “javascript:document.writeln(navigator.userAgent)” (just copy this into the “Open URL” dialog. The text that shows up is what the browser sends as the User-Agent header.

11.2.17 LDAP (DIRECTORY SERVER) TESTS

Simple check for an LDAP service. This check merely looks for any service running on the ldap/ldaps service port, but does not perform any actual LDAP transaction.

ldap://hostport/dn[?attrs[?scope[?filter[?exts]]]] Check for an LDAP service by performing an LDAP request. This tag is in the form of an LDAP URI (cf. RFC 2255). This type of LDAP test requires that *bbtest-net(1)* was built with support for LDAP, e.g. via the OpenLDAP library. The components of the LDAP URI are:

```
\emph{hostport}
is a host name with an optional ":portnumber"
\emph{dn}
is the search base
\emph{attrs}
is a comma separated list of attributes to request
\emph{scope}
is one of these three strings:
    base one sub (default=base)
\emph{filter}
is filter
\emph{exts}
are recognized set of LDAP and/or API extensions.
```

ldaps://hostport/dn[?attrs[?scope[?filter[?exts]]]] LDAP service check using LDAPv3 and STARTTLS for talking to an LDAP server that requires TLS encryption. See *bbtest-net(1)* for a discussion of the different ways of running LDAP servers with SSL/TLS, and which of these are supported by bbtest-net.

ldaplogin=username:password Define a username and password to use when binding to the LDAP server for ldap URI tests. If not specified, bbtest-net will attempt an anonymous bind.

Used with an LDAP URL test. If the LDAP query fails during the search of the directory, the ldap status is normally reported as “red” (alarm). This tag reduces a search failure to a “yellow” (warning) status.

11.2.18 PERFORMANCE MONITORING TESTS

apache[=URL] If you are running an Apache webserver, adding this tag makes *bbtest-net(1)* collect performance statistics from the Apache webserver by querying the URL **http://IP.ADDRESS.OF.HOST/server-status?auto**. The response is sent as a data-report and processed by the xymon xymond_rrd module into an RRD file and an “apache” graph. If your webserver requires e.g. authentication, or runs on a different URL for the server-status, you can provide the full URL needed to fetch the server-status page, e.g. **apache=http://LOGIN:PASSWORD@10.0.0.1/server-status?auto** for a password protected server-status page, or **apache=http://10.0.0.1:8080/apache/server-status?auto** for a server listening on port 8080 and with a different path to the server-status page.

Note that you need to enable the server-status URL in your Apache configuration. The following configuration is needed:

```
<Location/server-status>
SetHandlerserver-status
Orderdeny,allow
Denyfromall
allowfrom127.0.0.1
</Location>
```

ExtendedStatusOn

Change “127.0.0.1” to the IP-address of the server that runs your network tests.

11.2.19 DEFAULT HOST

If you have certain tags that you want to apply to all hosts, you can define a host name “.default.” and put the tags on that host. Note that per-host definitions will override the default ones.

NOTE: The “.default.” host entry will only accept the following tags - others are silently ignored: NOCOLUMNS, COMMENT, DESCRIPTOR, CLASS, dialup, testip, nobb2, nodisp, noinfo, notrends, TRENDS, NOPROPRED, NOPROPY-ELLOW, NOPROPPURPLE, NOPROPACK, REPORTTIME, WARNPCT, NET, nuclear, nosslcert, ssldays, DOWNTIME, depends, noping, noconn, trace, notrace, HIDEHTTP, browser, pulldata. Specifically, note that network tests, “badTEST” settings, and alternate pageset relations cannot be listed on the “.default.” host.

11.2.20 SENDING SUMMARIES TO REMOTE HOBBIT SERVERS

summary ROW.COLUMN IP URL If you have multiple xymon servers, the “summary” directive lets you form a hierarchy of servers by sending the overall status of this server to a remote xymon server, which then displays this in a special summary section. E.g. if your offices are spread over three locations, you can have a xymon server at each office. These branch-office xymons have a “summary” definition in their bb-hosts file that makes them report the overall status of their branch xymon to the central Xymon server you maintain at the corporate headquarters.

Multiple “summary” definitions are allowed.

The ROW.COLUMN setting defines how this summary is presented on the server that receives the summary. The ROW text will be used as the heading for a summary line, and the COLUMN defines the name of the column where this summary is shown - like the hostname and testname used in the normal displays. The IP is the IP-address of the **remote** (upstream) xymon server, where this summary is sent). The URL is the URL of your **local** xymon server.

The URL need not be that of your xymon server’s main page - it could be the URL of a subpage on the local xymon server. Xymon will report the summary using the color of the page found at the URL you specify. E.g. on your corporate xymon server you want a summary from the Las Vegas office - but you would like to know both what the overall status is, and what is the status of the servers on the critical Sales department back-office servers in Las Vegas. So you configure the Las Vegas xymon server to send **two** summaries:

```
summaryVegas.All10.0.1.1http://vegas.foo.com/xymon/
summaryVegas.Sales10.0.1.1http://vegas.foo.com/xymon/sales/
```

This gives you one summary line for Baltimore, with two columns: An “All” column showing the overall status, and a “Sales” column showing the status of the “sales” page on the Baltimore xymon server.

Note: Pages defined using alternate pageset definitions cannot be used, the URL must point to a webpage from the default set of xymon webpages.

11.2.21 OTHER TAGS

pulldata[=IP [:port]] This option is recognized by the *xymonfetch(8)* utility, and causes it to poll the host for client data. The optional IP-address and port-number can be used if the client-side *msgcache(8)* daemon is listening on a non-standard IP-address or port-number.

11.2.22 FILES

xymon/server/etc/bb-hosts

11.2.23 SEE ALSO

bbgen(1), *bbtest-net(1)*, *bbdigest(1)*, *xymonserver.cfg(5)*, *hobbit(7)*

11.3 xymon-alerts.cfg - Configuration for the hobbitd_alert module

11.3.1 SYNOPSIS

xymon/server/etc/hobbit-alerts.cfg

11.3.2 DESCRIPTION

The xymon-alerts.cfg file controls the sending of alerts by xymon when monitoring detects a failure.

11.3.3 FILE FORMAT

The configuration file consists of **rules**, that may have one or more **recipients** associated. A recipient specification may include additional rules that limit the circumstances when this recipient is eligible for receiving an alert.

Blank lines and lines starting with a hash mark (#) are treated as comments and ignored. Long lines can be broken up by putting a backslash at the end of the line and continuing the entry on the next line.

11.3.4 RULES

A rule consists of one or more filters using these keywords:

PAGE=targetstring Rule matching an alert by the name of the page in BB. This is the path of the page as defined in the bb-hosts file. E.g. if you have this setup:

Example

```
page servers All Servers
      subpage web Webservers
      10.0.0.1 www1.foo.com
      subpage db Database servers
      10.0.0.2 db1.foo.com
```

Then the “All servers” page is found with **PAGE=servers**, the “Webservers” page is **PAGE=servers/web** and the “Database servers” page is **PAGE=servers/db**. Note that you can also use regular expressions to specify the page name, e.g. **PAGE=%.*?db** would find the “Database servers” page regardless of where this page was placed in the hierarchy.

The PAGE name of top-level page is an empty string. To match this, use **PAGE=%\$** to match the empty string.

EXPAGE=targetstring Rule excluding an alert if the pagename matches.

HOST=targetstring Rule matching an alert by the hostname.

EXHOST=targetstring Rule excluding an alert by matching the hostname.

SERVICE=targetstring Rule matching an alert by the service name.

EXSERVICE=targetstring Rule excluding an alert by matching the service name.

GROUP=groupname Rule matching an alert by the group name. Groupnames are assigned to a status via the GROUP setting in the xymon-clients.cfg file.

EXGROUP=groupname Rule excluding an alert by the group name. Groupnames are assigned to a status via the GROUP setting in the xymon-clients.cfg file.

COLOR=color[,color] Rule matching an alert by color. Can be “red”, “yellow”, or “purple”. The forms “!red”, “!yellow” and “!purple” can also be used to NOT send an alert if the color is the specified one.

TIME=timespecification Rule matching an alert by the time-of-day. This is specified as the DOWNTIME time-specification in the bb-hosts file.

DURATION>time, DURATION<time Rule matcing an alert if the event has lasted longer/shorter than the given duration. E.g. DURATION>1h (lasted longer than 1 hour) or DURATION<30 (only sends alerts the first 30 minutes). The duration is specified as a number, optionally followed by ‘m’ (minutes, default), ‘h’ (hours) or ‘d’ (days).

RECOVERED Rule matches if the alert has recovered from an alert state.

NOTICE Rule matches if the message is a “notify” message. This type of message is sent when a host or test is disabled or enabled.

The “targetstring” is either a simple pagename, hostname or servicename, OR a ‘%’ followed by a Perl-compatible regular expression. E.g. “HOST=%www(.*)” will match any hostname that begins with “www”. The same for the “groupname” setting.

11.3.5 RECIPIENTS

The recipients are listed after the initial rule. The following keywords can be used to define recipients:

MAIL address[,address] Recipient who receives an e-mail alert. This takes one parameter, the e-mail address.

SCRIPT /path/to/script recipientID Recipient that invokes a script. This takes two parameters: The script filename, and the recipient that gets passed to the script.

IGNORE This is used to define a recipient that does NOT trigger any alerts, and also terminates the search for more recipients. It is useful if you have a rule that handles most alerts, but there is just that one particular server where you don't want CPU alerts on Monday morning. Note that the IGNORE recipient always has the STOP flag defined, so when the IGNORE recipient is matched, no more recipients will be considered. So the location of this recipient in your set of recipients is important.

FORMAT=formatstring Format of the text message with the alert. Default is “TEXT” (suitable for e-mail alerts). “PLAIN” is the same as text, but without the URL link to the status webpage. “SMS” is a short message with no subject for SMS alerts. “SCRIPT” is a brief message template for scripts.

REPEAT=time How often an alert gets repeated. As with DURATION, time is a number optionally followed by ‘m’, ‘h’ or ‘d’.

UNMATCHED The alert is sent to this recipient ONLY if no other recipients received an alert for this event.

STOP Stop looking for more recipients after this one matches. This is implicit on IGNORE recipients.

Rules You can specify rules for a recipient also. This limits the alerts sent to this particular recipient.

11.3.6 MACROS

It is possible to use **macros** in the configuration file. To define a macro:

`$MYMACRO=text extending to end of line`

After the definition of a macro, it can be used throughout the file. Wherever the text \$MYMACRO appears, it will be substituted with the text of the macro before any processing of rules and recipients.

It is possible to nest macros, as long as the macro is defined before it is used.

11.3.7 ALERT SCRIPTS

Alerts can go out via custom scripts, by using the **SCRIPT** keyword for a recipient. Such scripts have access to the following environment variables:

BBALPHAMSG The full text of the status log triggering the alert

ACKCODE The “cookie” that can be used to acknowledge the alert

RCPT The recipientID from the **SCRIPT** entry

BBHOSTNAME The name of the host that the alert is about

MACHIP The IP-address of the host that has a problem

BBSVCNAME The name of the service that the alert is about

BBSVCNUM The numeric code for the service. From the SVCODES definition.

BBHOSTSVC HOSTNAME.SERVICE that the alert is about.

BBHOSTSVCCOMMAS As BBHOSTSVC, but dots in the hostname replaced with commas

BBNUMERIC A 22-digit number made by BBSVCNUM, MACHIP and ACKCODE.

RECOVERED Is “1” if the service has recovered.

EVENTSTART Timestamp when the current status (color) began.

SECS Number of seconds the service has been down.

DOWNSECMSG When recovered, holds the text “Event duration : N” where N is the DOWNSecs value.

CFID Line-number in the xymon-alerts.cfg file that caused the script to be invoked. Can be useful when troubleshooting alert configuration rules.

11.3.8 SEE ALSO

xymond_alert(8), hobbitd(8), hobbit(7), the “Configuring xymon Alerts” guide in the Online documentation.

11.4 xymon-clients.cfg - Configuration file for the hobbitd_client module

11.4.1 SYNOPSIS

xymon/server/etc/hobbit-clients.cfg

11.4.2 DESCRIPTION

The xymon-clients.cfg file controls what color is assigned to the status-messages that are generated from the xymon client data - typically the cpu, disk, memory, procs- and msgs-columns. Color is decided on the basis of some **settings** defined in this file; settings apply to specific hosts through a set of **rules**.

Note: This file is only used on the xymon server - it is not used by the Xymon client, so there is no need to distribute it to your client systems.

11.4.3 FILE FORMAT

Blank lines and lines starting with a hash mark (#) are treated as comments and ignored.

11.4.4 CPU STATUS COLUMN SETTINGS

LOAD warnlevel paniclevel

If the system load exceeds “warnlevel” or “paniclevel”, the “cpu” status will go yellow or red, respectively. These are decimal numbers.

Defaults: warnlevel=5.0, paniclevel=10.0

UP bootlimit toolonglimit

The cpu status goes yellow if the system has been up for less than “bootlimit” time, or longer than “toolonglimit”. The time is in minutes, or you can add h/d/w for hours/days/weeks - eg. “2h” for two hours, or “4w” for 4 weeks.

Defaults: bootlimit=1h, toolonglimit=-1 (infinite).

CLOCK max.offset

The cpu status goes yellow if the system clock on the client differs more than “max.offset” seconds from that of the xymon server. Note that this is not a particularly accurate test, since it is affected by network delays between the client and the server, and the load on both systems. You should therefore not rely on this being accurate to more than +/- 5 seconds, but it will let you catch a client clock that goes completely wrong. The default is NOT to check the system clock. **NOTE:** Correct operation of this test obviously requires that the system clock of the xymon server is correct. You should therefore make sure that the Xymon server is synchronized to the real clock, e.g. by using NTP.

Example: Go yellow if the load average exceeds 5, and red if it exceeds 10. Also, go yellow for 10 minutes after a reboot, and after 4 weeks uptime. Finally, check that the system clock is at most 15 seconds offset from the clock of the xymon system.

Example

```
LOAD 5 10
UP 10m 4w
CLOCK 15
```

11.4.5 DISK STATUS COLUMN SETTINGS

DISK filesystem warnlevel paniclevel DISK filesystem IGNORE

If the utilization of “filesystem” is reported to exceed “warnlevel” or “paniclevel”, the “disk” status will go yellow or red, respectively. “warnlevel” and “paniclevel” are either the percentage used, or the space available as reported by the local “df” command on the host. For the latter type of check, the “warnlevel” must be followed by the letter “U”, e.g. “1024U”.

The special keyword “IGNORE” causes this filesystem to be ignored completely, i.e. it will not appear in the “disk” status column and it will not be tracked in a graph. This is useful for e.g. removable devices, backup-disks and similar hardware.

“filesystem” is the mount-point where the filesystem is mounted, e.g. “/usr” or “/home”. A filesystem-name that begins with “%” is interpreted as a Perl-compatible regular expression; e.g. “%^oracle.*” will match any filesystem whose mountpoint begins with “/oracle”.

Defaults: warnlevel=90%, paniclevel=95%

11.4.6 MEMORY STATUS COLUMN SETTINGS

MEMPHYS warnlevel paniclevel

MEMACT warnlevel paniclevel

MEMSWAP warnlevel paniclevel

If the memory utilization exceeds the “warnlevel” or “paniclevel”, the “memory” status will change to yellow or red, respectively. Note: The words “PHYS”, “ACT” and “SWAP” are also recognized.

Example: Go yellow if more than 20% swap is used, and red if more than 40% swap is used or the actual memory utilisation exceeds 90%. Dont alert on physical memory usage.

```
MEMSWAP 20 40
MEMACT 90 90
MEMPHYS 101 101
```

Defaults:

Example

```
MEMPHYS warnlevel=100 paniclevel=101 (i.e. it will never go red).
MEMSWAP warnlevel=50 paniclevel=80
MEMACT warnlevel=90 paniclevel=97
```

PAGING warnlevel paniclevel

Paging rates (Page in & out per second) for operating systems that measure that data (currently only z/VM and z/VME). Defaults are warnlevel=5, paniclevel=10.

11.4.7 PROCS STATUS COLUMN SETTINGS

PROC processname minimumcount maximumcount color [TRACK=id] [TEXT=text]

The “ps” listing sent by the client will be scanned for how many processes containing “processname” are running, and this is then matched against the min/max settings defined here. If the running count is outside the thresholds, the color of the “procs” status changes to “color”.

To check for a process that must NOT be running: Set minimum and maximum to 0.

“processname” can be a simple string, in which case this string must show up in the “ps” listing as a command. The scanner will find a ps-listing of e.g. “/usr/sbin/cron” if you only specify “processname” as “cron”. “processname” can also be a Perl-compatiable regular expression, e.g. “%^java.*inst[0123]” can be used to find entries in the ps-listing for “java -Xmx512m inst2” and “java -Xmx256 inst3”. In that case, “processname” must begin with “%” followed by the regular expression. Note that xymon defaults to case-insensitive pattern matching; if that is not what you want, put “(?i)” between the “%” and the regular expression to turn this off. E.g. “%^(?i)HTTPD” will match the word HTTPD only when it is upper-case. If “processname” contains whitespace (blanks or TAB), you must enclose the full string in double quotes - including the “%” if you use regular expression matching. E.g.

```
PROC "%xymond_channel-channel=data.*hobbitd_rrd" 1 1 yellow
```

or

PROC"java-DCLASSPATH=/opt/java/lib"25

You can have multiple "PROC" entries for the same host, all of the checks are merged into the "procs" status and the most severe check defines the color of the status.

The optional **TRACK=id** setting causes xymon to track the number of processes found in an RRD file, and put this into a graph which is shown on the "procs" status display. The **id** setting is a simple text string which will be used as the legend for the graph, and also as part of the RRD filename. It is recommended that you use only letters and digits for the ID. Note that the process counts which are tracked are only performed once when the client does a poll cycle - i.e. the counts represent snapshots of the system state, not an average value over the client poll cycle. Therefore there may be peaks or dips in the actual process counts which will not show up in the graphs, because they happen while the xymon client is not doing any polling.

The optional **TEXT=text** setting is used in the summary of the "procs" status. Normally, the summary will show the "processname" to identify the process and the related count and limits. But this may be a regular expression which is not easily recognizable, so if defined, the **text** setting string will be used instead. This only affects the "procs" status display - it has no effect on how the rule counts or recognizes processes in the "ps" output.

Example: Check that "cron" is running: PROC cron

Example: Check that at least 5 "httpd" processes are running, but not more than 20: PROC httpd 5 20

Defaults: mincount=1, maxcount=-1 (unlimited), color="red".

Note that no processes are checked by default.

11.4.8 MSGS STATUS COLUMN SETTINGS

LOG logfilepattern [COLOR=color] [IGNORE=excludepattern]

The xymon client extracts interesting lines from one or more logfiles - see the *client-local.cfg(5)* man-page for information about how to configure which logs a client should look at.

The **LOG** setting determine how these extracts of log entries are processed, and what warnings or alerts trigger as a result.

"logfilepattern" is the name of the logfile. Only logentries from this filename will be matched against this rule. Note that "logfilepattern" can be a regular expression (if prefixed with a '%' character).

"pattern" is a string or regular expression. If the logfile data matches "pattern", it will trigger the "msg" column to change color. If no "color" parameter is present, the default is to go "red" when the pattern is matched. To match against a regular expression, "pattern" must begin with a '%' sign - e.g "%WARNING—NOTICE" will match any lines containing either of these two words. Note that xymon defaults to case-insensitive pattern matching; if that is not what you want, put "(?-i)" between the "%" and the regular expression to turn this off. E.g. "%(?-i)WARNING" will match the word WARNING only when it is upper-case.

"excludepattern" is a string or regular expression that can be used to filter out any unwanted strings that happen to match "pattern".

Example: Trigger a red alert when the string "ERROR" appears in the "/var/adm/syslog" file: LOG /var/adm/syslog ERROR

Example: Trigger a yellow warning on all occurrences of the word "WARNING" or "NOTICE" in the "daemon.log" file, except those from the "lpr" system: LOG /var/log/daemon.log %WARNING—NOTICE COLOR=yellow IGNORE=lpr

Defaults: color="red", no "excludepattern".

Note that no logfiles are checked by default. Any log data reported by a client will just show up on the "msg" column with status OK (green).

11.4.9 FILES STATUS COLUMN SETTINGS

FILE filename [color] [things to check] [TRACK]

DIR directoryname [color] [size<MAXSIZE] [size>MINSIZE] [TRACK]

These entries control the status of the "files" column. They allow you to check on various data for files and directories.

filename and **directoryname** are names of files or directories, with a full path. You can use a regular expression to match the names of files and directories reported by the client, if you prefix the expression with a '%' character.

color is the color that triggers when one or more of the checks fail.

The **TRACK** keyword causes the size of the file or directory to be tracked in an RRD file, and presented in a graph on the “files” status display.

For files, you can check one or more of the following:

noexist triggers a warning if the file exists. By default, a warning is triggered for files that have a FILE entry, but which do not exist.

type=TYPE where TYPE is one of “file”, “dir”, “char”, “block”, “fifo”, “socket” or “symlink”. Triggers warning if the file is not of the specified type.

ownerid=OWNER triggers a warning if the owner does not match what is listed here. OWNER is specified either with the numeric uid, or the user name.

groupid=GROUP triggers a warning if the group does not match what is listed here. GROUP is specified either with the numeric gid, or the group name.

mode=MODE triggers a warning if the file permissions are not as listed. MODE is written in the standard octal notation, e.g. “644” for the rw-r-r- permissions.

size<MAX.SIZE and size>MIN.SIZE triggers a warning if the file size is greater than MAX.SIZE or less than MIN.SIZE, respectively. For filesizes, you can use the letters “K”, “M”, “G” or “T” to indicate that the filesize is in Kilobytes, Megabytes, Gigabytes or Terabytes, respectively. If there is no such modifier, Kilobytes is assumed. E.g. to warn if a file grows larger than 1MB, use **size<1024M**.

mtime>MIN.MTIME mtime<MAX.MTIME checks how long ago the file was last modified (in seconds). E.g. to check if a file was updated within the past 10 minutes (600 seconds): **mtime<600**. Or to check that a file has NOT been updated in the past 24 hours: **mtime>86400**.

mtime=TIMESTAMP checks if a file was last modified at TIMESTAMP. TIMESTAMP is a unix epoch time (seconds since midnight Jan 1 1970 UTC).

ctime>MIN.CTIME, ctime<MAX.CTIME, ctime=TIMESTAMP acts as the mtime checks, but for the ctime timestamp (when the directory entry of the file was last changed, eg. by chown, chgrp or chmod).

md5=MD5SUM, sha1=SHA1SUM, rmd160=RMD160SUM, sha256=SHA256SUM, sha512=SHA512SUM trigger a warning if the file checksum using the MD5, SHA1, RMD160, SHA-256 or SHA-512 message digest algorithms do not match the one configured here. Note: The “file” entry in the *client-local.cfg(5)* file must specify which algorithm to use.

For directories, you can check one or more of the following:

size<MAX.SIZE and size>MIN.SIZE triggers a warning if the directory size is greater than MAX.SIZE or less than MIN.SIZE, respectively. Directory sizes are reported in whatever unit the **du** command on the client uses - often KB or diskblocks - so MAX.SIZE and MIN.SIZE must be given in the same unit.

Experience shows that it can be difficult to get these rules right. Especially when defining minimum/maximum values for file sizes, when they were last modified etc. The one thing you must remember when setting up these checks is that the rules describe criteria that must be met - only when they are met will the status be green.

So “**mtime<600**” means “the difference between current time and the mtime of the file must be less than 600 seconds - if not, the file status will go red”.

11.4.10 PORTS STATUS COLUMN SETTINGS

PORT criteria [MIN=mincount] [MAX=maxcount] [COLOR=color] [TRACK=id] [TEXT=displaytext]

The “netstat” listing sent by the client will be scanned for how many sockets match the **criteria** listed. The criteria you can use are:

LOCAL=addr “addr” is a (partial) local address specification in the format used on the output from netstat.

EXLOCAL=addr Exclude certain local addresses from the rule.

REMOTE=addr “addr” is a (partial) remote address specification in the format used on the output from netstat.

EXREMOTE=addr Exclude certain remote addresses from the rule.

STATE=state Causes only the sockets in the specified state to be included, “state” is usually LISTEN or ESTABLISHED but can be any socket state reported by the clients “netstat” command.

EXSTATE=state Exclude certain states from the rule.

“addr” is typically “10.0.0.1:80” for the IP 10.0.0.1, port 80. Or “*:80” for any local address, port 80. Note that the xymon clients normally report only the numeric data for IP-addresses and port-numbers, so you must specify the port number (e.g. “80”) instead of the service name (“www”). “addr” and “state” can also be a Perl-compatible regular expression, e.g. “LOCAL=%[.](80—443)” can be used to find entries in the netstat local port for both http (port 80) and https (port 443). In that case, portname or state must begin with “%” followed by the reg.expression.

The socket count found is then matched against the min/max settings defined here. If the count is outside the thresholds, the color of the “ports” status changes to “color”. To check for a socket that must NOT exist: Set minimum and maximum to 0.

The optional **TRACK=id** setting causes xymon to track the number of sockets found in an RRD file, and put this into a graph which is shown on the “ports” status display. The **id** setting is a simple text string which will be used as the legend for the graph, and also as part of the RRD filename. It is recommended that you use only letters and digits for the ID. Note that the sockets counts which are tracked are only performed once when the client does a poll cycle - i.e. the counts represent snapshots of the system state, not an average value over the client poll cycle. Therefore there may be peaks or dips in the actual sockets counts which will not show up in the graphs, because they happen while the xymon client is not doing any polling.

The **TEXT=displaytext** option affects how the port appears on the “ports” status page. By default, the port is listed with the local/remote/state rules as identification, but this may be somewhat difficult to understand. You can then use e.g. “TEXT=Secure Shell” to make these ports appear with the name “Secure Shell” instead.

Defaults: mincount=1, maxcount=-1 (unlimited), color=”red”. Note: No ports are checked by default.

Example: Check that the SSH daemon is listening on port 22. Track the number of active SSH connections, and warn if there are more than 5.

```
PORTLOCAL=%[.]:22$STATE=LISTEN$TEXT=SSHlistener"  
PORTLOCAL=%[.]:22$STATE=ESTABLISHEDMAX=5$TRACK=ssh$TEXT=SSH
```

11.4.11 CHANGING THE DEFAULT SETTINGS

If you would like to use different defaults for the settings described above, then you can define the new defaults after a **DEFAULT** line. E.g. this would explicitly define all of the default settings:

Example

```
DEFAULT  
    UP      1h  
    LOAD    5.0 10.0  
    DISK    * 90 95  
    MEMPHYS 100 101  
    MEMSWAP 50 80  
    MEMACT  90 97
```

11.4.12 RULES TO SELECT HOSTS

All of the settings can be applied to a group of hosts, by preceding them with rules. A rule defines one or more filters using these keywords (note that this is identical to the rule definitions used in the *xymon-alerts.cfg(5)* file).

PAGE=targetstring Rule matching an alert by the name of the page in BB. “targetstring” is the path of the page as defined in the bb-hosts file. E.g. if you have this setup:

Example

```
page servers All Servers
subpage web Webservers
10.0.0.1 www1.foo.com
subpage db Database servers
10.0.0.2 db1.foo.com
```

Then the “All servers” page is found with **PAGE=servers**, the “Webservers” page is **PAGE=servers/web** and the “Database servers” page is **PAGE=servers/db**. Note that you can also use regular expressions to specify the page name, e.g. **PAGE=%.*/db** would find the “Database servers” page regardless of where this page was placed in the hierarchy.

The top-level page has a the fixed name **/**, e.g. **PAGE=/** would match all hosts on the xymon frontpage. If you need it in a regular expression, use **PAGE=%^/** to avoid matching the forward-slash present in subpage-names.

EXPAGE=targetstring Rule excluding a host if the pagename matches.

HOST=targetstring Rule matching a host by the hostname. “targetstring” is either a comma-separated list of hostnames (from the bb-hosts file), “*” to indicate “all hosts”, or a Perl-compatible regular expression. E.g. “HOST=dns.foo.com,www.foo.com” identifies two specific hosts; “HOST=%www.*.foo.com EXHOST=www-test.foo.com” matches all hosts with a name beginning with “www”, except the “www-test” host.

EXHOST=targetstring Rule excluding a host by matching the hostname.

CLASS=classname Rule match by the client class-name. You specify the class-name for a host when starting the client through the “–class=NAME” option to the runclient.sh script. If no class is specified, the host by default goes into a class named by the operating system.

EXCLAS=classname Exclude all hosts belonging to “classname” from this rule.

TIME=timespecification Rule matching by the time-of-day. This is specified as the DOWNTIME time specification in the bb-hosts file. E.g. “TIME=W:0800:2200” applied to a rule will make this rule active only on week-days between 8AM and 10PM.

11.4.13 DIRECTING ALERTS TO GROUPS

For some tests - e.g. “procs” or “msgs” - the right group of people to alert in case of a failure may be different, depending on which of the client rules actually detected a problem. E.g. if you have PROCS rules for a host checking both “httpd” and “sshd” processes, then the Web admins should handle httpd-failures, whereas “sshd” failures are handled by the Unix admins.

To handle this, all rules can have a “GROUP=groupname” setting. When a rule with this setting triggers a yellow or red status, the groupname is passed on to the xymon alerts module, so you can use it in the alert rule definitions in *xymon-alerts.cfg(5)* to direct alerts to the correct group of people.

11.4.14 RULES: APPLYING SETTINGS TO SELECTED HOSTS

Rules must be placed after the settings, e.g.

Example

```
LOAD 8.0 12.0  HOST=db.foo.com TIME=*:0800:1600
```

If you have multiple settings that you want to apply the same rules to, you can write the rules *only* on one line, followed by the settings. E.g.

Example

```
HOST=%db.*.foo.com TIME=W:0800:1600
    LOAD 8.0 12.0
    DISK /db 98 100
    PROC mysqld 1
```

will apply the three settings to all of the “db” hosts on week-days between 8AM and 4PM. This can be combined with per-settings rule, in which case the per-settings rule overrides the general rule; e.g.

Example

```
HOST=%.*.foo.com
    LOAD 7.0 12.0 HOST=bax.foo.com
    LOAD 3.0 8.0
```

will result in the load-limits being 7.0/12.0 for the “bax.foo.com” host, and 3.0/8.0 for all other foo.com hosts.

The entire file is evaluated from the top to bottom, and the first match found is used. So you should put the specific settings first, and the generic ones last.

11.4.15 NOTES

For the LOG, FILE and DIR checks, it is necessary also to configure the actual file- and directory-names in the *client-local.cfg(5)* file. If the filenames are not listed there, the clients will not collect any data about these files/directories, and the settings in the xymon-clients.cfg file will be silently ignored.

The ability to compute file checksums with MD5, SHA1 or RMD160 should not be used for general-purpose file integrity checking, since the overhead of calculating these on a large number of files can be significant. If you need this, look at tools designed for this purpose - e.g. Tripwire or AIDE.

At the time of writing (april 2006), the SHA-1 and RMD160 algorithms are considered cryptographically safe. The MD5 algorithm has been shown to have some weaknesses, and is not considered strong enough when a high level of security is required.

11.4.16 SEE ALSO

xymond_client(8), client-local.cfg(5), hobbitd(8), hobbit(7)

11.5 HOBBITCGI.CFG

xymoncgi.cfg - Command-line parameters for the xymon CGI tools

11.5.1 SYNOPSIS

\$BBHOME/etc/xymoncgi.cfg

11.5.2 DESCRIPTION

xymoncgi.cfg(1) controls the commandline options passed to all of the xymon CGI tools through their respective shell-script wrappers. Typically the options listed here are used for system-wide configuration of the CGI utilities, e.g. to define where they read configuration files.

The exact set of commandline options available are described in the man-page for each of the CGI utilities.

The file is “sourced” into the shell script wrapper, so assignments to the CGI-specific variables must follow standard shell-script syntax.

11.5.3 SETTINGS

Options for the *xymon-ackinfo.cgi(1)* utility.

Options for the *bb-ack.cgi(1)* utility.

Options for the *bb-csvinfo.cgi(1)* utility.

Options for the *bb-datepicker.cgi(1)* utility.

Options for the *xymon-enadis.cgi(8)* utility.

Options for the *bb-eventlog.cgi(1)* utility.

Options for the *bb-findhost.cgi(1)* utility.

Options for the *bb-hist.cgi(1)* utility.

xymon-specific options for column documentation. This uses the *bb-csvinfo.cgi(1)* utility with the **server/etc/columndoc.cfg** configuration file.

Options for the *xymon-confreport.cgi(1)* utility.

Options for the *xymongraph.cgi(1)* utility.

Options for the *xymon-hostgraphs.cgi(1)* utility.

Options for the *xymon-nkedit.cgi(1)* utility.

Options for the *xymon-nkview.cgi(1)* utility.

Options for the *bb-relog.cgi(1)* utility.

Options for the *bb-rep.cgi(1)* utility.

Options for the *bb-snapshot.cgi(1)* utility.

Options for the *xymonsvc.cgi(1)* utility when used to view historical logs. Note that the “–historical” option must be included in this setting.

Options for the *xymonsvc.cgi(1)* utility.

11.5.4 SEE ALSO

xymon(7), the individual CGI utility man-pages.

11.6 HOBBIT-NKVIEW.CGI

xymon-nkview.cgi - xymon Critical Systems view CGI

11.6.1 SYNOPSIS

xymon-nkview.cgi

11.6.2 DESCRIPTION

xymon-nkview.cgi is invoked as a CGI script via the `xymon-nkview.sh` CGI wrapper.

`xymon-nkview.cgi` matches the current critical statuses against the `hobbit-nkview.cfg(5)` file, and generates the “Critical Systems” view.

11.6.3 RELATION TO BBNK PAGE

This view is a replacement for the statically generated “bbnk” page provided in versions of xymon prior to version 4.2. Although the “bbnk” pages are supported throughout xymon 4.x, it is recommended that You switch to the newer Critical Systems view provided by this CGI.

11.6.4 OPTIONS

Sets the acknowledgment level for acknowledgments sent via the `xymon-ackinfo.cgi(1)` page. Note that this may be overridden by the configuration of the `xymon-ackinfo.cgi` utility.

Loads the environment defined in FILENAME before executing the CGI script.

Load environment variables for a specific area. NB: if used, this option must appear before any `-env=FILENAME` option.

Enables debugging output.

11.6.5 ENVIRONMENT VARIABLES

Used to locate the template files for the generated web pages.

Contains the parameters for the CGI script.

11.6.6 SEE ALSO

`xymon-ackinfo.cgi(1)`, `hobbit(7)`

11.7 CLIENT-LOCAL.CFG

11.7.1 NAME

client-local.cfg - Local configuration settings for xymon clients

11.7.2 SYNOPSIS

`xymon/server/etc/client-local.cfg`

11.7.3 DESCRIPTION

The client-local.cfg file contains settings that are used by each xymon client when it runs on a monitored host. It provides a convenient way of configuring clients from a central location without having to setup special configuration maintenance tools on all clients.

The client-local.cfg file is currently used to configure what logfiles the client should fetch data from, to be used as the basis for the “msgs” status column; and to configure which files and directories are being monitored in the “files” status column.

Note that there is a dependency between the client-local.cfg file and the *xymon-clients.cfg(5)* file. When monitoring e.g. a logfile, you must first enter it into the client-local.cfg file, to trigger the xymon client into reporting any data about the logfile. Next, you must configure xymon-clients.cfg so the xymon server knows what to look for in the file data sent by the client. So: client-local.cfg defines what raw data is collected by the client, and xymon-clients.cfg defines how to analyze them.

11.7.4 PROPAGATION TO CLIENTS

The client-local.cfg file resides on the xymon server.

When clients connect to the xymon server to send in their client data, they will receive part of this file back from the xymon server. The configuration received by the client is then used the next time the client runs.

This method of propagating the configuration means that there is a delay of up to two poll cycles (i.e. 5-10 minutes) from a configuration change is entered into the client-local.cfg file, and until you see the result in the status messages reported by the client.

11.7.5 FILE FORMAT

The file is divided into sections, delimited by “[name]” lines. A section name can be either an operating system identifier - linux, solaris, hp-ux, aix, freebsd, openbsd, netbsd, darwin - or a hostname. When deciding which section to send to a client, xymon will first look for a section named after the hostname of the client; if such a section does not exist, it will look for a section named by the operating system of the client. So you can configure special configurations for individual hosts, and have a default configuration for all other hosts of a certain type.

Apart from the section delimiter, the file format is free-form, or rather it is defined by the tools that make use of the configuration.

11.7.6 LOGFILE CONFIGURATION ENTRIES

A logfile configuration entry looks like this:

```
log:/var/log/messages:10240
ignoreMARK
triggerOops
```

The **log:FILENAME:SIZE** line defines the filename of the log, and the maximum amount of data (in bytes) to send to the xymon server. FILENAME is usually an explicit full-path filename on the client. If it is enclosed in backticks, it is a command which the xymon client runs and each line of output from this command is then used as

a filename. This allows scripting which files to monitor, e.g. if you have logfiles that are named with some sort of timestamp.

The **ignore PATTERN** line (optional) defines lines in the logfile which are ignored entirely, i.e. they are stripped from the logfile data before sending it to the xymon server. It is used to remove completely unwanted “noise” entries from the logdata processed by xymon. “PATTERN” is a regular expression.

The **trigger PATTERN** line (optional) is used only when there is more data in the log than the maximum size set in the “log:FILENAME:SIZE” line. The “trigger” pattern is then used to find particularly interesting lines in the logfile - these will always be sent to the xymon server. After picking out the “trigger” lines, any remaining space up to the maximum size is filled in with the most recent entries from the logfile. “PATTERN” is a regular expression.

11.7.7 COUNTING LOGENTRIES

A special type of log-handling is possible, where the number of lines matching a regular expressions are merely counted. This is **linecount:FILENAME**, followed by a number of lines of the form **ID:PATTERN**

. E.g.

```
linecount:/var/log/messages
diskerrors:I/Oerror.*device.*hd
badlogins:Failedlogin
```

11.7.8 FILE CONFIGURATION ENTRIES

A file monitoring entry is used to watch the meta-data of a file: Owner, group, size, permissions, checksum etc. It looks like this:

```
file:/var/log/messages[:HASH]
```

The **file:FILENAME** line defines the filename of the file to monitor. As with the “log:” entries, a filename enclosed in backticks means a command which will generate the filenames dynamically. The optional [:HASH] setting defines what type of hash to compute for the file: **md5**

, **sha1** , **rmd160** , **sha256** or **sha512** . By default, no hash is calculated. **NOTE:** If you want to check multiple files using a wildcard, you **must** use a command to generate the filenames. Putting wildcards directly into the **file:** entry will not work.

11.7.9 DIRECTORY CONFIGURATION ENTRIES

A directory monitoring entry is used to watch the size of a directory and any sub-directories. It looks like this:

```
dir:DIRECTORYNAME
```

The **dir:DIRECTORYNAME** line defines the filename of the file to monitor. As with the “log:” entries, a filename enclosed in backticks means a command which will generate the filenames dynamically. The xymon client will run the *du(1)* command with the directoryname as parameter, and send the output back to the xymon server. **NOTE:** If you want to check multiple directories using a wildcard, you **must** use a command to generate the directory names. Putting wildcards directly into the **dir:** entry will not work. E.g. use something like *dir:‘find /var/log -maxdepth 1 -type d’*

The “du” command used can be configured through the **DU** environment variable. On some systems, by default **du** reports data in disk blocks instead of KB (e.g. Solaris). So you may want to configure the xymon client to use a **du** command which reports data in KB, e.g. by setting

DU=”du-k” in the xymonclient.cfg file.

11.7.10 NOTES

The ability of the xymon client to calculate file hashes and monitor those can be used for file integrity validation on a small scale. However, there is a significant processing overhead in calculating these every time the xymon client runs, so this should not be considered a replacement for host-based intrusion detection systems such as Tripwire or AIDE.

Use of the directory monitoring on directory structures with a large number of files and/or sub-directories can be quite ressource-intensive.

11.7.11 SEE ALSO

xymon-clients.cfg(5), *hobbitd_client(8)*, *hobbitd(8)*, *hobbit(7)*

Chapter 12

Web page generation

12.1 BBGEN

bbgen - xymon webpage generator

12.1.1 SYNOPSIS

bbgen -?

bbgen --help

bbgen --version

bbgen [options] [output-directory]

(See the OPTIONS section for a description of the available commandline options).

12.1.2 DESCRIPTION

bbgen generates the overview webpages for the xymon monitor. These are the webpages that show the overall status of your hosts, not the detailed status pages for each test.

12.1.3 OPTIONS

bbgen has a large number of commandline options. The options can be used to change the behaviour of **bbgen** and affect the web pages generated by it.

12.1.4 GENERAL OPTIONS

-help or -? Provide a summary of available commandline options.

-version Prints the version number of **bbgen**

-docurl=URL Make hostnames be hyperlinks to documentation, accessed via a common web page (typically a CGI script or a PHP-driven dynamic page). The URL parameter is a formatting string with the name of the web page - you can put a "%s" in it which will be replaced by the hostname being accessed. E.g. if you use the bb-notes extension from www.deadcat.net, you would enable this with "**-docurl=xymon/admin/notes.php?host=%s**". For the host www.storner.dk this will result in a link to "[/xymon/admin/notes.php?host=www.storner.dk](http://xymon/admin/notes.php?host=www.storner.dk)".

-doccgi=URL This option is deprecated; please use **-docurl** instead.

-no-doc-window By default, links to documentation for hosts and services cause a new window to appear with the information. With this option, the documentation will appear in the same window as the xymon status.

- htmlextension=.EXTENSION** Sets the filename extension used for the webpages generated by bbgen. By default, an extension of “.html” is used. Note that you need to specify the “dot”.
- report[=COLUMNNAME]** With this option, bbgen will send a status message with details of how many hosts were processed, how many pages were generated, any errors that occurred during the run, and some timing statistics. The default columnname is “bbgen”.
- htaccess[=htaccess-filename]** Create .htaccess files when new web page directories are created. The content of the .htaccess files are determined by the BBHTACCESS environment variable (for the top-level directory with bb.html and bb2.html); by the BBPAGEHTACCESS variable (for the page-level directories); and by the BBSUBPAGEHTACCESS variable for subpage- and subparent-level directories. The filename of the .htaccess files default to “.htaccess” if no filename is given with this option. The BBHTACCESS variable is copied verbatim into the top-level .htaccess file. The BBPAGEHTACCESS variable may contain a “%s” where the name of the page is inserted. The BBSUBPAGEHTACCESS variable may contain two “%s” instances: The first is replaced with the pagename, the second with the subpagename.
- max-eventcount=N** Limit the eventlog on the BB2 page to only N events. Default: 100.
- max-eventtime=N** Limit the eventlog on the BB2 page to events that happened within the past N minutes. Default: 240.
- no-eventlog** Disable the eventlog normally displayed on the BB2 page
- max-ackcount=N** Limit the acknowledgment log on the BB2 page to only N events. Default: 25.
- max-acktime=N** Limit the acknowledgment log on the BB2 page to acks that happened within the past N minutes. Default: 240.
- no-acklog** Disable the acknowledgement log normally displayed on the BB2 page.
- nklog[=NK log column]** This generates a text-based log of what is shown on the bbnk.html status page, and sends a status message for the BBDISPLAY server itself reflecting the color of the NK status page. This allows you to track when problems have appeared on the bbnk status page. The logfile is stored in \$BBHOME/nkstatus.log

12.1.5 PAGE LAYOUT OPTIONS

These options affect how the webpages generated by bbgen appear in the browser.

-pages-last Put page- and subpage-links after hosts.

-pages-first Put page- and subpage-links before hosts (default).

These two options decide whether a page with links to subpages and hosts have the hosts or the subpages first.

-subpagecolumns=N Determines the number of columns used for links to pages and subpages. The default is N=1.

-maxrows=N Column headings on a page are by default only shown at the beginning of a page, subpage or group of hosts. This options causes the column headings to repeat for every N hosts shown.

-pagetitle-links Normally, only the colored “dots” next to a page or subpage act as links to the page itself. With this option, the page title will link to the page also.

-pagetext-headings Use the description text from the “page” or “subpage” tags as a heading for the page, instead of the “Pages hosted locally” or other standard heading.

-no-underline-headings Normally, page headings are underlined using an HTML “horizontal ruler” tag. This option disables the underlining of headings.

-recentgifs[=MINUTES] Use images named COLOR-recent.gif for tests, where the test status has changed within the past 24 hours. These GIF files need to be installed in the \$BBHOME/www/gifs/ directory. By default, the threshold is set to 24 hours - if you want it differently, you can specify the time limit also. E.g. “-recentgifs=3h” will show the recent GIFs for only 3 hours after a status change.

-sort-group-only-items In a normal “group-only” directive, you can specify the order in which the tests are displayed, from left to right. If you prefer to have the tests listed in alphabetical order, use this option - the page will then generate “group-only” groups like it generates normal groups, and sort the tests alphabetically.

-dialupskin=URL If you want to visually show that a test is a dialup-test, you can use an alternate set of icons for the green/red/yellow>/etc. images by specifying this option. The URL parameter specified here overrides the normal setting from the BBSKIN environment variable, but only for dialup tests.

-reverseskin=URL Same as “-dialupskin”, but for reverse tests (tests with ‘!’ in front).

-tooltips=[always,never,main] Determines which pages use tooltips to show the description of the host (from the COMMENT entry in the *bb-hosts(5)* file). If set to **always**, tooltips are used on all pages. If set to **never**, tooltips are never used. If set to **main**, tooltips are used on the main pages, but not on the BB2 (all non-green) or NK (critical systems) pages.

12.1.6 COLUMN SELECTION OPTIONS

These options affect which columns (tests) are included in the webpages generated by bbgen.

-ignorecolumns=test[,test] The given columns will be completely ignored by bbgen when generating webpages. Can be used to generate reports where you eliminate some of the more noisy tests, like “msgs”.

-nk-reds-only Only red status columns will be included on the NK page. By default, the NK page will contain hosts with red, yellow and clear status.

-bb2colors=COLOR[,COLOR] Defines which colors cause a test to appear on the “All non-green” status page (a.k.a. the BB2 page). COLOR is red, yellow or purple. The default is to include all three.

-bb2-ignorecolumns=test[,test] Same as the **-ignorecolumns**, but applies to hosts on the BB2 page only.

-bb2-ignorepurples Deprecated, use “**-bb2colors**” instead.

-bb2-ignoredialups Ignore all dialup hosts on the BB2 page, including the BB2 eventlog.

-includecolumns=test[,test] Always include these columns on bb2 page Will include certain columns on the bb2.html page, regardless of its color. Normally, bb2.html drops a test-column, if all tests are green. This can be used e.g. to always have a link to the trends column (with the RRD graphs) from your bb2.html page.

-eventignore=test[,test] Ignore these tests in the BB2 event log display.

12.1.7 STATUS PROPAGATION OPTIONS

These options suppress the normal propagation of a status upwards in the page hierarchy. Thus, you can have a test with status yellow or red, but still have the entire page green. It is useful for tests that need not cause an alarm, but where you still want to know the actual status. These options set global defaults for all hosts; you can use the NOPROPRED and NOPROPYELLOW tags in the *bb-hosts(5)* file to apply similar limits on a per-host basis.

-nopropyellow=test[,test] or **-noprop=test[,test]**] Disable upwards status propagation when YELLOW. The “**-noprop**” option is deprecated and should not be used.

-noproppurple=test[,test] Disable upwards status propagation when PURPLE.

-nopropred=test[,test] Disable upwards status propagation when RED or YELLOW.

-nopropack=test[,test] Disable upwards status propagation when status has been acknowledged. If you want to disable all acked tests from being propagated, use “-nopropack=*”.

12.1.8 PURPLE STATUS OPTIONS

Purple statuses occur when reporting of a test status stops. A test status is valid for a limited amount of time - normally 30 minutes - and after this time, the test becomes purple.

-purplelog=FILENAME Generate a logfile of all purple status messages.

12.1.9 ALTERNATE PAGESET OPTIONS

-pageset=PAGESETNAME Build webpages for an alternate pageset than the default. See the PAGESETS section below.

-template=TEMPLATE Use an alternate template for header and footer files. Typically used together with the “-pageset” option; see the PAGESETS section below.

12.1.10 ALTERNATE OUTPUT FORMATS

-wml[=test1,test2,...] This option causes bbgen to generate a set of WML “card” files that can be accessed by a WAP device (cell phone, PDA etc.) The generated files contain the hosts that have a RED or YELLOW status on tests specified. This option can define the default tests to include - the defaults can be overridden or amended using the “WML:” or “NK:” tags in the *bb-hosts(5)* file. If no tests are specified, all tests will be included.

-nstab=FILENAME Generate an HTML file suitable for a Netscape 6/Mozilla sidebar entry. To actually enable your users to obtain such a sidebar entry, you need this Javascript code in a webpage (e.g. you can include it in the \$BBHOME/web/bb_header file):

and then you can include a “Add this to sidebar” link using this as a template:

or if you prefer to have the standard Netscape “Add tab” button, you would do it with

The “add-button.gif” is available from Netscape at <http://developer.netscape.com/docs/manuals/browser/sidebar/add-button.gif>.

If FILENAME does not begin with a slash, the Netscape sidebar file is placed in the \$BBHOME/www/ directory.

-nslimit=COLOR The minimum color to include in the Netscape Sidebar - default is “red”, meaning only critical alerts are included. If you want to include warnings also, use “-nslimit=yellow”.

-rss Generate RSS/RDF content delivery stream of your xymon alerts. This output format can be dynamically embedded in other web pages, much like the live newsfeeds often seen on web sites. Two RSS files will be generated, one reflects the BB2 page, the other reflects the BBNK page. They will be in the “bb2.rss” and “bbnk.rss” files, respectively. In addition, an RSS file will be generated for each page and/or subpage listing the hosts present on that page or subpage.

The FILENAME parameter previously allowed on the -rss option is now obsolete. For more information about RSS/RDF content feeds, please see <http://www.syndic8.com/>.

-rssexception=.EXTENSION Sets the filename extension used for the RSS files generated by bbgen. By default, an extension of “.rss” is used. Note that you need to specify the “dot”.

-rssversion= 0.91—0.92—1.0—2.0 The desired output format of the RSS/RDF feed. Version 0.91 appears to be the most commonly used format, and is the default if this option is omitted.

-rsslimit=COLOR The minimum color to include in the RSS feed - default is “red”, meaning only critical alerts are included. If you want to include warnings also, use “-rsslimit=yellow”.

12.1.11 OPTIONS USED BY CGI FRONT-ENDS

-reportopts=START:END:DYNAMIC:STYLE Invoke bbgen in report-generation mode. This is normally used by the *bb-rep.cgi(1)*

CGI script, but may also be used directly when pre-generating reports. The START parameter is the start-time for the report in Unix time_t format (seconds since Jan 1st 1970 00:00 UTC); END is the end-time for the report; DYNAMIC is 0 for a pre-built report and 1 for a dynamic (on-line) report; STYLE is “crit” to include only critical (red) events, “non-crit” to include all non-green events, and “all” to include all events.

-csv=FILENAME Used together with –reportopts, this causes bbgen to generate an availability report in the form of a comma-separated values (CSV) file. This format is commonly used for importing into spreadsheets for further processing.

The CSV file includes Unix timestamps. To display these as human readable times in Excel, the formula
=C2/86400+DATEVALUE(1-jan-1970)

(if you have the Unix timestamp in the cell C2) can be used. The result cell should be formatted as a date/time field. Note that the timestamps are in UTC, so you may also need to handle local timezone and DST issues yourself.

-csvdelim=DELIMITER By default, a comma is used to delimit fields in the CSV output. Some non-english spreadsheets use a different delimiter, typically semi-colon. To generate a CSV file with the proper delimiter, you can use this option to set the character used as delimiter. E.g. “–csvdelim=;” - note that this normally should be in double quotes, to prevent the Unix shell from interpreting the delimiter character as a commandline delimiter.

-snapshot=TIME Generate a snapshot of the xymon pages, as they appeared at TIME. TIME is given as seconds since Jan 1st 1970 00:00 UTC. Normally used via the *bb-snapshot.cgi(1)* CGI script.

12.1.12 DEBUGGING OPTIONS

-debug Causes bbgen to dump large amounts of debugging output to stdout, if it was compiled with the -DDEBUG enabled. When reporting a problem with bbgen, please try to reproduce the problem and provide the output from running bbgen with this option.

-timing Dump information about the time spent by various parts of bbgen to stdout. This is useful to see what part of the processing is responsible for the run-time of bbgen.

Note: This information is also provided in the output sent to the xymon display when using the “–report” option.

12.1.13 BUILDING ALTERNATE PAGESETS

With version 1.4 of bbgen comes the possibility to generate multiple sets of pages from the same data.

Suppose you have two groups of people looking at the BB webpages. Group A wants to have the hosts grouped by the client, they belong to. This is how you have xymon set up - the default pageset. Now group B wants to have the hosts grouped by operating system - let us call it the “os” set. Then you would add the page layout to bb-hosts like this:

```
ospage win Microsoft Windows ossubpage win-nt4 MS Windows NT 4 osgroup NT4 File servers osgroup NT4
Mail servers ossubpage win-xp MS Windows XP ospage unix Unix ossubpage unix-sun Solaris ossubpage unix-linux
Linux
```

This defines a set of pages with one top-level page (the bb.html page), two pages linked from bb.html (win.html and unix.html), and from e.g. the win.html page there are subpages win-nt4.html and win-xp.html

The syntax is identical to the normal “page” and “subpage” directives in bb-hosts, but the directive is prefixed with the pageset name. Dont put any hosts in-between the page and subpage directives - just add all the directives at the top of the bb-hosts file.

How do you add hosts to the pages, then ? Simple - just put a tag “OS:win-xp” on the host definition line. The “OS” must be the same as prefix used for the pageset names, but in uppercase. The “win-xp” must match one of the pages or subpages defined within this pageset. E.g.

```
207.46.249.190 www.microsoft.com # OS:win-xp http://www.microsoft.com/ 64.124.140.181 www.sun.com
# OS:unix-sun http://www.sun.com/
```

If you want the host to appear inside a group defined on that page, you must identify the group by number, starting at 1. E.g. to put a host inside the “NT4 Mail servers” group in the example above, use “OS:win-nt4,2” (the second group on the “win-nt4” page).

If you want the host to show up on the frontpage instead of a subpage, use “OS:” .

All of this just defines the layout of the new pageset. To generate it, you must run bbgen once for each pageset you define - i.e. create an extension script like this:

```
#!/bin/sh

BBWEB="/xymon/os" $BBHOME/bin/bbgen \
--pageset=os --template=os \
$BBHOME/www/os/
```

Save this to \$BBHOME/ext/os-display.sh, and set this up to run as a xymon extension; this means adding an extra section to xymonlaunch.cfg to run it.

This generates the pages. There are some important options used here: * BBWEB=”/xymon/os” environment variable, and the “\$BBHOME/www/os/” option work together, and place the new pageset HTML files in a subdirectory off the normal xymon web root. If you normally access the Xymon pages as “<http://xymon.acme.com/hobbit/>”, you will then access the new pages as “<http://xymon.acme.com/hobbit/os/>”. NB: The directory given as BBWEB must contain a symbolic link to the \$BBHOME/www/html directory, or links to individual status messages will not work. Similar links should be made for the gifs/help/and not directories. * “–pageset=os” tells bbgen to structure the webpages using the “os” layout, instead of the default layout. * “–template=os” tells bbgen to use a different set of header-and-footer-templates. Normally bbgen uses the standard template in \$BBHOME/web/bb_header and .../bb_footer. With this option, it will instead use the files “os_header” and “os_footer” from the \$BBHOME/web/directory. This allows you to customize headers and footers for each pageset. If you just want to use the normal template, you can omit this option.

12.1.14 USING BBGEN FOR REPORTS

bbgen reporting is implemented via drop-in replacements for the standard xymon reporting scripts (bb-rep.sh and bb-replog.sh) installed in your webserver’s cgi-bin directory.

These two shell script have been replaced with two very small shell-scripts, that merely setup the xymon environment variables, and invoke the *bb-rep.cgi(1)* or *bb-replog.cgi(1)* scripts in \$BBHOME/bin/

You can use bbgen commandline options when generating reports, e.g. to exclude certain types of tests (e.g. “–ignorecolumns=msgs”) from the reports, to specify the name of the trends- and info- columns that should not be in the report, or to format the report differently (e.g. “–subpagecolumns=2”). If you want certain options to be used when a report is generated from the web interface, put these options into your \$BBHOME/etc/xymonserver.cfg file in the BBGENREOPTS environment variable.

The report files generated by bbgen are stored in individual directories (one per report) below the \$BBHOME/www/rep/ directory. These should be automatically cleaned up - as new reports are generated, the old ones get removed.

After installing, try generating a report. You will probably see that the links in the upper left corner (to bb-ack.html, bb2.html etc.) no longer works. To fix these, change your \$BBHOME/web/bbrep_header file so these links do not refer to “&BBWEB” but to the normal URL prefix for your xymon pages.

12.1.15 SLA REPORTING

bbgen reporting allows for the generation of true () reports, also for service periods that are not 24x7. This is enabled by defining a “REPORTTIME:timespec” tag for the hosts to define the service period, and optionally a “WARNPCT:level” tag to define the agreed availability.

Note: See *bb-hosts(5)* for the exact syntax of these options.

“REPORTTIME:timespec” specifies the time of day when the service is expected to be up and running. By default this is 24 hours a day, all days of the week. If your SLA only covers Mon-Fri 7am - 8pm, you define this as “REPORTTIME=W:0700:2000”, and the report generator will then compute both the normal 24x7 availability but also a “SLA availability” which only takes the status of the host during the SLA period into account.

The DOWNTIME:timespec parameter affects the SLA availability calculation. If an outage occurs during the time defined as possible “DOWNTIME”, then the failure is reported with a status of “blue”. (The same color is used if you “disable” then host using the xymon “disable” function). The time when the test status is “blue” is not included in the SLA calculation, neither in the amount of time where the host is considered down, nor in the total amount of time that the report covers. So “blue” time is effectively ignored by the SLA availability calculation, allowing you to have planned downtime without affecting the reported SLA availability.

Example: A host has “DOWNTIME:*:0700:0730 REPORTTIME=W:0600:2200” because it is rebooted every day between 7am and 7.30am, but the service must be available from 6am to 10pm. For the day of the report, it was down from 7:10am to 7:15am (the planned reboot), but also from 9:53pm to 10:15pm. So the events for the day are:

```
0700:greenfor10minutes(600seconds) 0710:bluefor5minutes(300seconds) 0715:greenfor14hours38minutes(52680seconds)
2153:redfor22minutes(1320seconds) 2215:green
```

The service is available for $600+52680 = 53280$ seconds. It is down (red) for 420 seconds (the time from 21:53 until 22:00 when the SLA period ends). The total time included in the report is 15 hours (7am - 10pm) except the 5 minutes blue = 53700 seconds. So the SLA availability is $53280/53700 = 99,22\%$

The “WARNPCT:level” tag is supported in the bb-hosts file, to set the availability threshold on a host-by-host basis. This threshold determines whether a test is reported as green, yellow or red in the reports. A default value can be set for all hosts with the via the BBREPWARN environment variable, but overridden by this tag. The level is given as a percentage, e.g. “WARNPCT:98.5”

12.1.16 PRE-GENERATED REPORTS

Normally, bbgen produce reports that link to dynamically generated webpages with the detailed status of a test (via the bb-relog.sh CGI script).

It is possible to have bbgen produce a report without these dynamic links, so the report can be exported to another server. It may also be useful to pre-generate the reports, to lower the load by having multiple users generate the same reports.

To do this, you must run bbgen with the “–reportopts” option to select the time interval that the report covers, the reporting style (critical, non-green, or all events), and to request that no dynamic pages are to be generated.

The syntax is:

```
bbgen-reportopts=starttime:endtime:nodynamic:style
```

“starttime” and “endtime” are specified as Unix time_t values, i.e. seconds since Jan 1st 1970 00:00 GMT. Fortunately, this can easily be computed with the GNU date utility if you use the “+%s” output option. If you don’t have the GNU date utility, either pick that up from www.gnu.org; or you can use the “etime” utility for the same purpose, which is available from the archive at www.deadcat.net.

“nodynamic” is either 0 (for dynamic pages, the default) or 1 (for no dynamic, i.e. pre-generated, pages).

“style” is either “crit” (include critical i.e. red events only), “nongr” (include all non-green events), or “all” (include all events).

Other bbgen options can be used, e.g. “–ignorecolumns” if you want to exclude certain tests from the report.

You will normally also need to specify the BBWEB environment variable (it must match the base URL for where the report will be made accessible from), and an output directory where the report files are saved. If you specify BBWEB, you should probably also define the BBHELPSSKIN and BBNOTESSIONSKIN environment variables. These

should point to the URL where your xymon help- and notes-files are located; if they are not defined, the links to help- and notes-files will point inside the report directory and will probably not work.

So a typical invocation of bbgen for a static report would be:

```
START='date+%s-date="22Jun200300:00:00'" END='date+%s-date="22Jun200323:59:59"' BBWEB=/reports/bigbrother/daily/2003/06/22  
BBHELPSSKIN=/xymon/help\ BBNOTESSKIN=/xymon/notes\ bbgen-reportopts=$START:$END:1:crit\ -subpagecolumns=2\ /var/www/docroot/reports/xymon/daily/2003/06/22
```

The “BBWEB” setting means that the report will be available with a URL of “<http://www.server.com/reports/xymon/daily/2003/06/22>”. The report contains internal links that use this URL, so it cannot be easily moved to another location.

The last parameter is the corresponding physical directory on your webserver matching the BBWEB URL. You can of course create the report files anywhere you like - perhaps on another machine - and then move them to the webserver later on.

Note how the *date(1)* utility is used to calculate the start- and end-time parameters.

12.1.17 SEE ALSO

bb-hosts(5), xymonserver.cfg(5), hobbitlaunch.cfg(5), bb-rep.cgi(1), bb-snapshot.cgi(1), hobbit(7)

12.2 BB-HIST.CGI

bb-hist.cgi - CGI program to display service history

12.2.1 SYNOPSIS

bb-hist.cgi

12.2.2 DESCRIPTION

bb-hist.cgi is invoked as a CGI script via the bb-hist.sh CGI wrapper. It is passed a QUERY_STRING environment variable with the following parameters:

HISTFILE(axymonservicehistoryfile) ENTRIES(the number of entries to show) The following non-standard parameters are handled by the bbgen version of bb-hist.cgi:

IP(IPAddressofhost-for display purposes only) PIXELS(width of colorbar when in pixel-mode) ENDTIME(when the colorbar begins, atime BARSUMS(which colorbars and summaries to show)

bb-hist.cgi analyses the service history file for changes that have occurred within the past 24 hours, and build a colorbar showing the status of the service over this period of time. A statistics summary is also produced, listing the amount of time for each status (green, yellow, red, purple, blue, clear).

Finally, a summary of the last N events is given, with links to the actual event logs.

Unlike the standard bb-hist.sh script, bb-hist.cgi provides a colorbar and statistics summaries also covering the past 1 week, 4 weeks and 1 year of data. Via links it is possible to browse the entire history of the service at the requested interval.

Note that since the resolution of the display is limited, events may be too short to show up on a colorbar; also, the exact placement of an event may not fully match up with the time-markers.

The graphs should correctly handle the display of months with different number of days, as well as the display of periods that involve beginning and end of Daylight Savings Time, if this occurs in your timezone.

All dates and times shown are in local time for the timezone defined on the BBDISPLAY server.

12.2.3 PARAMETERS

HISTFILE Defines the host and service whose history is presented.

ENTRIES The number of log-entries to show in the event log table. Default is 50; to view all log entries set this to “ALL”.

IP The IP-address of the host. This is only used for the title of the document.

PIXELS The width of the colorbar graph in pixels. If this is set to 0, a percentage-based graph will be shown, similar to the one provided by the standard bb-hist.sh script. Pixel-based graphs can have a higher resolution, but do not resize automatically to suit the size of a browser window. The default value for this parameter is defined at compile-time; 960 is a good value for displays with a 1024x768 resolution.

BARSUMS Defines which colorbars and summaries to show. This is a number made up from a bitmask. The 1-day graph uses the value “1”; the 1-week graph uses the value “2”; the 4-week graph uses the value “4” and the 1-year graph the value “8”. To show multiple graph, add the values - e.g. “6” will show the 1-week and 4-weeks graphs, whereas “15” will show all the graphs. The default is defined at compile-time.

ENDTIME The history display by default ends with the current time. Setting the ENDTIME parameter causes it to end at the time specified - this is given as a Unix “time_t” value, i.e. as the number of seconds elapsed since Jan 1 1970 00:00 UTC.

12.2.4 OPTIONS

-env=FILENAME Load the environment from FILENAME before executing the CGI.

12.2.5 SEE ALSO

bb-hosts(5), xymonserver.cfg(5)

12.3 HOBBIT-NKVIEW.CGI

xymon-nkview.cgi - xymon Critical Systems view CGI

12.3.1 SYNOPSIS

xymon-nkview.cgi

12.3.2 DESCRIPTION

xymon-nkview.cgi is invoked as a CGI script via the `hobbit-nkview.sh` CGI wrapper.

`xymon-nkview.cgi` matches the current critical statuses against the `xymon-nkview.cfg(5)` file, and generates the “Critical Systems” view.

12.3.3 RELATION TO BBNK PAGE

This view is a replacement for the statically generated “bbnk” page provided in versions of xymon prior to version 4.2. Although the “bbnk” pages are supported throughout xymon 4.x, it is recommended that You switch to the newer Critical Systems view provided by this CGI.

12.3.4 OPTIONS

- nkacklevel=NUMBER** Sets the acknowledgment level for acknowledgments sent via the `xymon-ackinfo.cgi(1)` page. Note that this may be overridden by the configuration of the `xymon-ackinfo.cgi` utility.
- env=FILENAME** Loads the environment defined in FILENAME before executing the CGI script.
- area=NAME** Load environment variables for a specific area. NB: if used, this option must appear before any `-env=FILENAME` option.
- debug** Enables debugging output.

12.3.5 ENVIRONMENT VARIABLES

BBHOME Used to locate the template files for the generated web pages.

QUERY_STRING Contains the parameters for the CGI script.

12.3.6 SEE ALSO

`xymon-ackinfo.cgi(1)`, `hobbit(7)`

12.4 BBHOSTGREP

`bbhostgrep` - pick out lines in bb-hosts

12.4.1 SYNOPSIS

```
bbhostgrep --help
bbhostgrep --version
bbhostgrep [-noextras] [-test-untagged] [-bbdisp] [-bbnet] TAG [TAG...]
```

12.4.2 DESCRIPTION

bbhostgrep(1) is for use by extension scripts that need to pick out the entries in a bb-hosts file that are relevant to the script.

The utility accepts test names as parameters, and will then parse the bb-hosts file and print out the host entries that have at least one of the wanted tests specified. Tags may be given with a trailing asterisk '*', e.g. "bbhostgrep http*" is needed to find all http and https tags.

The `bbhostgrep` utility supports the use of "include" directives inside the bb-hosts file, and will find matching tags in all included files.

If the DOWNTIME or SLA tags are used in the *bb-hosts(5)* file, these are interpreted relative to the current time. `bbhostgrep` then outputs a "INSIDESLA" or "OUTSIDESLA" tag for easier use by scripts that want to check if the current time is inside or outside the expected uptime window.

12.4.3 OPTIONS

-noextras Remove the "testip", "dialup", "INSIDESLA" and "OUTSIDESLA" tags from the output.

-test-untagged When using the BBLOCATION environment variable to test only hosts on a particular network segment, `bbtest-net` will ignore hosts that do not have any "NET:x" tag. So only hosts that have a NET:\$BBLOCATION tag will be tested. With this option, hosts with no NET: tag are included in the test, so that all hosts that either have a matching NET: tag, or no NET: tag at all are tested.

-no-down[=TESTNAME] `bbhostgrep` will query the xymon server for the current status of the "conn" test, and if TESTNAME is specified also for the current state of the specified test. If the status of the "conn" test for a host is non-green, or the status of the TESTNAME test is disabled, then this host is ignored and will not be included in the output. This can be used to ignore hosts that are down, or hosts where the custom test is disabled.

-bbdisp Search the bb-hosts file following include statements as a BBDISPLAY server would.

-bbnet Search the bb-hosts file following include statements as a BBNET server would.

12.4.4 EXAMPLE

If your bb-hosts file looks like this

192.168.1.1www.test.com#ftptelnet!oracle 192.168.1.2db1.test.com#oracle 192.168.1.3mail.test.com#smtp
and you have a custom xymon extension script that performs the "oracle" test, then running "bbhostgrep oracle" would yield

192.168.1.1www.test.com#!oracle 192.168.1.2db1.test.com#oracle

so the script can quickly find the hosts that are of interest.

Note that the reverse-test modifier - "!oracle" - is included in the output; this also applies to the other test modifiers defined by xymon (the dialup and always-true modifiers).

If your extension scripts use more than one tag, just list all of the interesting tags on the command line.

`bbhostgrep` also supports the "NET:location" tag used by `bbtest-net`, so if your script performs network checks then it will see only the hosts that are relevant for the test location that the script currently executes on.

12.4.5 USE IN EXTENSION SCRIPTS

To integrate bbhostgrep into an existing script, look for the line in the script that grep's in the \$BBHOSTS file. Typically it will look somewhat like this:

```
$GREP -i "[0-9].*#.TESTNAME" $BBHOSTS —...codetohandletest
```

Instead of the grep, we will use bbhostgrep. It then becomes

```
$BBHOME/bin/bbhostgrep TESTNAME —...codetohandletest
```

which is simpler, less error-prone and more efficient.

12.4.6 ENVIRONMENT VARIABLES

BBLOCATION If set, bbhostgrep outputs only lines from bb-hosts that have a matching NET:\$BBLOCATION setting.

BBHOSTS Filename for the xymon *bb-hosts(5)* file.

12.4.7 FILES

\$BBHOSTS The xymon bb-hosts file

12.4.8 SEE ALSO

bb-hosts(5), *xymonserver.cfg(5)*

12.5 HOBBITD

xymond - Master network daemon for a xymon server

12.5.1 SYNOPSIS

xymond [options]

12.5.2 DESCRIPTION

xymond is the core daemon in the xymon Monitor. It is designed to handle monitoring of a large number of hosts, with a strong focus on being a high-speed, low-overhead implementation of a Big Brother compatible server.

To achieve this, xymond stores all information about the state of the monitored systems in memory, instead of storing it in the host filesystem. A number of plug-ins can be enabled to enhance the basic operation; e.g. a set of plugins are provided to implement persistent storage in a way that is compatible with the Big Brother daemon. However, even with these plugins enabled, hobbitd still performs much faster than the standard bbd daemon.

xymond is normally started and controlled by the *hobbitlaunch(8)* tool, and the command used to invoke xymond should therefore be in the *hobbitlaunch.cfg* file.

12.5.3 OPTIONS

-bbhosts=FILENAME Specifies the path to the xymon bb-hosts file. This is used to check if incoming status messages refer to known hosts; depending on the “-ghosts” option, messages for unknown hosts may be dropped. If this option is omitted, the default path used is set by the BBHOSTS environment variable.

-checkpoint-file=FILENAME With regular intervals, xymond will dump all of its internal state to this check-point file. It is also dumped when hobbitd terminates, or when it receives a SIGUSR1 signal.

-checkpoint-interval=N Specifies the interval (in seconds) between dumps to the check-point file. The default is 900 seconds (15 minutes).

-restart=FILENAME Specifies an existing file containing a previously generated xymond checkpoint. When starting up, hobbitd will restore its internal state from the information in this file. You can use the same filename for “-checkpoint-file” and “-restart”.

-ghosts= allow—drop—log How to handle status messages from unknown hosts. The “allow” setting accepts all status messages, regardless of whether the host is known in the bb-hosts file or not. “drop” silently ignores reports from unknown hosts. “log” works like drop, but logs the event in the xymond output file. The default is “allow”, unless the BBGHOSTS environment variable is set.

-no-purple Prevent status messages from going purple when they are no longer valid. Unlike the standard bbd daemon, purple-handling is done by xymond.

-listen=IP[:PORT] Specifies the IP-address and port where xymond will listen for incoming connections. By default, hobbitd listens on IP 0.0.0.0 (i.e. all IP- addresses available on the host) and port 1984.

-daemon xymond is normally started by *hobbitlaunch(8)* it will then detach from the terminal and continue running as a background task.

-timeout=N Set the timeout used for incoming connections. If a status has not been received more than N seconds after the connection was accepted, then the connection is dropped and any status message is discarded. Default: 10 seconds.

-env=FILENAME Loads the content of FILENAME as environment settings before starting xymond. This is mostly used when running as a stand-alone daemon; if hobbitd is started by hobbitlaunch, the environment settings are controlled by the hobbitlaunch *hobbitlaunch.cfg* file.

-pidfile=FILENAME xymond writes the process-ID it is running with to this file. This is for use in automated startup scripts. The default file is \$BBSERVERLOGS/hobbitd.pid.

-log=FILENAME Redirect all output from xymond to FILENAME.

-store-clientlogs[!=! COLUMN]] Determines which status columns can cause a client message to be broadcast to the CLICHG channel. By default, no client messages are pushed to the CLICHG channel. If this option is specified with no parameter list, all status columns that go into an alert state will trigger the client data to be sent to the CLICHG channel. If a parameter list is added to this option, only those status columns listed in the list will cause the client data to be sent to the CLICHG channel. Several column names can be listed, separated by commas. If all columns are given as “!COLUMNNAME”, then all status columns except those listed will cause the client data to be sent.

-status-senders=IP[/MASK] [,IP/MASK]] Controls which hosts may send “status”, “combo”, “config” and “query” commands to xymond.

By default, any host can send status-updates. If this option is used, then status-updates are accepted only if they are sent by one of the IP-addresses listed here, or if they are sent from the IP-address of the host that the updates pertains to (this is to allow xymon clients to send in their own status updates, without having to list all clients here). So typically you will need to list your BBNET servers here.

The format of this option is a list of IP-addresses, optionally with a network mask in the form of the number of bits. E.g. if you want to accept status-updates from the host 172.16.10.2, you would use

`-status-senders=172.16.10.2` whereas if you want to accept status updates from both 172.16.10.2 and from all of the hosts on the 10.0.2.* network (a 24-bit IP network), you would use

`-status-senders=172.16.10.2,10.0.2.0/24`

-maint-senders=IP[/MASK] [,IP/MASK]] Controls which hosts may send maintenance commands to xymond. Maintenance commands are the “enable”, “disable”, “ack” and “notes” commands. Format of this option is as for the `-status-senders` option. It is strongly recommended that you use this to restrict access to these commands, so that monitoring of a host cannot be disabled by a rogue user - e.g. to hide a system compromise from the monitoring system.

Note: If messages are sent through a proxy, the IP-address restrictions are of little use, since the messages will appear to originate from the proxy server address. It is therefore strongly recommended that you do NOT include the address of a server running bbproxy in the list of allowed addresses.

-www-senders=IP[/MASK] [,IP/MASK]] Controls which hosts may send commands to retrieve the state of xymond. These are the “hobbitdlog”, “hobbitdboard” and “hobbitdxboard” commands, which are used by *bbgen(1)* and *bbcombotest(1)* to retrieve the state of the xymon system so they can generate the Xymon webpages.

Note: If messages are sent through a proxy, the IP-address restrictions are of little use, since the messages will appear to originate from the proxy server address. It is therefore strongly recommended that you do NOT include the address of a server running bbproxy in the list of allowed addresses.

-admin-senders=IP[/MASK] [,IP/MASK]] Controls which hosts may send administrative commands to xymond. These commands are the “drop” and “rename” commands. Access to these should be restricted, since they provide an un-authenticated means of completely disabling monitoring of a host, and can be used to remove all traces of e.g. a system compromise from the xymon monitor.

Note: If messages are sent through a proxy, the IP-address restrictions are of little use, since the messages will appear to originate from the proxy server address. It is therefore strongly recommended that you do NOT include the address of a server running bbproxy in the list of allowed addresses.

-no-download Disable the “download” and “config” commands which can be used by clients to pull files from the xymon server. The use of these may be seen as a security risk since they allow file downloads.

-debug Enable debugging output.

-dbghost=HOSTNAME For troubleshooting problems with a specific host, it may be useful to track the exact communications from a single host. This option causes xymond to dump all traffic from a single host to the file “/tmp/hobbitd.dbg”.

12.5.4 HOW ALERTS TRIGGER

When a status arrives, xymond matches the old and new color against the “alert” colors (from the “ALERTCOLORS” setting) and the “OK” colors (from the “OKCOLORS” setting). The old and new color falls into one of three categories:

OK: The color is one of the “OK” colors (e.g. “green”).

ALERT: The color is one of the “alert” colors (e.g. “red”).

UNDECIDED: The color is neither an “alert” color nor an “OK” color (e.g. “yellow”).

If the new status shows an ALERT state, then a message to the *xymond_alert(8)* module is triggered. This may be a repeat of a previous alert, but *xymond_alert(8)* will handle that internally, and only send alert messages with the interval configured in *xymon-alerts.cfg(5)*.

If the status goes from a not-OK state (ALERT or UNDECIDED) to OK, and there is a record of having been in a ALERT state previously, then a recovery message is triggered.

The use of the OK, ALERT and UNDECIDED states make it possible to avoid being flooded with alerts when a status flip-flops between e.g yellow and red, or green and yellow.

12.5.5 CHANNELS

A lot of functionality in the xymon server is delegated to “worker modules” that are fed various events from xymond via a “channel”. Programs access a channel using IPC mechanisms - specifically, shared memory and semaphores - or by using an instance of the *hobbitd_channel(8)* intermediate program. xymond_channel enables access to a channel via a simple file I/O interface.

A skeleton program for hooking into a xymond channel is provided as part of xymon in the *hobbitd_sample(8)* program.

The following channels are provided by xymond:

status This channel is fed the contents of all incoming “status” and “summary” messages.

stachg This channel is fed information about tests that change status, i.e. the color of the status-log changes.

page This channel is fed information about tests where the color changes between an alert color and a non-alert color. It also receives information about “ack” messages.

data This channel is fed information about all “data” messages.

notes This channel is fed information about all “notes” messages.

enadis This channel is fed information about hosts or tests that are being disabled or enabled.

client This channel is fed the contents of the client messages sent by xymon clients installed on the monitored servers.

clichg This channel is fed the contents of a host client messages, whenever a status for that host goes red, yellow or purple.

Information about the data stream passed on these channels is in the xymon source-tree, see the “xymond/new-daemon.txt” file.

12.5.6 SIGNALS

SIGHUP Re-read the bb-hosts configuration file.

SIGUSR1 Force an immediate dump of the checkpoint file.

12.5.7 BUGS

Timeout of incoming connections are not strictly enforced. The check for a timeout only triggers during the normal network handling loop, so a connection that should timeout after N seconds may persist until some activity happens on another (unrelated) connection.

12.5.8 FILES

If ghost-handling is enabled via the “–ghosts” option, the bb-hosts file is read to determine the names of all known hosts.

12.5.9 SEE ALSO

xymon(7), hobbitserver.cfg(5).

12.6 HOBBITPING

xymonping - xymon ping tool

12.6.1 SYNOPSIS

xymonping [-retries=N] [-timeout=N] [IP-adresses]

12.6.2 DESCRIPTION

xymonping(1) is used for ping testing of the hosts monitored by the *xymon(7)* monitoring system. It reads a list of IP adresses from stdin, and performs a “ping” check to see if these hosts are alive. It is normally invoked by the *bbtest-net(1)* utility, which performs all of the xymon network tests.

Optionally, if a list of IP-adresses is passed as commandline arguments, it will ping those IP's instead of reading them from stdin.

xymonping only handles IP-adresses, not hostnames.

xymonping was inspired by the *fping(1)* tool, but has been written from scratch to implement a fast ping tester without much of the overhead found in other such utilities. The output from xymonping is similar to that of “fping -Ae”.

xymonping probes multiple systems in parallel, and the runtime is therefore mostly dependant on the timeout-setting and the number of retries. With the default options, xymonping takes approximately 18 seconds to ping all hosts (tested with an input set of 1500 IP adreses).

12.6.3 SUID-ROOT INSTALLATION REQUIRED

xymonping needs to be installed with suid-root privileges, since it requires a “raw socket” to send and receive ICMP Echo (ping) packets.

xymonping is implemented such that it immediately drops the root privileges, and only regains them to perform two operations: Obtaining the raw socket, and optionally binding it to a specific source address. These operations are performed as root, the rest of the time xymonping runs with normal user privileges. Specifically, no user-supplied data or network data is used while running with root privileges. Therefore it should be safe to provide xymonping with the necessary suid-root privileges.

12.6.4 OPTIONS

-retries=N Sets the number of retries for hosts that fail to respond to the initial ping, i.e. the number of ping probes sent in addition to the initial probe. The default is -retries=2, to ping a host 3 times before concluding that it is not responding.

-timeout=N Determines the timeout (in seconds) for ping probes. If a host does not respond within N seconds, it is regarded as unavailable, unless it responds to one of the retries. The default is -timeout=5.

-responses=N xymonping normally stops pinging a host after receiving a single response, and uses that to determine the round-trip time. If the first response takes longer to arrive - e.g. because of additional network overhead when first determining the route to the target host - it may skew the round-trip-time reports. You can then use this option to require N responses, and xymonping will calculate the round-trip time as the average of all of responses times.

-max-pps=N Maximum number of packets per second. This limits the number of ICMP packets xymonping will send per second, by enforcing a brief delay after each packet is sent. The default setting is to send a maximum of 50 packets per second. Note that increasing this may cause flooding of the network, and since ICMP packets can be discarded by routers and other network equipment, this can cause erratic behaviour with hosts recorded as not responding when they are in fact OK.

-source=ADDRESS Use ADDRESS as the source IP address of the ping packets sent. On multi-homed systems, allows you to select the source IP of the hosts going out, which might be necessary for ping to work.

-debug Enable debug output. This prints out all packets sent and received.

12.6.5 SEE ALSO

xymon(7), bbtest-net(1), fping(1)

12.7 HOBBITD_ALERT

xymond_alert - hobbitd worker module for sending out alerts

12.7.1 SYNOPSIS

xymond_channel –channel=page hobbitd_alert [options]

12.7.2 DESCRIPTION

xymond_alert is a worker module for hobbitd, and as such it is normally run via the *xymond_channel(8)* program. It receives xymond page- and ack-messages from the “page” channel via stdin, and uses these to send out alerts about failed and recovered hosts and services.

The operation of this module is controlled by the *xymon-alerts.cfg(5)* file. This file holds the definition of rules and recipients, that determine who gets alerts, how often, for what servers etc.

12.7.3 OPTIONS

- config=FILENAME** Sets the filename for the *xymon-alerts.cfg* file. The default value is “etc/xymon-alerts.cfg” below the xymon server directory.
- dump-config** Dumps the configuration after parsing it. May be useful to track down problems with configuration file errors.
- checkpoint-file=FILENAME** File where the current state of the xymond_alert module is saved. When starting up, hobbitd_alert will also read this file to restore the previous state.
- checkpoint-interval=N** Defines how often (in seconds) the checkpoint-file is saved.
- cfid** If this option is present, alert messages will include a line with “cfid:N” where N is the linenumber in the xymon-alerts.cfg file that caused this message to be sent. This can be useful to track down problems with duplicate alerts.
- test HOST SERVICE [options]** Shows which alert rules matches the given HOST/SERVICE combination. Useful to debug configuration problems, and see what rules are used for an alert.
The possible options are:
 - color=COLORNAME** The COLORNAME parameter is the color of the alert: red, yellow or purple.
 - duration=SECONDS** The SECONDS parameter is the duration of the alert in seconds.
 - group=GROUPNAME** The GROUPNAME paramater is a groupid string from the xymon-clients.cfg file.
 - time=TIMESTRING** The TIMESTRING parameter is the time-of-day for the alert, expressed as an absolute time in the epoch format (seconds since Jan 1 1970). This is easily obtained with the GNU date utility using the “+%s” output format.
- debug** Enable debugging output.

12.7.4 HOW HOBBIT DECIDES WHEN TO SEND ALERTS

The xymond_alert module is responsible for sending out all alerts. When a status first goes to one of the ALERTCOLUMNS, xymond_alert is notified of this change. It notes that the status is now in an alert state, and records the **timestamp** when this event started, and adds the alert to the list statuses that may potentially trigger one or more alert messages.

This list is then matched against the xymon-alerts.cfg configuration. This happens at least once a minute, but may happen more often. E.g. when status first goes into an alert state, this will always trigger the matching to happen.

When scanning the configuration, xymond_alert looks at all of the configuration rules. It also checks the DURATION setting against how long time has elapsed since the event started - i.e. against the timestamp logged when xymond_alert first heard of this event.

When an alert recipient is found, the alert is sent and it is recorded when this recipient is due for his next alert message, based on the REPEAT setting defined for this recipient. The next time xymond_alert scans the configuration for what alerts to send, it will still find this recipient because all of the configuration rules are fulfilled, but an alert message will not be generated until the repeat interval has elapsed.

It can happen that a status first goes yellow and triggers an alert, and later it goes red - e.g. a disk filling up. In that case, xymond_alert clears the internal timer for when the next (repeat) alert is due for all recipients. You generally want to be told when something that has been in a warning state becomes critical, so in that case the REPEAT setting is ignored and the alert is sent. This only happens the first time such a change occurs - if the status switches between yellow and red multiple times, only the first transition from yellow->red causes this override.

When a status recovers, a recovery message may be sent - depending on the configuration - and then xymond_alert forgets everything about this status. So the next time it goes into an alert state, the entire process starts all over again.

12.7.5 ENVIRONMENT

MAIL The first part of a command line used to send out an e-mail with a subject, typically set to “/usr/bin/mail -s” . xymond_alert will add the subject and the mail recipients to form the command line used for sending out email alerts.

MAILC The first part of a command line used to send out an e-mail without a subject. Typically this will be “/usr/bin/mail”. xymond_alert will add the mail recipients to form the command line used for sending out email alerts.

12.7.6 FILES

xymon/server/etc/hobbit-alerts.cfg

12.7.7 SEE ALSO

xymon-alerts.cfg(5), hobbitd(8), hobbitd_channel(8), hobbit(7)

12.8 BBHOSTSHOW

- output the full bb-hosts file

12.8.1 SYNOPSIS

bbhostshow [-bbdisp] [-bbnet] [filename]

12.8.2 DESCRIPTION

bbhostshow(1) dumps the full bb-hosts file to stdout. It follows “include” tags in the bb-hosts files, and prints the full contents as seen by the *bbgen(1)* and *bbtest-net(1)* utilities.

If no filename is given, bbhostshow displays the file pointed to by the BBHOSTS environment variable.

12.8.3 OPTIONS

-bbdisp Show the bb-hosts file following include statements as a BBDISPLAY server would.

-bbnet Show the bb-hosts file following include statements as a BBNET server would.

12.8.4 ENVIRONMENT VARIABLES

BBHOSTS Filename for the *bb-hosts(5)* file.

12.8.5 SEE ALSO

bb-hosts(5), *xymonserver.cfg(5)*

12.9 HOBBITD_CAPTURE

xymond_capture - catch selected messages from a hobbitd channel

12.9.1 SYNOPSIS

xymond_channel –channel=status hobbitd_capture [options]

12.9.2 DESCRIPTION

xymond_capture is a worker module for hobbitd, and as such it is normally run via the *xymond_channel(8)* program. It receives messages from xymond via stdin and filters them to select messages based on the hostname, testname or color of the status. By default the resulting messages are printed on stdout, but they can also be fed into a command for further processing.

xymond_capture supports the **status** , **data** and **hostdata** channels.

12.9.3 OPTIONS

-hosts=PATTERN Select messages only from hosts matching PATTERN (regular expression).

-exhosts=PATTERN Exclude messages from hosts matching PATTERN. If used with the **-hosts** option, then the hostname must match the **-hosts** pattern, but NOT the **-exhosts** pattern.

-tests=PATTERN Select messages only from tests matching PATTERN (regular expression).

-extests=PATTERN Exclude messages from tests matching PATTERN. If used with the **-tests** option, then the test-name must match the **-tests** pattern, but NOT the **-extests** pattern.

-colors=COLOR[,color] Select messages based on the color of the status message. Multiple colors can be listed, separated by comma. Default: Accept all colors.

-batch-command=COMMAND Instead of printing the messages to stdout, feed them to COMMAND on stdin. COMMAND can be any command which accepts the message on standard input.

-batch-timeout=SECONDS Collect messages until no messages have arrived in SECONDS seconds, before sending them to the **-batch-command** COMMAND.

-debug Enable debugging output.

12.9.4 SEE ALSO

xymond_channel(8), *hobbitd(8)*, *hobbit(7)*

12.10 HOBBIT-STATUSREPORT.CGI

xymon-statusreport.cgi - CGI program to report a status for a group of servers

12.10.1 SYNOPSIS

xymon-statusreport.cgi –column=COLUMNNAME [options]

12.10.2 DESCRIPTION

xymon-statusreport.cgi is a CGI tool to generate a simple HTML report showing the current status of a single column for a group of xymon hosts.

E.g. You can use this report to get an overview of all of the SSL certificates that are about to expire.

The generated webpage is a simple HTML table, suitable for copying into other documents or e-mail.

xymon-statusreport.cgi runs as a CGI program, invoked by your webserver. It is normally run via a wrapper shell-script in the CGI directory for xymon.

12.10.3 EXAMPLES

The xymon installation includes two web report scripts using this CGI tool: The **xymon-certreport.sh** script generates a list of SSL server certificates that are yellow or red (i.e. they will expire soon); and the **xymon-nongreen.sh** script generates a report of all statuses that are currently non-green. These can be accessed from a web browser through a URL referencing the script in the xymon CGI directory (e.g. “/xymon-cgi/hobbit-nongreen.sh”).

12.10.4 OPTIONS

-column=COLUMNNAME Report the status of the COLUMNNAME column.

-all Report the status for all hosts known to xymon. By default, this tool reports only on the hosts found on the current page from where the CGI was invoked (by looking at the “pagepath” cookie).

-filter=CRITERIA Only report on statuses that match the CRITERIA setting. See the *bb(1)* man-page - in the “xymondboard” command description - for details about specifying filters.

-heading=HTML Defines the webpage heading - i.e. the “title” tag in the generated HTML code.

-show-column Include the column name in the display.

-show-colors Show the status color on the generated webpage. The default is to not show the status color.

-no-colors Do not include text showing the current color of each status in the report. This is the default.

-show-summary Show only a summary of the important lines in the status message. By default, the entire status message appears in the generated HTML code. This option causes the first non-blank line of the status message to be shown, and also any lines beginning with “&COLOR” which is used by many status messages to point out lines of interest (non-green lines only, though).

-show-message Show the entire message on the webpage. This is the default.

-link Include HTML links to the host “info” page, and the status page.

-embedded Only generate the HTML table, not a full webpage. This can be used to embed the status report into an external webpage.

-env=FILENAME Load the environment from FILENAME before executing the CGI.

-area=NAME Load environment variables for a specific area. NB: if used, this option must appear before any – env=FILENAME option.

12.10.5 SEE ALSO

xymon(7)

12.11 BBMESSAGE.CGI

bbmessage.cgi - CGI utility used for proxying xymon data over HTTP

12.11.1 SYNOPSIS

bbmessage.cgi

12.11.2 DESCRIPTION

bbmessage.cgi(8) is the server-side utility receiving xymon messages sent by the *bb(1)* utility over an HTTP transport. The **bb** utility normally sends data over a dedicated TCP protocol, but it may use HTTP to go through proxies or through restrictive firewalls. In that case, the webserver must have this CGI utility installed, which takes care of receiving the message via HTTP, and forwards it to a local xymon server through the normal Xymon transport.

The CGI expects to be invoked from an HTTP “POST” request, with the POST-data being the status-message. **bbmessage.cgi** simply collects all of the POST data, and send it off as a message to the xymon daemon running on IP 127.0.0.1. This destination IP currently cannot be changed.

The CGI will return any output provided by the xymon daemon back to the requestor as the response to the HTTP POST, so this allows for all normal xymon commands to work.

12.11.3 SECURITY

bbmesage.cgi will only send data to a xymon server through the loopback interface, i.e. IP-address 127.0.0.1.

Access to the CGI should be restricted through webserver access controls, since the CGI provides no authentication at all to validate incoming messages.

If possible, consider using the *bbproxy(8)* utility instead for native proxying of xymon data between networks.

12.11.4 SEE ALSO

bb(1), *bbproxy(8)*, *xymon(7)*

12.12 BB-WEBPAGE

bb-webpage - Utility to show a webpage using header and footer

12.12.1 SYNOPSIS

bb-webpage [options]

12.12.2 DESCRIPTION

bb-webpage is a tool to generate a webpage in the xymon style, with a standard header- and footer as well as a xymon background. The data to present on the webpage, apart from the header and footer, are passed to bb-webpage in stdin. The generated webpage is printed to stdout.

12.12.3 OPTIONS

-env=FILENAME Loads the environment defined in FILENAME before executing the CGI script.

-hfile=PREFIX Use the header- and footer-files in \$BBHOME/web/PREFIX_header and PREFIX_footer. If not specified, bb_header and bb_footer are used.

-color=COLOR Set the background color of the generated webpage to COLOR. Default: Blue

-debug Enable debugging output.

12.12.4 SEE ALSO

xymon(7)

Chapter 13

Report Generation

13.1 HOBBIT-CONFREPORT.CGI

xymon-confreport.cgi - xymon Configuration report

13.1.1 SYNOPSIS

xymon-confreport.cgi

13.1.2 DESCRIPTION

xymon-confreport.cgi is invoked as a CGI script via the xymon-confreport.sh CGI wrapper.

xymon-confreport.cgi provides a plain HTML (Web) report of the xymon configuration for a group of hosts; which hosts are included is determined by the hosts available on the webpage from where the CGI script is invoked.

The configuration report include the hostnames, a list of the statuses monitored for each host, and if applicable any configuration settings affecting these. Alerts that may be triggered by status changes are also included.

The report is plain HTML without any images included, and therefore suitable for inclusion into e-mails or other documents that may be accessed outside the xymon system.

13.1.3 OPTIONS

-critical Report only on the statuses that are configured to show up on the **Critical Systems** view.

-old-nk-config Use the deprecated **NK** tag in bb-hosts to determine if tests appear on the Critical Systems view.

-env=FILENAME Loads the environment defined in FILENAME before executing the CGI script.

-area=NAME Load environment variables for a specific area. NB: if used, this option must appear before any **-env=FILENAME** option.

-debug Enables debugging output.

13.1.4 BUGS

Client-side configuration done in the *xymon-clients.cfg(5)* is not currently reflected in the report.

Critical Systems view configuration is not reflected in the report.

13.1.5 SEE ALSO

bb-hosts(5), *xymon-alerts.cfg(5)*, *hobbit-clients.cfg(5)*, *hobbit(7)*

13.2 BB-REP.CGI

bb-rep.cgi - CGI front-end to bbgen reporting

13.2.1 SYNOPSIS

bb-rep.cgi [-noclean] [bbgen-options]

13.2.2 DESCRIPTION

bb-rep.cgi is invoked as a CGI script via the `bb-rep.sh` CGI wrapper. It triggers the generation of a xymon availability report for the timeperiod specified by the CGI parameters.

`bb-rep.cgi` is passed a `QUERY_STRING` environment variable with the following parameters:

`start-mon(Startmonthoftheresport) start-day(Startday-of-monthoftheresport) start-yr(Startyearoftheresport) end-mon(Endmonthoftheresport) end-day(Endday-of-monthoftheresport) end-yr(Endyearoftheresport) style(Reportstyle)` The following non-standard parameters are handled by the `bbgen` version of `bb-rep.cgi`:

`suburl(Pageinreporttogo,ifnotthetoppage)`

The “month” parameters must be specified as the three-letter english month name abbreviation: Jan, Feb, Mar ...

Start- and end-days are in the range 1..31; the start- and end-year must be specified including century (e.g. “2003”).

End-times beyond the current time are silently replaced with the current time.

The generated report will include data for the start- and end-days, i.e. the report will begin at 00:00:00 of the start-day, and end at 23:59:59 of the end-day.

The “style” parameter is passed directly to `bbgen(1)` and should be “crit”, “non-crit” or “all”. Other values result in undefined behaviour.

All of the processing involved in generating the report is done by invoking `bbgen(1)` with the proper “`-reportopts`” option.

13.2.3 OPTIONS

-noclean Do not clean the BBREP directory of old reports. Makes the report-tool go a bit faster - instead, you can clean up the BBREP directory e.g. via a cron-job.

-env=FILENAME Load the environment from FILENAME before executing the CGI.

bbgen-options All other options passed to `bb-rep.cgi` are passed on to the `bbgen(1)` program building the report files.

13.2.4 FILES

\$BBHOME/web/report_header HTML template header for the report request form

\$BBHOME/web/report_footer HTML template footer for the report request form

\$BBHOME/web/report_form HTML template report request form

13.2.5 ENVIRONMENT VARIABLES

BBGENREPOPTS `bbgen` options passed by default to the `bb-rep.cgi`. This happens in the `bb-rep.sh` wrapper.

BBHOME Home directory of the xymon server installation

BBREP Directory where generated reports are stored. This directory must be writable by the userid executing the CGI script, typically “www”, “apache” or “nobody”. Default: `$BBHOME/www/rep/`

BBREPURL The URL prefix to use when accessing the reports via a browser. Default: `$BBWEB/rep`

13.2.6 SEE ALSO

`bbgen(1)`, `bb-hosts(5)`, `xymonserver.cfg(5)`

13.3 HOBBITD_FILESTORE

xymond_filestore - hobbitd worker module for storing xymon data

13.3.1 SYNOPSIS

```
xymond_channel --channel=status hobbitd_filestore --status [options]
xymond_channel --channel=data hobbitd_filestore --data [options]
xymond_channel --channel=notes hobbitd_filestore --notes [options]
xymond_channel --channel=enadis hobbitd_filestore --enadis [options]
```

13.3.2 DESCRIPTION

xymond_filestore is a worker module for hobbitd, and as such it is normally run via the *hobbitd_channel(8)* program. It receives xymond messages from a hobbitd channel via stdin, and stores these in the filesystem in a manner that is compatible with the Big Brother daemon, bbd.

This program can be started multiple times, if you want to store messages for more than one channel.

13.3.3 OPTIONS

- status** Incoming messages are “status” messages, they will be stored in the \$BBLOGS/ directory. If you are using *xymon(7)* throughout your xymon server, you will not need to run this module to save status messages, unless you have a third-party add-on that reads the status-logs directly. This module is NOT needed to get trend graphs, you should run the *xymond_rrd(8)* module instead.
- data** Incoming messages are “data” messages, they will be stored in the \$BBDATA directory. This module is not needed, unless you have a third-party module that processes the data-files. This module is NOT needed to get trend graphs, you should run the *xymond_rrd(8)* module instead.
- notes** Incoming messages are “notes” messages, they will be stored in the \$BBNOTES directory. This modules is only needed if you want to allow people to remotely update the notes-files available on the xymon webpages.
- enadis** Incoming messages are enable/disable messages, they will update files in the \$BBDISABLED directory. This is only needed if you have third-party add-ons that use these files.
- dir=DIRECTORY** Overrides the default output directory.
- html** Used together with “-status”. Tells xymond_filestore to also save an HTML version of the status-log. Should not be used unless you must run with “BBLOGSTATUS=static”.
- htmldir=DIRECTORY** The directory where HTML-versions of the status logs are stored. Default: \$BBHTML
- htmlext=.EXT** Set the filename extension for generated HTML files. By default, HTML files are saved with a “.html” extension.
- multigraphs=TEST1[,TEST2]** This causes xymond_filestore to generate HTML status pages with links to service graphs that are split up into multiple images, with at most 5 graphs per image. If not specified, only the “disk” status is split up this way.
- only=test[,test,test]** Save status messages only for the listed set of tests. This can be useful if you have an external script that needs to parse some of the status logs, but you do not want to save all status logs.
- debug** Enable debugging output.

13.3.4 FILES

This module does not rely on any configuration files.

13.3.5 SEE ALSO

xymond_channel(8), hobbitd_rrd(8), hobbitd(8), hobbit(7)

Chapter 14

xymon Server Administration

14.1 HOBBIT-ENADIS.CGI

xymon-enadis.cgi - CGI program to enable/disable xymon tests

14.1.1 SYNOPSIS

xymon-enadis.cgi (invoked via CGI from webserver)

14.1.2 DESCRIPTION

xymon-enadis.cgi is a CGI tool for disabling and enabling hosts and tests monitored by xymon. You can disable monitoring of a single test, all tests for a host, or multiple hosts - immediately or at a future point in time.

xymon-enadis.cgi runs as a CGI program, invoked by your webserver. It is normally run via a wrapper shell-script in the secured CGI directory for xymon.

xymon-enadis.cgi is the back-end script for the enable/disable form present on the “info” status-pages. It can also run in “stand-alone” mode, in which case it displays a web form allowing users to select what to enable or disable.

14.1.3 OPTIONS

-no-cookies Normally, xymon-enadis.cgi uses a cookie sent by the browser to initially filter the list of hosts presented. If this is not desired, you can turn off this behaviour by calling bb-ack.cgi with the **-no-cookies** option. This would normally be placed in the **CGI_ENADIS_OPTS** setting in *xymoncgi.cfg(5)*

-env=FILENAME Load the environment from FILENAME before executing the CGI.

-area=NAME Load environment variables for a specific area. NB: if used, this option must appear before any **-env=FILENAME** option.

subsectionFILES

\$BBHOME/web/maint_header,form,footer HTML template header

14.1.4 BUGS

When using alternate pagesets, hosts will only show up on the Enable/Disable page if this is accessed from the primary page in which they are defined. So if you have hosts on multiple pages, they will only be visible for disabling from their main page which is not what you would expect.

14.1.5 SEE ALSO

xymon(7)

14.2 BB-ACK.CGI

bb-ack.cgi - xymon CGI script to acknowledge alerts

14.2.1 SYNOPSIS

bb-ack.cgi?ACTION=action&NUMBER=acknum&DELAY=validity&MESSAGE=text

14.2.2 DESCRIPTION

bb-ack.cgi is invoked as a CGI script via the bb-ack.sh CGI wrapper.

bb-ack.cgi is passed a QUERY_STRING environment variable with the ACTION, NUMBER, DELAY and MESSAGE parameters.

14.2.3 PARAMETERS

ACTION is the action to perform. The only supported action is “Ack” to acknowledge an alert.

NUMBER is the number identifying the host/service to be acknowledged. It is included in all alert-messages sent out by xymon.

DELAY is the time (in minutes) that the acknowledge is valid.

MESSAGE is an optional text which will be shown on the status page while the acknowledgment is active. You can use it to e.g. tell users not to contact you about the problem, or inform them when the problem is expected to be resolved.

14.2.4 OPTIONS

-no-pin bb-ack.cgi normally requires the user to enter the acknowledgment code received in an alert message. If you run it with this option, the user will instead get a list of the current non-green statuses, and he may send an acknowledge without knowing the code.

-no-cookies Normally, bb-ack.cgi uses a cookie sent by the browser to initially filter the list of hosts presented. If this is not desired, you can turn off this behaviour by calling bb-ack.cgi with the -no-cookies option. This would normally be placed in the CGI_ACK_OPTS setting in *xymoncgi.cfg(5)*

-env=FILENAME Loads the environment defined in FILENAME before executing the CGI script.

-debug Enables debugging output.

14.2.5 FILES

\$BBHOME/web/acknowledge_header HTML header file for the generated web page

\$BBHOME/web/acknowledge_footer HTML footer file for the generated web page

\$BBHOME/web/acknowledge_form Query form displayed when bb-ack.cgi is called with no parameters.

14.2.6 ENVIRONMENT VARIABLES

BBHOME Used to locate the template files for the generated web pages.

QUERY_STRING Contains the parameters for the CGI script.

14.2.7 BUGS

When using alternate pagesets, hosts will only show up on the acknowledgment page if this is accessed from the primary page in which they are defined. So if you have hosts on multiple pages, they will only be visible for acknowledging from their main page which is not what you would expect.

14.2.8 SEE ALSO

`bbgen(1)`, `bb-hosts(5)`, `xymonserver.cfg(5)`

14.3 HOBBIT-GHOSTS.CGI

xymon-ghosts.cgi - CGI program to view ghost clients

14.3.1 SYNOPSIS

xymon-ghosts.cgi

14.3.2 DESCRIPTION

xymon-ghosts.cgi is invoked as a CGI script via the `hobbit-ghosts.sh` CGI wrapper.

It generates a listing of the xymon clients that have reported data to the xymon server, but are not listed in the `bb-hosts(5)` file. Data from these clients - called “ghosts” - are ignored, since xymon does not know which webpage to present the data on.

The listing includes the hostname that the client reports with, the IP-address where the report came from, and how long ago the report arrived.

By far the most common reason for hosts showing up here is that the client uses a hostname without a DNS domain, but the `bb-hosts` file uses the hostname with the DNS domain. Or vice versa. You can then use a **CLIENT** setting in the `bb-hosts` file to match the two hostnames together.

14.3.3 OPTIONS

-env=FILENAME Loads the environment defined in FILENAME before executing the CGI script.

14.3.4 SEE ALSO

`bb-hosts(5)`, `xymonserver.cfg(5)`

14.4 TRIMHISTORY

14.4.1 NAME

`"/opt/bin/trimhistory"` - Remove old xymon history-log entries
 subsectionSYNOPSIS **trimhistory –cutoff=TIME [options]**

14.4.2 DESCRIPTION

The **trimhistory** tool is used to purge old entries from the xymon history logs. These logfiles accumulate information about all status changes that have occurred for any given service, host, or the entire xymon system, and is used to generate the event- and history-log webpages.

Purging old entries can be done while xymon is running, since the tool takes care not to commit updates to a file if it changes mid-way through the operation. In that case, the update is aborted and the existing logfile is left untouched.

Optionally, this tool will also remove logfiles from hosts that are no longer defined in the xymon *bb-hosts(5)* file. As an extension, even logfiles from services can be removed, if the service no longer has a valid status-report logged in the current xymon status.

14.4.3 OPTIONS

-cutoff=TIME This defines the cutoff-time when processing the history logs. Entries dated before this time are discarded. TIME is specified as the number of seconds since the beginning of the Epoch. This is easily generated by the GNU *date(1)* utility, e.g. the following command will trim history logs of all entries prior to Oct. 1st 2004:

```
trimhistory–cutoff='date+%s–date="1Oct2004'"
```

-outdir=DIRECTORY Normally, files in the BBHIST directory are replaced. This option causes trimhistory to save the shortened history logfiles to another directory, so you can verify that the operation works as intended. The output directory must exist.

-drop Causes trimhistory to delete files from hosts that are not listed in the *bb-hosts(5)* file.

-dropsvcs Causes trimhistory to delete files from services that are not currently tracked by xymon. Normally these files would be left untouched if only the host exists.

-droplogs Process the BBHISTLOGS directory also, and delete status-logs from events prior to the cut-off time. Note that this can dramatically increase the processing time, since there are often lots and lots of files to process.

-progress[=N] This will cause trimhistory to output a status line for every N history logs or status-log collections it processes, to indicate how far it has progressed. The default setting for N is 100.

-env=FILENAME Loads the environment from FILENAME before executing trimhistory.

-debug Enable debugging output.

14.4.4 FILES

\$BBHIST/allevents The eventlog of all events that have happened in xymon.

\$BBHIST/HOSTNAME The per-host eventlogs.

\$BBHIST/HOSTNAME.SERVICE The per-service eventlogs.

\$BBHISTLOGS/*/* The historical status-logs.

14.4.5 ENVIRONMENT VARIABLES

BBHIST The directory holding all history logs.

BBHISTLOGS The top-level directory for the historical status-log collections.

BBHOSTS The location of the bb-hosts file, holding the list of currently known hosts in xymon.

14.4.6 SEE ALSO

xymon(7), bb-hosts(5)

14.5 HOBBIT-ACKINFO.CGI

xymon-ackinfo.cgi - xymon CGI script to acknowledge alerts

14.5.1 SYNOPSIS

xymon-ackinfo.cgi

14.5.2 DESCRIPTION

xymon-ackinfo.cgi is invoked as a CGI script via the xymon-ackinfo.sh CGI wrapper.

xymon-ackinfo.cgi is used to acknowledge an alert on the xymon “Critical Systems” view, generated by the *xymon-nkview.cgi(1)* utility. This allows the staff viewing the Critical Systems view to acknowledge alerts with a “Level 1” alert, thereby removing the alert from the Critical Systems view.

Note that the Level 1 alert generated by the xymon-ackinfo.cgi utility does **NOT** stop alerts from being sent.

In a future version of xymon (after Xymon 4.2), this utility will also be used for acknowledging alerts at other levels.

14.5.3 OPTIONS

-level=NUMBER Sets the acknowledgment level. This is typically used to force a specific level of the acknowledgment, e.g. a level 1 acknowledge when called from the Critical Systems view.

-validity=TIME Sets the validity of the acknowledgment. By default this is taken from the CGI parameters supplied by the user.

-sender=STRING Logs STRING as the sender of the acknowledgment. By default, this is taken from the loginname of the webuser sending the acknowledgment.

-env=FILENAME Loads the environment defined in FILENAME before executing the CGI script.

-area=NAME Load environment variables for a specific area. NB: if used, this option must appear before any – env=FILENAME option.

-debug Enables debugging output.

14.5.4 ENVIRONMENT VARIABLES

BBHOME Used to locate the template files for the generated web pages.

QUERY_STRING Contains the parameters for the CGI script.

14.5.5 SEE ALSO

xymon-nkview.cgi(1), hobbit(7)

Chapter 15

Network Service Testing

15.1 BBTEST-NET

`bbtest-net` - xymon network test tool

15.1.1 SYNOPSIS

`bbtest-net -?`

`bbtest-net --help`

`bbtest-net --version`

`bbtest-net [options]`

(See the OPTIONS section for a description of the available commandline options).

15.1.2 DESCRIPTION

`bbtest-net(1)` handles the network tests of hosts defined in the xymon configuration file, `bb-hosts`. It is normally run at regular intervals by `xymonlaunch(8)` via an entry in the `xymonlaunch.cfg(5)` file.

`bbtest-net` does all of the normal tests of TCP-based network services (telnet, ftp, ssh, smtp, pop, imap) - i.e. all of the services listed as BBNETSVCS in `bbdef.sh`. For these tests, a completely new and very speedy service- checker has been implemented.

`bbtest-net` has built-in support for testing SSL-enabled protocols, e.g. imaps, pop3s, nntps, telnets, if SSL-support was enabled when configuring `bbgen`. The full list of known tests is found in the `bb-services(5)` file in `$BBHOME/etc/bb-services`.

In addition, it implements the “dns” and “dig” tests for testing DNS servers. This is done in the same way as `bb-network.sh` does it.

`bbtest-net` also implements a check for NTP servers - this test is called “ntp”. If you want to use it, you must define the `NTPDATE` environment variable to point at the location of your `ntpdate(1)` program.

Note: `bbtest-net` performs the connectivity test (ping) based on the hostname, unless the host is tagged with “testip” or the “–dns=ip” option is used. So the target of the connectivity test can be determined by your `/etc/hosts` file or DNS.

15.1.3 GENERAL OPTIONS

-timeout=N Determines the timeout (in seconds) for each service that is tested. For TCP tests (those from BBNETSVCS), if the connection to the service does not succeed within N seconds, the service is reported as being down. For HTTP tests, this is the absolute limit for the entire request to the webserver (the time needed to connect to the server, plus the time it takes the server to respond to the request). Default: 10 seconds

-conntimeout=N This option is deprecated, and will be ignored. Use the `-timeout` option instead.

-cmdtimeout=N This option sets a timeout for the external commands used for testing of NTP and RPC services, and to perform traceroute.

-concurrency=N Determines the number of network tests that run in parallel. Default is operating system dependent, but will usually be 256. If bbtest-net begins to complain about not being able to get a “socket”, try running bbtest-net with a lower value like 50 or 100.

-dns-timeout=N (default: 30 seconds) bbtest-net will timeout all DNS lookups after N seconds. Any pending DNS lookups are regarded as failed, i.e. the network tests that depend on this DNS lookup will report an error.

Note: If you use the `-no-ares` option, timeout of DNS lookups cannot be controlled by bbtest-net.

-dns-max-all=N Same as “`-dns-timeout=N`”. The “`-dns-max-all`” option is deprecated and should not be used.

-dns=[ip—only—standard] Determines how bbtest-net finds the IP addresses of the hosts to test. By default (the “standard”), bbtest-net does a DNS lookup of the hostname to determine the IP address, unless the host has the “testip” tag, or the DNS lookup fails.

With “`-dns=only`” bbtest-net will ONLY do the DNS lookup; if it fails, then all services on that host will be reported as being down.

With “`-dns=ip`” bbtest-net will never do a DNS lookup; it will use the IP address specified in *bb-hosts* for the tests. Thus, this setting is equivalent to having the “testip” tag on all hosts. Note that http tests will ignore this setting and still perform a DNS lookup for the hostname given in the URL; see the “*bbtest-net* tags for HTTP tests” section in *bb-hosts(5)*

-no-ares Disable the ARES resolver built into bbtest-net. This makes bbtest-net resolve hostnames using your system resolver function. You should only use this as a last resort if bbtest-net cannot resolve the hostnames you use in the normal way (via DNS or /etc/hosts). One reason for using this would be if you need to resolve hostnames via NIS/NIS+ (a.k.a. Yellow Pages).

The system resolver function does not provide a mechanism for controlling timeouts of the hostname lookups, so if your DNS or NIS server is down, bbtest-net can take a very long time to run. The `-dns-timeout` option is effectively disabled when using this option.

-dnslog=FILENAME Log failed hostname lookups to the file FILENAME. FILENAME should be a full pathname.

-report[=COLUMNNAME] With this option, bbtest-net will send a status message with details of how many hosts were processed, how many tests were generated, any errors that occurred during the run, and some timing statistics. The default columnname is “bbtest”.

-test-untagged When using the BBLOCATION environment variable to test only hosts on a particular network segment, bbtest-net will ignore hosts that do not have any “NET:x” tag. So only hosts that have a NET:\$BBLOCATION tag will be tested.

With this option, hosts with no NET: tag are included in the test, so that all hosts that either have a matching NET: tag, or no NET: tag at all are tested.

-frequenttestlimit=N Used with the *bbretest-net.sh(1)* bbgen extension. This option determines how long failed tests remain in the frequent-test queue. The default is 1800 seconds (30 minutes).

-timelimit=N Causes bbtest-net to generate a warning if the run-time of bbtest-net exceeds N seconds. By default N is set to the value of BBSLEEP, so a warning triggers if the network tests cannot complete in the time given for one cycle of the BBNET server. Apart from the warning, this option has no effect, i.e. it will not terminate bbtest-net prematurely. So to eliminate any such warnings, use this option with a very high value of N.

-huge=N Warn if the response from a TCP test is more than N bytes. If you see from the bbtest status report that you are transferring large amounts of data for your tests, you can enable this option to see which tests have large replies. Default: 0 (disabled).

-validity=N Make the test results valid for N minutes before they go purple. By default test results are valid for 30 minutes; if you run bbtest-net less often than that, the results will go purple before the next run of bbtest-net. This option lets you change how long the status is valid.

15.1.4 OPTIONS FOR TESTS OF THE SIMPLE TCP SERVICES

-checkresponse[=COLOR] When testing well-known services (e.g. FTP, SSH, SMTP, POP-2, POP-3, IMAP, NNTP and rsync), bbtest-net will look for a valid service-specific “OK” response. If another reponse is seen, this will cause the test to report a warning (yellow) status. Without this option, the response from the service is ignored.

The optional color-name is used to select a color other than yellow for the status message when the response is wrong. E.g. “-checkresponse=red” will cause a “red” status message to be sent when the service does not respond as expected.

-no-flags By default, bbtest-net sends some extra information in the status messages, called “flags”. These are used by bbgen e.g. to pick different icons for reversed tests when generating the xymon webpages. This option makes bbtest-net omit these flags from the status messages.

15.1.5 OPTIONS FOR THE PING TEST

Note: bbtest-net uses the program defined by the FPING environment to execute ping-tests - by default, that is the *xymonping(1)* utility. See *xymonserver:cfg(5)* for a description of how to customize this, e.g. if you need to run it with “sudo” or a similar tool.

-ping Enables bbtest-net’s ping test. The column name used for ping test results is defined by the PINGCOLUMN environment variable in *xymonserver.cfg(5)*.

If not specified, bbtest-net uses the CONNTEST environment variable to determine if it should perform the ping test or not. So if you prefer to use another tool to implement ping checks, either set the CONNTEST environment variable to false, or run bbtest-net with the “-noping”.

-noping Disable the connectivity test.

-trace

-notrace Enable/disable the use of traceroute when a ping-test fails. Performing a traceroute for failed ping tests is a slow operation, so the default is not to do any traceroute, unless it is requested on a per-host basis via the “trace” tag in the *bb-hosts(5)* entry for each host. The “-trace” option changes this, so the default becomes to run traceroute on all hosts where the ping test fails; you can then disable it on specific hosts by putting a “notrace” tag on the host-entry.

15.1.6 OPTIONS FOR HTTP (WEB) TESTS

-content=CONTENTTESTNAME Determines the name of the column xymon displays for content checks. The default is “content”. If you have used the “cont.sh” or “cont2.sh” scripts earlier, you may want to use “-content=cont” to report content checks using the same test name as these scripts do.

15.1.7 OPTIONS FOR SSL CERTIFICATE TESTS

-ssl=SSLCERTTESTNAME Determines the name of the column xymon displays for the SSL certificate checks. The default is “sslcert”.

-no-ssl Disables reporting of the SSL certificate check.

-sslwarn=N

-sslalarm=N Determines the number of days before an SSL certificate expires, where bbtest-net will generate a warning or alarm status for the SSL certificate column.

15.1.8 DEBUGGING OPTIONS

-no-update Don't send any status updates to the BBDISPLAY server. Instead, all messages are dumped to stdout.

-timing Causes bbtest-net to collect information about the time spent in different parts of the program. The information is printed on stdout just before the program ends. Note that this information is also included in the status report sent with the “-report” option.

-debug Dumps a bunch of status about the tests as they progress to stdout.

-dump[=before—=after—=both] Dumps internal memory structures before and/or after the tests have executed.

15.1.9 INFORMATIONAL OPTIONS

-help or -? Provide a summary of available commandline options.

-version Prints the version number of bbtest-net

-services Dump the list of defined TCP services bbtest-net knows how to test. Do not run any tests.

15.1.10 USING COOKIES IN WEB TESTS

If the file \$BBHOME/etc/cookies exist, cookies will be read from this file and sent along with the HTTP requests when checking websites. This file is in the Netscape Cookie format, see http://www.netscape.com/newsref/std/cookie_spec.html for details on this format. The *curl(1)* utility can output a file in this format if run with the “-cookie-jar FILENAME” option.

15.1.11 ABOUT SSL CERTIFICATE CHECKS

When bbtest-net tests services that use SSL- or TLS-based protocols, it will check that the server certificate has not expired. This check happens automatically for https (secure web), pop3s, imaps, nntps and all other SSL-enabled services (except ldap, see LDAP TESTS below).

All certificates found for a host are reported in one status message.

Note: On most systems, the end-date of the certificate is limited to Jan 19th, 2038. If your certificate is valid after this date, bbtest-net will report it as valid only until Jan 19, 2038. This is due to limitations in your operating system C library.

15.1.12 LDAP TESTS

ldap testing can be done in two ways. If you just put an “ldap” or “ldaps” tag in bb-hosts, a simple test is performed that just verifies that it is possible to establish a connection to the port running the ldap service (389 for ldap, 636 for ldaps).

Instead you can put an LDAP URI in bb-hosts. This will cause bbtest-net to initiate a full-blown LDAP session with the server, and do an LDAP search for the objects defined by the URI. This requires that bbtest-net was built with LDAP support, and relies on an existing LDAP library to be installed. It has been tested with OpenLDAP 2.0.26 (from Red Hat 9) and 2.1.22. The Solaris 8 system ldap library has also been confirmed to work for un-encrypted (plain ldap) access.

The format of LDAP URI's is defined in RFC 2255. LDAP URLs look like this:

```
\textbf{\{ldap://\}}
\emph{hostport}
\textbf{/}
\emph{dn}
[\textbf{?}]
\emph{attrs}
[\textbf{?}]
\emph{scope}
[\textbf{?}]
\emph{filter}
[\textbf{?}]
\emph{exts}
]]]
```

where:

```
\emph{hostport}
is a host name with an optional ":portnumber"
\emph{dn}
is the search base
\emph{attrs}
is a comma separated list of attributes to request
\emph{scope}
is one of these three strings:
base one sub (default=base)
\emph{filter}
is filter
\emph{exts}
are recognized set of LDAP and/or API extensions.
```

Example:

```
ldap://ldap.example.net/dc=example,dc=net?cn,sn?sub?(cn=*)
```

All “bind” operations to LDAP servers use simple authentication. Kerberos and SASL are not supported. If your LDAP server requires a username/password, use the “ldaplogin” tag to specify this, cf. *bb-hosts(5)*. If no user-name/password information is provided, an anonymous bind will be attempted.

SSL support requires both a client library and an LDAP server that support LDAPv3; it uses the LDAP “START-TLS” protocol request after establishing a connection to the standard (non-encrypted) LDAP port (usually port 389). It has only been tested with OpenSSL 2.x, and probably will not work with any other LDAP library.

The older LDAPv2 experimental method of tunnelling normal LDAP traffic through an SSL connection - ldaps, running on port 636 - is not supported, unless someone can explain how to get the OpenLDAP library to support it. This method was never formally described in an RFC, and implementations of it are non-standard.

For a discussion of the various ways of running encrypted ldap, see <http://www.openldap.org/lists/openldap-software/200305/msg00079.html> <http://www.openldap.org/lists/openldap-software/200305/msg00084.html> <http://www.openldap.org/lists/openldap-software/200201/msg00042.html> <http://www.openldap.org/lists/openldap-software/200206/msg00387.html>

When testing LDAP URI's, all of the communications are handled by the ldap library. Therefore, it is not possible to obtain the SSL certificate used by the LDAP server, and it will not show up in the “sslcert” column.

15.1.13 USING MULTIPLE NETWORK TEST SYSTEMS

If you have more than one system running network tests - e.g. if your network is separated by firewalls - then it is problematic to maintain multiple bb-hosts files for each of the systems. bbtest-net supports the NET:location tag in *bb-hosts(5)* to distinguish between hosts that should be tested from different network locations. If you set the environment variable BBLOCATION e.g. to "dmz" before running bbtest-net, then it will only test hosts that have a "NET:dmz" tag in bb-hosts. This allows you to keep all of your hosts in the same bb-hosts file, but test different sets of hosts by different BBNET systems.

15.1.14 BBTEST-NET INTERNALS

bbtest-net first reads the bb-services file to see which network tests are defined. It then scans the bb-hosts file, and collects information about the TCP service tests that need to be tested. It picks out only the tests that were listed in the bb-services file, plus the "dns", "dig" and "ntp" tests - those tests that bb-network.sh would normally use the "bbnet" tool to test.

It then runs two tasks in parallel: First, a separate process is started to run the "xymonping" tool for the connectivity tests. While xymonping is busy doing the "ping" checks, bbtest-net runs all of the TCP-based network tests.

All of the TCP-based service checks are handled by a connection tester written specifically for this purpose. It uses only standard Unix-style network programming, but relies on the Unix "select(2)" system-call to handle many simultaneous connections happening in parallel. Exactly how many parallel connections are being used depends on your operating system - the default is FD_SETSIZE/4, which amounts to 256 on many Unix systems.

You can choose the number of concurrent connections with the "--concurrency=N" option to bbtest-net.

Connection attempts timeout after 10 seconds - this can be changed with the "--timeout=N" option.

Both of these settings play a part in deciding how long the testing takes. A conservative estimate for doing N TCP tests is:

$$(1+(N/\text{concurrency}))*\text{timeout}$$

In real life it will probably be less, as the above formula is for every test to require a timeout. Since the most normal use of BB is to check for services that are active, you should have a lot less timeouts.

The "ntp" and "rpcinfo" checks rely on external programs to do each test. Thus, they perform only marginally better than the standard bb-network.sh script.

15.1.15 ENVIRONMENT VARIABLES

BBLOCATION Defines the network segment where bbtest-net is currently running. This is used to filter out only the entries in the *bb-hosts(5)* file that have a matching "NET:LOCATION" tag, and execute the tests for only those hosts.

BMAXMSGSPERCOMBO Defines the maximum number of status messages that can be sent in one combo message. Default is 0 - no limit.

In practice, the maximum size of a single xymon message sets a limit - the default value for the maximum message size is 32 KB, but that will easily accomodate 100 status messages per transmission. So if you want to experiment with this setting, I suggest starting with a value of 10.

BBSLEEPBETWEENMSGS Defines a delay (in microseconds) after each message is transmitted to the BBDISPLAY server. The default is 0, i.e. send the messages as fast as possible. This gives your BBDISPLAY server some time to process the message before the next message comes in. Depending on the speed of your BBDISPLAY server, it may be necessary to set this value to half a second or even 1 or 2 seconds. Note that the value is specified in MICROseconds, so to define a delay of half a second, this must be set to the value "500000"; a delay of 1 second is achieved by setting this to "1000000" (one million).

FPING Command used to run the *xymonping(1)* utility. Used by bbtest-net for connectivity (ping) testing. See *xymonserver.cfg(5)* for more information about how to customize the program that is executed to do ping tests.

TRACEROUTE Location of the *traceroute(8)* utility, or an equivalent tool e.g. *mtr(8)*. Optionally used when a connectivity test fails to pinpoint the network location that is causing the failure.

NTPDATE Location of the *ntpdate(1)* utility. Used by bbtest-net when checking the “ntp” service.

RPCINFO Location of the *rpcinfo(8)* utility. Used by bbtest-net for the “rpc” service checks.

15.1.16 FILES

/server/etc/bb-services (xymon) This file contains definitions of TCP services that bbtest-net can test. Definitions for a default set of common services is built into bbtest-net, but these can be overridden or supplemented by defining services in the bb-services file. See *bb-services(5)* for details on this file.

\$BBHOME/etc/netrc - authentication data for password-protected webs If you have password-protected sites, you can put the usernames and passwords for these here. They will then get picked up automatically when running your network tests. This works for web-sites that use the “Basic” authentication scheme in HTTP. See *ftp(1)* for details - a sample entry would look like this

machinewww.acme.comloginfredpasswordWilma1 Note that the machine-name must be the name you use in the <http://machinename/> URL setting - it need not be the one you use for the system-name in xymon.

\$BBHOME/etc/cookies This file may contain website cookies, in the Netscape HTTP Cookie format. If a website requires a static cookie to be present in order for the check to complete, then you can add this cookie to this file, and it will be sent along with the HTTP request. To get the cookies into this file, you can use the “curl –cookie-jar FILE” to request the URL that sets the cookie.

\$BBTMP/*.status - test status summary Each time bbtest-net runs, if any tests fail (i.e. they result in a red status) then they will be listed in a file name TESTNAME.[LOCATION].status. The LOCATION part may be null. This file is used to determine how long the failure has lasted, which in turn decides if this test should be included in the tests done by *bbretest-net.sh(1)*

It is also used internally by bbtest-net when determining the color for tests that use the “badconn” or “badTEST-NAME” tags.

\$BBTMP/frequenttests.[LOCATION] This file contains the hostnames of those hosts that should be retested by the *bbretest-net.sh(1)* test tool. It is updated only by bbtest-net during the normal runs, and read by *bbretest-net.sh*.

15.1.17 SEE ALSO

bb-hosts(5), *bb-services(5)*, *xymonserver.cfg(5)*, *hobbitping(1)*, *curl(1)*, *ftp(1)*, *fping(1)*, *ntpdate(1)*, *rpcinfo(8)*

15.2 BB-SERVICES

bb-services - Configuration of TCP network services

15.2.1 SYNOPSIS

\$BBHOME/etc/bb-services

15.2.2 DESCRIPTION

bb-services contains definitions of how *bbtest-net(1)* should test a TCP-based network service (i.e. all common network services except HTTP and DNS). For each service, a simple dialogue can be defined to check that the service is functioning normally, and optional flags determine if the service has e.g. a banner or requires SSL- or telnet-style handshaking to be tested.

15.2.3 FILE FORMAT

bb-services is a text file. A simple service definition for the SMTP service would be this:

```
[smtp]
send"mail\r\nquit\r\n"
expect"220"
optionsbanner
```

This defines a service called “smtp”. When the connection is first established, bbtest-net will send the string “mail\r\nquit\r\n” to the service. It will then expect a response beginning with “220”. Any data returned by the service (a so-called “banner”) will be recorded and included in the status message.

The full set of commands available for the bb-services file are:

[NAME] Define the name of the TCP service, which will also be the column-name in the resulting display on the test status. If multiple tests share a common definition (e.g. ssh, ssh1 and ssh2 are tested identically), you may list these in a single “[ssh—ssh1—ssh2]” definition, separating each service-name with a pipe-sign.

send STRING

expect STRING Defines the strings to send to the service after a connection is established, and the response that is expected. Either of these may be omitted, in which case *bbtest-net(1)* will simply not send any data, or match a response against anything.

The send- and expect-strings use standard escaping for non-printable characters. “\r” represents a carriage-return (ASCII 13), “\n” represents a line-feed (ASCII 10), “\t” represents a TAB (ASCII 8). Binary data is input as “\xNN” with NN being the hexadecimal value of the byte.

port NUMBER Define the default TCP port-number for this service. If no portnumber is defined, *bbtest-net(1)* will attempt to lookup the portnumber in the standard /etc/services file.

options option1[,option2 [,option3]] Defines test options. The possible options are

```
banner-includereceivedatainthestatusmessage
ssl-serviceusesSSLsopermanSSLhandshake
telnet-serviceistelnet,soexchangetelnetoptions
```

15.2.4 FILES

\$BBHOME/etc/bb-services

15.2.5 SEE ALSO

`bbtest-net(1)`

15.3 HOBBITFETCH

15.3.1 NAME

xymonfetch - fetch client data from passive clients

15.3.2 SYNOPSIS

xymonfetch [-server=HOBBIT.SERVER.IP] [options]

15.3.3 DESCRIPTION

This utility is used to collect data from xymon clients.

Normally, xymon clients will themselves take care of sending all of their data directly to the xymon server. In that case, you do not need this utility at all. However, in some network setups clients may be prohibited from establishing a connection to an external server such as the xymon server, due to firewall policies. In such a setup you can configure the client to store all of the client data locally by enabling the *msgcache(8)* utility on the client, and using **xymonfetch** on the xymon server to collect data from the clients.

xymonfetch will only collect data from clients that have the **pulldata** tag listed in the *bb-hosts(5)* file. The IP-address listed in the bb-hosts file must be correct, since this is the IP-address where xymonfetch will attempt to contact the client. If the msgcache daemon is running on a non-standard IP-address or portnumber, you can specify the portnumber as in **pulldata=192.168.1.2:8084** for contacting the msgcache daemon using IP 192.168.1.2 port 8084. If the IP-address is omitted, the default IP in the bb-hosts file is used. If the port number is omitted, the portnumber from the BBPORT setting in *xymonserver.cfg(5)* is used (normally, this is port 1984).

15.3.4 OPTIONS

-server=HOBBIT.SERVER.IP Defines the IP address of the xymon server where the collected client messages are forwarded to. By default, messages are sent to the loopback address 127.0.0.1, i.e. to a xymon server running on the same host as xymonfetch.

-interval=N Sets the interval (in seconds) between polls of a client. Default: 60 seconds.

-id=N Used when you have a setup with multiple xymon servers. In that case, you must run xymonfetch on each of the xymon servers, with xymonfetch instance using a different value of N. This allows several xymon servers to pick up data from the clients running msgcache, and msgcache can distinguish between which messages have already been forwarded to which server.

N is a number in the range 1-31.

-log-interval=N Limit how often xymonfetch will log problems with fetching data from a host, in seconds. Default: 900 seconds (15 minutes). This is to prevent a host that is down or where msgcache has not been started from flooding the xymonfetch logs. Note that this is ignored when debugging is enabled.

-debug Enable debugging output.

15.3.5 SEE ALSO

msgcache(8), *xymond(8)*, *hobbit(7)*

15.4 bbcmd

“/opt/bin/bbcmcmd” - Run a xymon command with environment set

15.4.1 SYNOPSIS

bbcmcmd –env=ENVFILE COMMAND

15.4.2 DESCRIPTION

bbcmcmd(1) is a utility that can setup the xymon environment variables as defined in a *xymonlaunch(8)* compatible environment definition file, and then execute a command with this environment in place. It is mostly used for testing extension scripts or in other situations where you need to run a single command with the environment in place.

The “–env=ENVFILE” option points bbcmcmd to the file where the environment definitions are loaded from.

COMMAND is the command to execute after setting up the environment.

If you want to run multiple commands, it is often easiest to just use “sh” as the COMMAND - this gives you a sub-shell with the environment defined globally.

15.4.3 SEE ALSO

xymonlaunch(8), *hobbit(7)*

15.5 CLIENTLAUNCH.CFG

clientlaunch.cfg - Task definitions for the xymonlaunch utility

15.5.1 SYNOPSIS

`xymon/client/etc/clientlaunch.cfg`

15.5.2 DESCRIPTION

The clientlaunch.cfg file holds the list of tasks that xymonlaunch runs on a xymon client. This is typically just the Xymon client itself, but you can add custom test scripts here and have them executed regularly by the xymon scheduler.

15.5.3 FILE FORMAT

See the *xymonlaunch.cfg(5)* description.

15.5.4 SEE ALSO

`xymonlaunch(8)`, `hobbit(7)`

Chapter 16

Combination Tests

16.1 BBCOMBOTEST

`bbcombotest` - xymon combination test tool

16.1.1 SYNOPSIS

`bbcombotest –help bbcombotest –version bbcombotest [-debug] [-quiet]`

16.1.2 DESCRIPTION

`bbcombotest` is a xymon extension script that runs on the BBDISPLAY server. It combines the results of one or more of the normal xymon test results into a combined test result, using standard arithmetic og logical operators.

The resulting tests are sent to the xymon display server as any normal test - so all of the standard xymon functions (history, statistics etc.) are available for the combined tests.

The tool was born from the need to monitor systems with built-in redundancy and automatic failover - e.g. load-balanced web servers. But other uses are possible.

16.1.3 OPTIONS

-quiet Normally, the test status sent by `bbcombotest` includes information about the underlying test results used to determine the current value of the combined test. “`-quiet`” eliminates this information from the test status page.

-debug Provide debugging output for use in troubleshooting problems with `bbcombotest`.

-no-update Dont send any status messages - instead, the result of the combotests is simply dumped to stdout. Useful for debugging.

16.1.4 FILES

`“/etc/opt/xymonserver42/bbcombotest .cfg”` Configuration file for `bbcombotest`, where the combined tests are defined

`“/etc/opt/xymonserver42/xymonlaunch .cfg”` Configuration file controlling when `bbcombotest` is run.

16.1.5 SEE ALSO

`bbcombotest.cfg(5)`, `bb-hosts(5)`, `xymonserver.cfg(5)`, `hobbitlaunch.cfg(5)`

16.2 HOBBITGRAPH.CGI

16.2.1 NAME

xymongraph.cgi - CGI to generate xymon trend graphs

16.2.2 SYNOPSIS

xymongraph [options]

16.2.3 DESCRIPTION

xymongraph.cgi is invoked as a CGI script via the xymongraph.sh CGI wrapper.

xymongraph.cgi is passed a QUERY_STRING environment variable with the following parameters:

host Name of the host to generate a graph for

service Name of the service to generate a graph for

disp Display-name of the host, used on the generated graphs instead of hostname.

graph Can be “hourly”, “daily”, “weekly” or “monthly” to select the time period that the graph covers.

first Used to split multi-graphs into multiple graphs. This causes xymongraph.cgi to generate only the graphs starting with the “first’th” graph and continuing for “count”.

count Number of graphs in a multi-graph.

upper Set the upper limit of the graph. See *rrdgraph(1)* for a description of the “-u” option.

lower Set the lower limit of the graph. See *rrdgraph(1)* for a description of the “-l” option.

graph_start Set the starttime of the graph. This is used in zoom-mode.

graph_end Set the end-time of the graph. This is used in zoom-mode.

action=menu Generate an HTML page with links to 4 graphs, representing the hourly, weekly, monthly and yearly graphs. Doesn’t actually generate any graphs, only the HTML that links to the graphs.

action=selzoom Generate an HTML page with link to single graph, and with JavaScript code that lets the user select part of the graph for a zoom-operation. Doesn’t actually generate graph, only the HTML that links to the graph.

action=view Generate a single graph image.

16.2.4 OPTIONS

-config=FILENAME Loads the graph configuration file from FILENAME. If not specified, the file “/etc/opt/xmonserver42/xymongraph.cfg” is used. See the *hobbitgraph.cfg(5)* for details about this file.

-env=FILENAME Loads the environment settings defined in FILENAME before executing the CGI.

-rrddir=DIRECTORY The top-level directory for the RRD files. If not specified, the directory given by the BBR-RDS environment is used.

-save=FILENAME Instead of returning the image via the CGI interface (i.e. on stdout), save the generated image to FILENAME.

-debug Enable debugging output.

16.2.5 ENVIRONMENT

QUERY_STRING Provided by the webserver CGI interface, this decides what graph to generate.

16.2.6 FILES

xymongraph.cfg: The configuration file determining how graphs are generated from RRD files.

16.2.7 SEE ALSO

xymongraph.cfg(5), hobbit(7), rrdtool(1)

16.3 BBCOMBOTEST.CFG

16.3.1 NAME

`bbcombotest.cfg` - Configuration of bbcombotest tool

16.3.2 SYNOPSIS

`"/etc/opt/xymonserver42/bbcombotest.cfg"`

16.3.3 DESCRIPTION

`bbcombotest(1)` uses it's own configuration file, `"/etc/opt/xymonserver42/bbcombotest.cfg"` Each line in this file defines a combined test.

16.3.4 FILE FORMAT

Each line of the file defines a new combined test. Blank lines and lines starting with a hash mark (#) are treated as comments and ignored.

The configuration file uses the hostnames and testnames that are already used in your xymon bb-hosts file. These are then combined using normal logical operators - “`—`” for “or”, “`&&`” for “and” etc.

A simple test - e.g. “`Web1.http`” - results in the value “1” if the “`http`” test for server “`Web1`” is green, yellow or clear. It yields the value “0” if it is red, purple or blue.

Apart from the logical operations, you can also do integer arithmetic and comparisons. E.g. the following is valid:

`WebCluster.http = (Web1.http + Web2.http + Web3.http) >= 2`

This test is green if two or more of the http tests for Web1, Web2 and Web3 are green.

The full range of operators are:

`+Add -Subtract *Multiply /Divide %Modulo —Bit-wise”or” &Bit-wise”and” —Logical”or” &&Logical”and”`
`>Greaterthan <Lessthan >=Greaterthanorequal <=Lessthanorequal ==Equal`

There is currently no support for a “not” operator. If you need it, use the transcription “`(host.test == 0)`” instead of “`!host.test`”.

NB: All operators have EQUAL PRECEDENCE. If you need something evaluated in a specific order, use parentheses to group the expressions together.

If the expression comes out as “0”, the combined test goes red. If it comes out as non-zero, the combined test is green.

Note: If the expression involves hostnames with a character that is also an operator - e.g. if you have a host “`t1-router-newyork.foo.com`” with a dash in the hostname - then the operator-character must be escaped with a backslash ‘\’ in the expression, or it will be interpreted as an operator. E.g. like this:

`nyc.conn=(t1\‐router\‐nyc.conn——backup\‐router\‐nyc.conn)`

16.3.5 EXAMPLE

```
WebCluster.http = (Web1.http — Web2.http) AppSrvCluster.procs = (AppSrv1.conn && AppSrv1.procs) —
(AppSrv2.conn && AppSrv2.procs) Customer.cluster = WebCluster.http && AppSrvCluster.procs
```

The first line defines a new test, with hostname “`WebCluster`” and the columnname “`http`”. It will be green if the `http` test on either the “`Web1`” or the “`Web2`” server is green.

The second line defines a “`procs`” test for the “`AppSrvCluster`” host. Each of the `AppSrv1` and `AppSrv2` hosts is checked for “`conn`” (ping) and their “`procs`” test. On each host, both of these must be green, but the combined test is green if that condition is fulfilled on just one of the hosts.

The third line uses the two first tests to build a “double combined” test, defining a test that shows the overall health of the system.

16.3.6 FILES

`“/etc/opt/xymonserver42/bbcombotest.cfg”`

16.3.7 SEE ALSO

`bbcombotest(1)`

16.4 HOBBITGRAPH.CGI

16.4.1 NAME

xymongraph.cgi - CGI to generate xymon trend graphs

16.4.2 SYNOPSIS

xymongraph [options]

16.4.3 DESCRIPTION

xymongraph.cgi is invoked as a CGI script via the xymongraph.sh CGI wrapper.

xymongraph.cgi is passed a QUERY_STRING environment variable with the following parameters:

host Name of the host to generate a graph for

service Name of the service to generate a graph for

disp Display-name of the host, used on the generated graphs instead of hostname.

graph Can be “hourly”, “daily”, “weekly” or “monthly” to select the time period that the graph covers.

first Used to split multi-graphs into multiple graphs. This causes xymongraph.cgi to generate only the graphs starting with the “first”th graph and continuing for “count”.

count Number of graphs in a multi-graph.

upper Set the upper limit of the graph. See *rrdgraph(1)* for a description of the “-u” option.

lower Set the lower limit of the graph. See *rrdgraph(1)* for a description of the “-l” option.

graph_start Set the starttime of the graph. This is used in zoom-mode.

graph_end Set the end-time of the graph. This is used in zoom-mode.

action=menu Generate an HTML page with links to 4 graphs, representing the hourly, weekly, monthly and yearly graphs. Doesn’t actually generate any graphs, only the HTML that links to the graphs.

action=selzoom Generate an HTML page with link to single graph, and with JavaScript code that lets the user select part of the graph for a zoom-operation. Doesn’t actually generate graph, only the HTML that links to the graph.

action=view Generate a single graph image.

16.4.4 OPTIONS

-config=FILENAME Loads the graph configuration file from FILENAME. If not specified, the file “/etc/opt/xmonserver42/xymongraph.cfg” is used. See the *hobbitgraph.cfg(5)* for details about this file.

-env=FILENAME Loads the environment settings defined in FILENAME before executing the CGI.

-rrddir=DIRECTORY The top-level directory for the RRD files. If not specified, the directory given by the BBR-RDS environment is used.

-save=FILENAME Instead of returning the image via the CGI interface (i.e. on stdout), save the generated image to FILENAME.

-debug Enable debugging output.

16.4.5 ENVIRONMENT

QUERY_STRING Provided by the webserver CGI interface, this decides what graph to generate.

16.4.6 FILES

xymongraph.cgi: The configuration file determining how graphs are generated from RRD files.

16.4.7 SEE ALSO

xymongraph.cfg(5), hobbit(7), rrdtool(1)

16.5 BB-SNAPSHOT.CGI

16.5.1 NAME

bb-snapshot.cgi - CGI program to rebuild the xymon webpages for a specific point in time.

16.5.2 SYNOPSIS

bb-snapshot.cgi

16.5.3 DESCRIPTION

bb-snapshot.cgi is invoked as a CGI script via the **bb-snapshot.sh** CGI wrapper. It rebuilds the xymon web pages to the look they had at a particular point in time, based upon the historical information logged about events.

bb-snapshot.cgi is passed a **QUERY_STRING** environment variable with the following parameters:

mon(Startmonthofthesnapshot) day(Startday-of-monthofthesnapshot) yr(Startyearofthesnapshot) hour(Starthourofthesnapshot)
min(Startminuteofthesnapshot) sec(Startsecondofthesnapshot)

The “month” parameters must be specified as the three-letter english month name abbreviation: Jan, Feb, Mar ...

“day” must be in the range 1..31; “yr” must be specified including century (e.g. “2003”). “hour” must be specified using a 24-hour clock.

All of the processing involved in generating the report is done by invoking **bbgen(1)** with the proper “–snapshot” option.

16.5.4 OPTIONS

–env=FILENAME Load environment from FILENAME before executing the CGI.

bbgen-options All options except “–env” are passed on to the **bbgen(1)** program building the snapshot files.

16.5.5 ENVIRONMENT VARIABLES

BBGENSNAPOPTS bbgen options passed by default to the **bb-snapshot.cgi** script. This happens in the **bb-snapshot.sh** CGI wrapper script.

BBHOME Home directory of the xymon server files

BBSNAP Directory where generated snapshots are stored. This directory must be writable by the userid executing the CGI script, typically “www”, “apache” or “nobody”. Default: \$BBHOME/www/snap/

BBSNAPURL The URL prefix to use when accessing the reports via a browser. Default: \$BBWEB/snap

16.5.6 SEE ALSO

bbgen(1), **bb-hosts(5)**, **xymonserver.cfg(5)**

16.6 HOBBITD_SAMPLE

16.6.1 NAME

xymond_sample - example of a hobbitd worker module

16.6.2 SYNOPSIS

xymond_channel -channel=status hobbitd_sample [options]

16.6.3 DESCRIPTION

xymond_sample is a worker module for hobbitd, and as such it is normally run via the *xymond_channel(8)* program. It receives messages from xymond via stdin, and simply displays these on stdout. It can be used with all types of xymond channels.

xymond_sample is not designed to actually run, except as a demonstration. The purpose of this tool is to show how xymond worker modules can be implemented to handle different tasks that need to hook into the xymond processing.

16.6.4 OPTIONS

-timeout=N Read messages with a timeout of N seconds.

-debug Enable debugging output.

16.6.5 SEE ALSO

xymond_channel(8), *hobbitd(8)*, *hobbit(7)*

16.7 BB-CSVINFO.CGI

16.7.1 NAME

bb-csvinfo.cgi - CGI program to show host information from a CSV file

16.7.2 SYNOPSIS

bb-csvinfo.cgi

16.7.3 DESCRIPTION

bb-csvinfo.cgi is invoked as a CGI script via the `bb-csvinfo.sh` CGI wrapper. Based on the parameters it receives, it searches a comma-separated file for the matching host, and presents the information found as a table.

`bb-csvinfo.cgi` is passed a `QUERY_STRING` environment variable with the following parameters:

`key(stringtosearchfor,typicallyhostname) column(columnnumbertosearch-default0) db(nameoftheCSVdatabasefilein$BBHOME/etc delimiter(delimitercharacterforcolumns,defaultsemi-colon)`

CSV files are easily created from e.g. spreadsheets, by exporting them in CSV format. You should have one host per line, with the first line containing the column headings. Despite their name, the default delimiter for CSV files is the semi-colon - if you need a different delimiter, invoke `bb-csvinfo.cgi` with the “`delimiter=<character>`” in the query string.

16.7.4 Example usage

This example shows how you can use the `bb-csvinfo` CGI. It assumes you have a CSV-formatted file with information about the hosts stored as “`/etc/opt/xymonserver42/hostinfo.csv`”, and the hostname is in the first column of the file.

Use with the `bbgen -docurl` The `-docurl` option to `bbgen(1)` sets up all of the hostnames on your xymon webpages to act as links to a CGI script. To invoke the `bb-csvinfo` CGI script, run `bbgen` with the option

`-docurl=/cgi-bin/bb-csvinfo.sh?db=hostinfo.csv&key=%s`

16.7.5 SEE ALSO

`bb-hosts(5)`, `xymonserver.cfg(5)`, `bbgen(1)`

16.8 HOBBIT-HOSTGRAPHS.CGI

xymon-hostgraphs.cgi - CGI program to show multiple graphs

16.8.1 SYNOPSIS

xymon-hostgraph.cgi

16.8.2 DESCRIPTION

xymon-hostgraph.cgi is invoked as a CGI script via the hobbit-hostgraph.sh CGI wrapper.

If no parameters are provided when invoked, it will present a form where the user can select a time period, one or more hosts, and a set of graphs.

The parameters selected by the user are passed to a second invocation of xymon-hostgraph.cgi, and result in a webpage showing a list of graph images based on the trend data stored about the hosts.

If multiple graph-types are selected, xymon-hostgraph.cgi will display a list of graphs, with one graph per type.

If multiple hosts are selected, xymon-hostgraph.cgi will attempt to display a multi-host graph for each type where the graphs for all hosts are overlayed in a single image, allowing for easy comparison of the hosts.

The hostlist uses the PAGEPATH cookie provided by xymon webpages to select the list of hosts to present. Only the hosts visible on the page where xymon-hostgraph.cgi is invoked from will be visible.

The resulting graph page can be bookmarked, but the bookmark also fixates the time period shown.

16.8.3 OPTIONS

-env=FILENAME Loads the environment defined in FILENAME before executing the CGI script.

16.8.4 BUGS

This utility is experimental. It may change in a future release of xymon.

It is possible for the user to select graphs which do not exist. This results in broken image links.

The set of graph-types is fixed in the server/web/hostgraphs_form template and does not adjust to which graphs are available.

If the tool is invoked directly, all hosts defined in xymon will be listed.

16.8.5 SEE ALSO

bb-hosts(5), **xymonserver.cfg(5)**

16.9 BB-DATEPAGE.CGI

`bb-datepage.cgi` - xymon CGI script to view pre-built reports by date

16.9.1 SYNOPSIS

`bb-datepage.cgi?type=day,week,month -url=URLPREFIX [options]`

16.9.2 DESCRIPTION

bb-datepage.cgi is invoked as a CGI script via the `bb-datepage.sh` CGI wrapper.

`bb-datepage.cgi` is passed a `QUERY_STRING` environment variable with the type of time-selection that is desired: Either “day”, “week” or “month” can be requested. It will then generate a web form with appropriate day/week/month selection boxes, and based on the users’ selection a resulting url is built from the `URLPREFIX` and the time selection. The browser is then redirected to this URL.

The URL is constructed from the `URLPREFIX`, the type-parameter, the value of the “pagepath” or “host” cookie, and the users’ selection as follows:

type=day The final URL is `URLPREFIX/daily/YEAR/MONTH/DAY/PAGEPATH`.

type=week The final URL is `URLPREFIX/weekly/YEAR/WEEK/PAGEPATH`.

type=month The final URL is `URLPREFIX/monthly/YEAR/MONTH/PAGEPATH`.

`YEAR` is the full year (4 digits, including century). `MONTH` is the two-digit number of the month (01..12).

`DAY` is the number of the day in the month (01..31). `WEEK` is the ISO 8601:1988 week-number (01..53).

`PAGEPATH` is the current value of the “pagepath” cookie if set; if it is not set but the “host” cookie is set, then this host is looked up in the `bb-hosts` file and the page where this host is found is used for `PAGEPATH`. These two cookies are set by the default web-header templates supplied with xymon.

16.9.3 OPTIONS

-url=URLPREFIX This specifies the initial part of the final URL. This option is required.

-hfile=FILENAME Specifies the template files (from `$BBHOME/web/`) to use. The default is “`-hfile=report`”.

-color=COLOR Sets the background color of the generated webpage. The default is blue.

-env=FILENAME Loads the environment defined in `FILENAME` before executing the CGI script.

-debug Enables debugging output.

\$BBHOME/web/report_form.daily HTML form template for the date selection form when `type=daily`.

\$BBHOME/web/report_form.weekly HTML form template for the date selection form when `type=weekly`.

\$BBHOME/web/report_form.monthly HTML form template for the date selection form when `type=monthly`.

\$BBHOME/web/report_header HTML header file for the generated web page

\$BBHOME/web/report_footer HTML footer file for the generated web page

16.9.4 ENVIRONMENT VARIABLES

BBHOME Used to locate the template files for the generated web pages.

QUERY_STRING Contains the parameters for the CGI script.

16.9.5 SEE ALSO

`bbgen(1)`, `bb-hosts(5)`, `xymonserver.cfg(5)`

16.10 BBDIGEST

`bbdigest` - calculate message digests

16.10.1 SYNOPSIS

`bbdigest md5—sha1—rmd160 [filename]`

16.10.2 DESCRIPTION

bbdigest(1) is a utility to calculate message digests for a file or document. It is used when defining HTTP- or FTP-based content checks, where *bbtest-net(1)* checks that a URL returns a specific document; instead of having to compare the entire document, the comparison is done against a pre-computed message digest value using the MD5, RIPEMD160, SHA1 or any of the SHA2 (SHA-512, SHA-256, SHA-384, SHA-224) message digest algorithms.

The optional **filename** parameter is the input file whose message digest should be calculated; if no filename is given, the data is read from standard input.

`bbdigest` outputs a string containing the digest algorithm and the computed message digest. This is in a format suitable for use in the *bb-hosts(5)* definition of a content check.

16.10.3 EXAMPLE

```
$ bbdigestmd5index.html md5:88b81b110a85c83db56a939caa2e2cf6
$ curl-shttp://www.foo.com/—bbdigestsha1 sha1:e5c69784cb971680e2c7380138e04021a20a45a2
```

16.10.4 SEE ALSO

`bbtest-net(1)`, `bb-hosts(5)`

16.11 HOBBITLAUNCH.CFG

xymonlaunch.cfg - Task definitions for the hobbitlaunch utility

16.11.1 SYNOPSIS

`"/etc/opt/xymonserver42/xymonlaunch.cfg"`

16.11.2 DESCRIPTION

The xymonlaunch.cfg file holds the list of tasks that hobbitlaunch runs to perform all of the tasks needed by the xymon monitor.

16.11.3 FILE FORMAT

A task is defined by a **key** , a **command** , and optionally also **interval** , **environment** , and **logfile** .

Blank lines and lines starting with a hash mark (#) are treated as comments and ignored. Long lines can be broken up by putting a backslash at the end of the line and continuing the entry on the next line.

An entry looks like this:

```
[xymond]
ENVFILE/usr/local/xymon/server/etc/hobbitserver.cfg
CMD/usr/local/xymon/server/bin/hobbitd
[updateweb]
ENVFILE/usr/local/xymon/server/etc/hobbitserver.cfg
CMD/usr/local/xymon/server/bin/bbgen
NEEDSxymond
GROUPwebupdates
INTERVAL5m
LOGFILE/var/log/xymon/updateweb.log
```

The **key** is enclosed in angle brackets, and must be unique for each task. You can choose your key-names as you like, they are only used internally in xymonlaunch to identify each task.

The **command** is defined by the CMD keyword. This is the full command including any options you want to use for this task. This is required for all tasks.

The **DISABLED** keyword means that this command is disabled. xymonlaunch will not start this task. It is recommended that you use this to disable standard tasks, instead of removing them or commenting them out. Upgrades to xymon will add standard tasks back into the file, so unless you have them listed as DISABLED then tasks may re-appear unexpectedly after an upgrade.

The **ONHOST** keyword tells xymonlaunch that this task should only run on specific hosts. After the ONHOST keyword, you must provide a “regular expression”; if the hostname where xymonlaunch runs matches this expression, then the task will run. If it doesn’t match, then the task is treated as if it were DISABLED.

The **NEEDS** instructs xymonlaunch not to run this task unless the task defined by the NEEDS keyword is already running. This is used e.g. to delay the start of some application until the needed daemons have been started. The task that must be running is defined by its **key**

The **GROUP** keyword can be used to limit the number of tasks that may run simultaneously. E.g. if you are generating multiple pagesets of webpages, you don't want them to run at the same time. Putting them into a GROUP will cause xymonlaunch to delay the start of new tasks, so that only one task will run per group. You can change the limit by defining the group before the tasks, with a “GROUP groupname maxtasks” line.

The **INTERVAL** keyword defines how often this command is executed. The example shows a command that runs every 5 minutes. If no interval is given, the task is only run once - this is useful for tasks that run continually as daemons - although if the task stops for some reason, then xymonlaunch will attempt to restart it. Intervals can be specified in seconds (if you just put a number there), or in minutes (5m), hours (2h), or days (1d).

The **ENVFILE** setting points to a file with definitions of environment variables. Before running the task, xymonlaunch will setup all of the environment variables listed in this file. Since this is a per-task setting, you can use the same xymonlaunch instance to run e.g. both the server- and client-side xymon tasks. If this option is not present, then the environment defined to xymonlaunch is used.

The **ENVAREA** setting modifies which environment variables are loaded, by picking up the ones that are defined for this specific “area”. See *xymonserver:cfg(5)* for information about environment areas.

The **LOGFILE** setting defines a logfile for the task. xymonlaunch will start the task with stdout and stderr redirected to this file. If this option is not present, then the output goes to the same location as the xymonlaunch output.

16.11.4 SEE ALSO

xymonlaunch(8), *hobbitd(8)*, *hobbit(7)*

16.12 BB-EVENTLOG.CGI

bb-eventlog.cgi - CGI program to report the xymon eventlog

16.12.1 SYNOPSIS

bb-eventlog.cgi

16.12.2 DESCRIPTION

bb-eventlog.cgi is invoked as a CGI script via the bb-eventlog.sh CGI wrapper. Based on the parameters it receives, it generates the xymon event log for a period. This log shows all status changes that have occurred for all hosts and services.

bb-eventlog.cgi is passed a QUERY_STRING environment variable with the following parameters:
MAXTIME(maximumminutestogobackinthelog) MAXCOUNT(maximumnumberofeventstoreport)

16.12.3 OPTIONS

-top=N Shows a summary of the top-N changing hosts and services.

-env=FILENAME Loads the environment defined in FILENAME before executing the CGI script.

16.12.4 SEE ALSO

bb-hosts(5), xymonserver.cfg(5)

16.13 HOBBIT-MAILACK

xymon-mailack - permit acknowledging alerts via e-mail

16.13.1 SYNOPSIS

xymon-mailack **-env=FILENAME** [**-debug**]

16.13.2 DESCRIPTION

xymon-mailack normally runs as an input mail-filter for the hobbit user, e.g. by being called from the hobbit users' *procmailrc(5)* file. xymon-mailack recognizes e-mails that are replies to *hobbitd_alert(8)* mail alerts, and converts the reply mail into an acknowledge message that is sent to the xymon system. This permits an administrator to acknowledge an alert via e-mail.

16.13.3 ADDING INFORMATION TO THE REPLY MAIL

By default, an acknowledgment is valid for 1 hour. If you know in advance that solving the problem is going to take longer, you can change this by adding **delay=DURATION** to the subject of your mail reply or on a line in the reply message. Duration is in minutes, unless you add a trailing 'h' (for 'hours'), 'd' (for 'days') or 'w' (for 'weeks').

You can also include a message that will show up on the status-page together with the acknowledgment, e.g. to provide an explanation for the issue or some other information to the users. You can either put it at the end of the subject line as **msg=Some random text**, or you can just enter it in the e-mail as the first non-blank line of text in the mail (a "delay=N" line is ignored when looking for the message text).

16.13.4 USE WITH PROCMAIL

To setup xymon-mailack, create a **.procmailrc** file in the xymon-users home-directory with the following contents:

```
DEFAULT=$HOME/Mailbox
LOGFILE=$HOME/procmail.log
:0
| $HOME/server/bin/xymon-mailack --env=$HOME/server/etc/hobbitserver.cfg
```

16.13.5 USE WITH QMAIL

If you are using Qmail to deliver mail locally, you can run xymon-mailack directly from a **.qmail** file. Setup the xymon-users .qmail file like this:

```
| $HOME/server/bin/xymon-mailack --env=$HOME/server/etc/hobbitserver.cfg
```

16.13.6 OPTIONS

-env=FILENAME Load environment from FILENAME, usually xymonserver.cfg.

-debug Dont send a message to xymond, but dump the message to stdout.

16.13.7 SEE ALSO

xymond_alert(8), *hobbitd(8)*, *hobbit(7)*

16.14 BB-FINDHOST.CGI

bb-findhost.cgi - xymon CGI script to find hosts

16.14.1 SYNOPSIS

bb-findhost.cgi?host=REGEX

16.14.2 DESCRIPTION

bb-findhost.cgi is invoked as a CGI script via the bb-findhost.sh CGI wrapper.

bb-findhost.cgi is passed a QUERY_STRING environment variable with the “host=REGEX” parameter. The REGEX is a Posix regular expression (see *regex(7)*) describing the hostnames to look for. A trailing wildcard is assumed on all hostnames - e.g. requesting the hostname “www” will match any host whose name begins with “www”.

It then produces a single web page, listing all of the hosts that matched any of the hostnames, with links to the xymon webpages where they are located.

The output page lists hosts in the order they appear in the *bb-hosts(5)* file.

A sample web page implementing the search facility is included with bbgen, you access it via the URL /bb/help/bb-findhost.html.

16.14.3 OPTIONS

-env=FILENAME Loads the environment from FILENAME before executing the CGI.

16.14.4 FILES

\$BBHOME/web/findhost_header HTML header file for the generated web page

\$BBHOME/web/findhost_footer HTML footer file for the generated web page

\$BBHOME/web/findhost_form Query form displayed when bb-findhost.cgi is called with no parameters.

16.14.5 ENVIRONMENT VARIABLES

BBHOSTS bb-findhost.cgi uses the BBHOSTS environment variable to find the bb-hosts file listing all known hosts and their page locations.

BBHOME Used to locate the template files for the generated web pages.

16.14.6 SEE ALSO

bbgen(1), bb-hosts(5), xymonserver.cfg(5)

16.15 HOBBITCLIENT.CFG

xymonclient.cfg - xymon client environment variables

16.15.1 DESCRIPTION

xymon programs use multiple environment variables beside the normal set of variables. For the xymon client, the environment definitions are stored in the xymon/client/etc/hobbitclient.cfg file. Each line in this file is of the form **NAME=VALUE** and defines one environment variable NAME with the value VALUE.

16.15.2 SETTINGS

BBDISP The IP-address used to contact the xymon server. Default: Chosen when the xymon client was compiled.

BBDISPLAYS List of IP-adresses of xymon servers. Data will be sent to all of the servers listed here. This setting is only used if BBDISP=0.0.0.0.

BBPORT The portnumber for used to contact the xymon server. Default: 1984.

BBHOME The xymon client top-level directory. Default: The \$HOBBITCLIENTHOME setting inherited from the “runclient.sh” script which starts the xymon client.

BBCLIENTLOGS The directory for the xymon clients’ own logfiles. Default: \$BBHOME/logs

BBTMP Directory used for temporary files. Default: \$BBHOME/tmp/

BB Full path to the *bb(1)* client program. Default: \$BBHOME/bin/bb.

Commands Many extension scripts expect a series of environment variables to point at various system utilities. These are included in the file when the client is built.

16.15.3 INHERITED SETTINGS

Some environment variables are inherited from the “runclient.sh” script which launches the xymon client:

MACHINEDOTS The hostname of the local system. Default: Taken from “uname -n”.

MACHINE The hostname of the local system, with dots replaced by commas. For compatibility with Big Brother extension scripts.

BBOSTYPE The operating system of the local system, in lowercase. Default: taken from “uname -s”.

HOBBITCLIENTHOME The top-level directory for the xymon client. Default: The location of the “runclient.sh” script.

16.15.4 SEE ALSO

xymon(7)

Chapter 17

xymon/BB Messages Protocols

17.1 BNF

Extended Backus.Naur Form (http://en.wikipedia.org/wiki/Backus-Naur_form BNF) is a metasyntax used to express context-free grammars: that is, a formal way to describe formal languages. We use BNF to describe the formal language in used between xymon client and server.

17.1.1 xymon messages syntax

```
<hb-messages> ::= <status_msg> <notify_msg> <data_msg> <enable_msg> <query_msg>
                 <config_msg> <drop_msg> <rename_msg> <hobbitlog_msg> <hobbitdxlog_msg>
                 <xymondboard_msg> <download_msg> <client_msg> <clientlog_msg> <ping_msg>
                 <pullclient_msg> <ghostlist_msg> <schedule_msg>

<drop_msg>   ::= <drop HOSTNAME> <EOL> | <drop HOSTNAME TESTNAME> <EOL>
<HOSTNAME>   ::= <a..z>
<TESTNAME>   ::= <cpu|trends|conn|info|>
<status_msg> ::= <status [+LIFETIME] [/group:GROUP] HOSTNAME.TESTNAME COLOR |
                  <additional text> ] > <EOL>
<notify_msg> ::= <notify HOSTNAME.TESTNAME <message text> ] >
<data_msg>    ::= <data HOSTNAME.DATANAME <newline> <additional text> >
<enable_msg>  ::= [enable HOSTNAME.TESTNAME]
<query_msg>   ::= [query HOSTNAME.TESTNAME]
<config_msg>  ::= [config FILENAME]
<drop_msg>    ::= [drop HOSTNAME]
<rename_msg>  ::= [rename OLDHOSTNAME NEWHOSTNAME]
<hobbitlog_msg> ::= 
<hobbitdxlog_msg> ::= [xymondxlog HOSTNAME.TESTNAME]
<xymondboard_msg> ::= [hobbitboard [CRITERIA] [fields=FIELDLIST]]
<download_msg> ::= [download FILENAME] <EOL>
<client_msg>   ::= [client HOSTNAME.OSTYPE [HOSTCLASS]]
<clientlog_msg> ::= [clientlog HOSTNAME [section=SECTIONNAME[, SECTIONNAME...]]]
<ping_msg>    ::= [ping] <EOL>
<pullclient_msg> ::= [pullclient] <EOL>
<ghostlist_msg> ::= [ghostlist] <EOL>
<schedule_msg> ::= [schedule [TIMESTAMP COMMAND]] <EOL>
<SECTIONNAME> ::= [cpu|la| others sectionname]
```

17.1.2 Big Brother messages syntax

```

<bb-messages> ::= <status_msg> <notify_msg> <data_msg> <enable_msg> <query_msg>
                 <config_msg> <drop_msg> <rename_msg> <hobbitlog_msg> <hobbitxlog_msg>
                 <xymondboard_msg> <download_msg> <client_msg> <clientlog_msg> <ping_msg>
                 <pullclient_msg> <ghostlist_msg> <schedule_msg>

<drop_msg>   ::= <drop HOSTNAME> <EOL> | <drop HOSTNAME TESTNAME> <EOL>
<HOSTNAME>   ::= <a...z>
<TESTNAME>   ::= <cpu|trends|conn|info|>
<status_msg> ::= <status[+LIFETIME] [/group:GROUP] HOSTNAME.TESTNAME COLOR |
                  <additional text> ]> <EOL>
<notify_msg> ::= <notify HOSTNAME.TESTNAME <message text> ] >
<data_msg>   ::= <data HOSTNAME.DATANAME <newline> <additional text> >
<enable_msg> ::= [enable HOSTNAME.TESTNAME]
<query_msg>  ::= [query HOSTNAME.TESTNAME]
<config_msg> ::= [config FILENAME]
<drop_msg>   ::= [drop HOSTNAME]
<rename_msg> ::= [rename OLDHOSTNAME NEWHOSTNAME]
<hobbitlog_msg> ::= 
<hobbitxlog_msg> ::= [xymondxlog HOSTNAME.TESTNAME]
<xymondboard_msg> ::= [hobbitboard [CRITERIA] [fields=FIELDLIST]]
<download_msg> ::= [download FILENAME] <EOL>
<client_msg>  ::= [client HOSTNAME.OSTYPE [HOSTCLASS]]
<clientlog_msg> ::= [clientlog HOSTNAME [section=SECTIONNAME[, SECTIONNAME...]]]
<ping_msg>   ::= [ping] <EOL>
<pullclient_msg> ::= [pullclient] <EOL>
<ghostlist_msg> ::= [ghostlist] <EOL>
<schedule_msg> ::= [schedule [TIMESTAMP COMMAND]] <EOL>
<SECTIONNAME> ::= [cpu|la| others sectionname]
```

17.2 bb - xymon client communication program

17.2.1 SYNOPSIS

`"/opt/bin/bb" [options] RECIPIENT message`

17.2.2 DESCRIPTION

bb(1) is the client program used to communicate with a xymon server. It is frequently used by xymon client systems to send in status messages and pager alerts on local tests.

In xymon, the bb program is also used for administrative purposes, e.g. to rename or delete hosts, or to disable hosts that are down for longer periods of time.

17.2.3 OPTIONS AND PARAMETERS

-debug Enable debugging. This prints out details about how the connection to the BBDISPLAY server is being established.

-proxy=http://PROXYSERVER:PROXPORT/ When sending the status messages via HTTP, use this server as an HTTP proxy instead of connecting directly to the BBDISPLAY server.

-timeout=N Specifies the timeout for connecting to the xymon server, in seconds. The default is 5 seconds.

RECIPIENT The **RECIPIENT** parameter defines which server receives the message. If **RECIPIENT** is given as “0.0.0.0”, then the message is sent to all of the servers listed in the BBDISPLAYS or BBPAGERS environment variable (for “status” and “page” messages, respectively).

Usually, a client will use “\$BBDISP” for the **RECIPIENT** parameter, as this is defined for the client scripts to automatically contain the correct value.

The **RECIPIENT** parameter may be a URL for a webserver that has the *bbmessage.cgi* or similar script installed. This tunnels the xymon messages to the BBDISPLAY server using standard HTTP protocol. The *bbmessage.cgi(8)* CGI tool (included in xymon) must be installed on the webserver for the HTTP transport to work.

MESSAGE The **message** parameter is the message to be sent across to the xymon server. Messages must be enclosed in quotes, but by doing so they can span multiple lines. The maximum size of a message is defined by the maximum allowed length of your shell’s commandline, and is typically 8-32 KB.

If you need to send longer status messages, you can specify “@” as the message: bb will then read the status message from its stdin.

17.2.4 HOBBIT MESSAGE SYNTAX

This section lists the most commonly used messages in the xymon protocol.

status[+LIFETIME [/group:GROUP] HOSTNAME.TESTNAME COLOR <additional text>] This sends in a status message for a single test (column) on a single host. TESTNAME is the name of the column where this test will show up; any name is valid except that using dots in the testname will not work. COLOR must be one of the valid colors: “green”, “yellow”, “red” or “clear”. The colors “blue” and “purple” - although valid colors - should not be sent in a status-message, as these are handled specially by the xymon server.

The “additional text” normally includes a local timestamp and a summary of the test result on the first line. Any lines following the first one are free-form, and can include any information that may be useful to diagnose the problem being reported.

The LIFETIME defines how long this status is valid after being received by the xymon server. The default is 30 minutes, but you can set any period you like. E.g. for a custom test that runs once an hour, you will want

to set this to at least 60 minutes - otherwise the status will go purple after 30 minutes. It is a good idea to set the LIFETIME to slightly more than the interval between your tests, to allow for variations in the time it takes your test to complete. The LIFETIME is in minutes, unless you add an “h” (hours), “d” (days) or “w” (weeks) immediately after the number, e.g. “status+5h” for a status that is valid for 5 hours.

The GROUP option is used to direct alerts from the status to a specific group. It is currently used for status generated from the xymon clients’ data, e.g. to direct alerts for a “procs” status to different people, depending on exactly which process is down.

notify HOSTNAME.TESTNAME <message text> This triggers an informational message to be sent to those who receive alerts for this HOSTNAME+TESTNAME combination, according to the rules defined in *xymon-alerts.cfg(5)*. This is used by the *xymon-enadis.cgi(1)* tool to notify people about hosts being disabled or enabled, but can also serve as a general way of notifying server administrators.

data HOSTNAME.DATANAME<newline><additional text> The “data” message allows tools to send data about a host, without it appearing as a column on the xymon webpages. This is used e.g. to report statistics about a host, e.g. vmstat data which does not in itself represent something that has a red, yellow or green identity. It is used by RRD bottom-feeder modules, among others. In xymon, data messages are by default processed only by the *xymond_rrd(8)* module. If you want to handle data-messages by an external application, you may want to enable the *xymond_filestore(8)* module for data-messages, to store data-messages in a format compatible with how the Big Brother daemon does.

Disables a specific test for DURATION minutes. This will cause the status of this test to be listed as “blue” on the BBDISPLAY server, and no alerts for this host/test will be generated. If DURATION is given as a number followed by s/m/h/d, it is interpreted as being in seconds/minutes/hours/days respectively. **To**

enable HOSTNAME.TESTNAME Re-enables a test that had been disabled.

query HOSTNAME.TESTNAME Query the BBDISPLAY server for the latest status reported for this particular test. If the host/test status is known, the response is the first line of the status report - the current color will be the first word on the line. Additional lines of text that might be present on the status-message cannot be retrieved.

This allows any xymon client to determine the status of a particular test, whether it is one pertaining to the host where the client is running, some other host, or perhaps the result of a combined test from multiple hosts managed by *bbcombotest(1)*. This will typically be useful to xymon client extension scripts, that need to determine the status of other hosts e.g. to decide if an automatic recovery action should be initiated.

config FILENAME Retrieve one of the xymon configuration files from the server. This command allows a client to pull files from the \$BBHOME/etc/ directory on the server, allowing for semi-automatic updates of the client configuration. Since the configuration files are designed to have a common file for the configuration of all hosts in the system - and this is in fact the recommended way of configuring your clients - this makes it easier to keep the configuration files synchronized.

drop HOSTNAME Removes all data stored about the host HOSTNAME. It is assumed that you have already deleted the host from the bb-hosts configuration file.

drop HOSTNAME TESTNAME Remove data about a single test (column).

rename OLDHOSTNAME NEWHOSTNAME Rename all data for a host that changes its name. You should do this before changing the hostname in the bb-hosts configuration file.

rename HOSTNAME OLDTESTNAME NEWTESTNAME Rename data about a single test (column).

xymondlog HOSTNAME.TESTNAME Retrieve the xymon status-log for a single test. The first line of the response contain a series of fields separated by a pipe-sign:

hostname The name of the host

testname The name of the test

color Status color (green, yellow, red, blue, clear, purple)

testflags For network tests, the flags indicating details about the test (used by bbgen).

lastchange Unix timestamp when the status color last changed.

logtime Unix timestamp when the log message was received.

validtime Unix timestamp when the log message is no longer valid (it goes purple at this time).

acktime -1, or Unix timestamp when an active acknowledgement expires.

disabletime -1, or Unix timestamp when the status is no longer disabled.

sender IP-address where the status was received from.

cookie -1, or the cookie value used to acknowledge an alert.

ackmsg Empty, or the acknowledgment message sent when the status was acknowledged. Newline, pipe-signs and backslashes are escaped by with a backslash in C-style.

dismsg Empty, or the message sent when the status was disabled. Newline, pipe-signs and backslashes are escaped by with a backslash in C-style.

After the first line comes the full status log in plain text format.

xmondxlog HOSTNAME.TESTNAME Retrieves an XML-string with the status log as for the “xmondlog” command.

xmonboard [CRITERIA [fields=FIELDLIST]] Retrieves a summary of the status of all known tests available to the xmon daemon.

By default - if no CRITERIA is provided - it returns one line for all status messages that are found in xmon. You can filter the response by selecting a page, a host, a test or a color - wildcards are not supported, so you can pick only one page, host, test or color.

page=PAGEPATH Include only tests from hosts found on the PAGEPATH page in the bb-hosts file.

host=HOSTNAME Include only tests from the host HOSTNAME

test=TESTNAME Include only tests with the testname TESTNAME

color=COLORNAME Include only tests where the status color is COLORNAME

You can filter on e.g. both a hostname and a testname.

The response is one line for each status that matches the CRITERIA, or all statuses if no criteria is specified. The line is composed of a number of fields, separated by a pipe-sign. You can select what fields to retrieve by listing them in the FIELDLIST. The following fields are available:

hostname The name of the host

testname The name of the test

color Status color (green, yellow, red, blue, clear, purple)

flags For network tests, the flags indicating details about the test (used by bbgen).

lastchange Unix timestamp when the status color last changed.

logtime Unix timestamp when the log message was received.

validtime Unix timestamp when the log message is no longer valid (it goes purple at this time).

acktime -1, or Unix timestamp when an active acknowledgement expires.

disabletime -1, or Unix timestamp when the status is no longer disabled.

sender IP-address where the status was received from.

cookie -1, or the cookie value used to acknowledge an alert.

line1 First line of status log.

ackmsg Empty (if no acknowledge is active), or the text of the acknowledge message.

dismsg Empty (if the status is currently enabled), or the text of the disable message.

msg The full text of the current status message.

The ackmsg, dismsg and msg fields have certain characters encoded: Newline is “\n”, TAB is “\t”, carriage return is “\r”, a pipe-sign is “\|”, and a backslash is “\\”.

If the “fields” parameter is omitted, a default set of hostname,testname,color,flags,lastchange,logtime,validtime,acktime,disabletime is used.

xymondxboard Retrieves an XML-string with the summary of all status logs as for the “xymonboard” command.

download FILENAME Download a file from the xymon servers’ download directory.

client HOSTNAME.OSTYPE [HOSTCLASS] Used to send a “client” message to the xymon server. Client messages are generated by the xymon client; when sent to the Xymon server they are matched against the rules in the *xymon-clients.cfg(5)* configuration file, and status messages are generated for the client-side tests.

clientlog HOSTNAME [section=SECTIONNAME[,SECTIONNAME...]] Retrieves the current raw client message last sent by HOSTNAME. The optional “section” filter is used to select specific sections of the client data.

ping Attempts to contact the xymon server. If successful, the xymon server version ID is reported.

pullclient This message is used when fetching client data via the “pull” mechanism implemented by *xymonfetch(8)* and *msgcache(8)* for clients that cannot connect directly to the xymon server.

ghostlist Report a list of **ghost** clients seen by the xymon server. Ghosts are systems that report data to the Xymon server, but are not listed in the bb-hosts file.

schedule [TIMESTAMP COMMAND] Schedules an command sent to the xymon server for execution at a later time. E.g. used to schedule disabling of a host or service at sometime in the future. COMMAND is a complete xymon command such as the ones listed above. TIMESTAMP is the Unix epoch time when the command will be executed.

If no parameters are given, the currently scheduled tasks are listed in the response. The response is one line per scheduled command, with the job-id, the time when the command will be executed, the IP address from which this was sent, and the full command string. To cancel an previously scheduled command, “**schedule cancel JOBID**” can be used. JOBID is a number provided as the first item in the output from the schedule list.

17.2.5 EXAMPLE

Send a normal status message to the BBDISPLAY server, using the standard xymon protocol on TCP port 1984:

\$\$BB\$BBDISP”statuswww.foo.com.httpgreen‘date‘WebOK”

Send the same status message, but using HTTP protocol via the webservers’ bbmessage.cgi script:

\$\$BB`http://bb.foo.com/cgi-bin/bbmessage.cgi”statuswww.foo.com.httpgreen‘date‘WebOK”`

Use “query” message to determine the color of the “www” test, and restart Apache if it is red:

\$\$WWW='\$\$BB\$BBDISP”querywww.foo.com.www“—awk’{print\$1}‘‘ \$if[“\$WWW”=”red”];then/etc/init.d/apacherestart;fi

Use “config” message to update the local bb-dftab file (but only if we get a response):

\$\$BB\$BBDISP”configbb-dftab”>/tmp/bb-dftab.new \$if[-s/tmp/bb-dftab.new];then mv/tmp/bb-dftab.new\$BBHOME/etc/bb-dftab fi

Send a very large status message that has been built in the file “statusmsg.txt”. Instead of providing it on the command line, pass it via stdin to the bb command:

\$catstatusmsg.txt—\$\$BB\$BBDISP”@”

17.2.6 NOTES

This man-page describes the bb client program provided as part of xymon . This implementation provides features not present in the standard Big Brother bb client - specifically, the support for sending messages over HTTP, and many commands such as “query” and “config” are not part of the bb client shipped with Big Brother.

17.2.7 SEE ALSO

bbcombotest(1), bb-hosts(5), xymonserver.cfg(5), hobbit(7)

Chapter 18

xymon server programs

18.1 HOBBITD_CHANNEL

xymond_channel - Feed a hobbitd channel to a worker module

18.1.1 SYNOPSIS

xymond_channel **-channel=CHANNEL [options]** **workerprogram [worker-options]**

18.1.2 DESCRIPTION

xymond_channel hooks into one of the *hobbitd(8)* channels that provide information about events occurring in the xymon system. It retrieves messages from the xymond daemon, and passes them on to the **workerprogram** on the STDIN (file descriptor 1) of the worker program. Worker programs can then handle messages as they like.

A number of worker programs are shipped with xymond, e.g. *xymond_filestore(8)*

xymond_history(8) *xymond_alert(8)* *xymond_rrd(8)* *xymond_client(8)*

If you want to write your own worker module, a sample worker module is provided as part of the xymond distribution in the *xymond_sample.c* file. This illustrates how to easily fetch and parse messages.

18.1.3 OPTIONS

xymond_channel accepts a few options.

-channel=CHANNELNAME Specifies the channel to receive messages from, only one channel can be used. This option is required. The following channels are available:

“status” receives all xymon status- and summary-messages “stachg” receives information about status changes
“page” receives information about statuses triggering alerts “data” receives all xymon “data” messages “notes” receives all xymon “notes” messages “enadis” receives information about hosts being disabled or enabled.
“client” receives all data sent from xymon client systems

-net=PEERSERVER:PEERPORT Instead of launching a worker module as a local task, the messages in the channel are forwarded over a TCP/IP connection to another host where the worker module is running. This is typically used for sharing the load of the heavier worker modules across multiple systems, e.g. the *xymond_client* and *xymond_rrd* workers may receive data this way. With this option, the **workerprogram** parameter is ignored and may be omitted.

On the remote server, the worker modules are usually launched via inetd.

-daemon `xymond_channel` is normally started by `xymonlaunch(8)` as a task defined in the `xymonlaunch.cfg(5)` file. If you are not using `hobbitlaunch`, then starting `xymond_channel` with this option causes it to run as a stand-alone background task.

-pidfile=FILENAME If running as a stand-alone daemon, `xymond_channel` will save the process-ID of the daemon in FILENAME. This is useful for automated startup- and shutdown- scripts.

-env=FILENAME Loads the environment variables defined in FILENAME before starting `xymond_channel`. This is normally used only when running as a stand-alone daemon; if `xymond_channel` is started by `xymonlaunch`, then the environment is controlled by the task definition in the `xymonlaunch.cfg(5)` file.

-log=FILENAME Redirect output to this log-file.

-debug Enable debugging output.

18.1.4 FILES

This program does not use any configuration files.

18.1.5 SEE ALSO

`xymond(8)`, `hobbit(7)`

18.2 BB-HOSTSVC.CGI

hgcmdxymonsvc.cgi - CGI program to view xymon status logs

18.2.1 SYNOPSIS

xymonsvc.cgi [-hobbitd—historical] [-history=top—bottom]

18.2.2 DESCRIPTION

xymonsvc.cgi is a CGI program to present a xymon status log in HTML form (ie, as a web page). It can be used both for the logs showing the current status, and for historical logs from the “histlogs” directory. It is normally invoked as a CGI program, and therefore receives most of the input parameters via the CGI QUERY_STRING environment variable.

Unless the “–historical” option is present, the current status log is used. This assumes a QUERY_STRING environment variable of the form

HOSTSVC=hostname.servicename where “hostname” is the name of the host with commas instead of dots, and “servicename” is the name of the service (the column name in xymon). Such links are automatically generated by the *bbgen(1)* tool when the environment contains “BBLOGSTATUS=dynamic”.

With the “–historical” option present, a historical logfile is used. This assumes a QUERY_STRING environment variable of the form

HOST=hostname&SERVICE=servicename&TIMEBUF=timestamp where “hostname” is the name of the host with commas instead of dots, “servicename” is the name of the service, and “timestamp” is the time of the log. This is automatically generated by the *bb-hist.cgi(1)* tool.

18.2.3 OPTIONS

-xymond Retrieve the current status log from *hobbitd(1)* rather than from the logfile. This is for use with the xymon daemon from the Xymon monitor version 4.

-historical Use a historical logfile instead of the current logfile.

-history=top—bottom—none When showing the current logfile, provide a “HISTORY” button at the top or the bottom of the webpage, or not at all. The default is to put the HISTORY button at the bottom of the page.

-env=FILENAME Load the environment from FILENAME before executing the CGI.

-templates=DIRECTORY Where to look for the HTML header- and footer-templates used when generating the webpages. Default: \$BBHOME/web/

-no-svcid Do not include the HTML tags to identify the hostname/service on the generated web page. Useful is this already happens in the hostsvc_header template file, for instance.

-multigraphs=TEST1[,TEST2] This causes xymonsvc.cgi to generate links to service graphs that are split up into multiple images, with at most 5 graphs per image. This option only works in xymon mode. If not specified, only the “disk” status is split up this way.

-no-disable By default, the info-column page includes a form allowing users to disable and re-enable tests. If your setup uses the default separation of administration tools into a separate, password-protected area, then use of the disable- and enable-functions requires access to the administration tools. If you prefer to do this only via the dedicated administration page, this option will remove the disable-function from the info page.

-no-jvalidation The disable-function on the info-column page by default uses JavaScript to validate the form before submitting the input to the xymon server. However, some browsers cannot handle the Javascript code correctly so the form does not work. This option disables the use of Javascript for form-validation, allowing these browsers to use the disable-function.

18.2.4 FILES

\$BBHOME/web/hostsvc_header HTML template header

\$BBHOME/web/hostsvc_footer HTML template footer

18.2.5 ENVIRONMENT

NONHISTS=info,trends,graphs A comma-separated list of services that does not have meaningful history, e.g. the “info” and “trends” columns. Services listed here do not get a “History” button.

TEST2RRD=test,test A comma-separated list of the tests that have an RRD graph.

18.2.6 SEE ALSO

xymon(7), hobbitd(1)

18.3 BBPROXY

`bbproxy` - xymon message proxy

18.3.1 SYNOPSIS

`bbproxy [options] –servers=IP`

18.3.2 DESCRIPTION

`bbproxy(8)` is a proxy for forwarding xymon messages from one server to another. It will typically be needed if you have clients behind a firewall, so they cannot send status messages to the Xymon server directly.

[Wau]hgcmdbbproxy serves three purposes. First, it acts as a regular proxy server, allowing clients that cannot connect directly to the xymon server to send messages to the Xymon servers. Although bbproxy is optimized for handling status messages, it will forward all types of messages.

Second, it acts as a buffer, smoothing out peak loads if many clients try to send status messages simultaneously. bbproxy can absorb messages very quickly, but will queue them up internally and forward them to the xymon server at a reasonable pace. This helps even out the load on your Xymon server.

Third, bbproxy merges small “status” messages into larger “combo” messages. This can dramatically decrease the number of connections that need to go from bbproxy to the xymon server, and is a slightly more efficient way of transmitting data to the Xymon server. The merging of messages causes “status” messages to be delayed for up to 0.25 seconds before being sent off to the Hobbit server.

18.3.3 OPTIONS

–servers=SERVERIP[:PORT] [,SERVER2IP[:PORT]]] Specifies the IP-address and optional portnumber where incoming messages are forwarded to. The default portnumber is 1984, the standard xymon port number. Up to 3 servers can be specified; incoming messages are sent to all of them (except “config”, “query” and “download” messages, which go to the LAST server only). If you have Xymon clients sending their data via this proxy, note that the clients will receive their configuration data from the LAST of the servers listed here. This option is required.

–bbdisplay=SERVERIP[:PORT] [,SERVER2IP[:PORT]] Obsolete. Use “–servers” instead.

–listen=LOCALIP[:PORT] Specifies the IP-adress where bbproxy listens for incoming connections. By default, bbproxy listens on all IP-adresses assigned to the host. If no portnumber is given, port 1984 will be used.

–timeout=N Specifies the number of seconds after which a connection is aborted due to a timeout. Default: 10 seconds.

–report=[PROXYHOSTNAME. PROXYSERVICE] If given, this option causes bbproxy to send a status report every 5 minutes to the xymon server about itself. If you have set the standard Xymon environment, you can use “–report=bbproxy” to have bbproxy report its status to a “bbproxy” column in Hobbit. The default for PROXYHOSTNAME is the \$MACHINE environment variable, i.e. the hostname of the server running bbproxy. See REPORT OUTPUT below for an explanation of the report contents.

–lqueue=N Size of the listen-queue where incoming connections can queue up before being processed. This should be large to accomodate bursts of activity from clients. Default: 512.

–daemon Run in daemon mode, i.e. detach and run as a background proces. This is the default.

–no-daemon Runs bbproxy as a foreground proces.

–pidfile=FILENAME Specifies the location of a file containing the proces-ID of the bbproxy daemon proces. Default: /var/run/bbproxy.pid.

-logfile=FILENAME Sends all logging output to the specified file instead of stderr.

-log-details Log details (IP-address, message type and hostname) to the logfile. This can also be enabled and disabled at run-time by sending the bbproxy process a SIGUSR1 signal.

-debug Enable debugging output.

18.3.4 REPORT OUTPUT

If enabled via the “–report” option, bbproxy will send a status message about itself to the xymon server once every 5 minutes.

The status message includes the following information:

Incoming messages The total number of connections accepted from clients since the proxy started. The “(N msgs/second)” is the average number of messages per second over the past 5 minutes.

Outbound messages The total number of messages sent to the xymon servers. Note that this is probably smaller than the number of incoming messages, since bbproxy merges messages before sending them.

Incoming - Combo messages The number of “combo” messages received from a client.

Incoming - Status messages The number of “status” messages received from a client. bbproxy attempts to merge these into “combo” messages. The “Messages merged” is the number of “status” messages that were merged into a combo message, the “Resulting combos” is the number of “combo” messages that resulted from the merging.

Incoming - Page messages The number of “page” messages received from a client. These are discarded, they are generated by the old Big Brother clients, but have no meaning in xymon.

Incoming - Other messages The number of other messages (data, notes, ack, query, ...) messages received from a client.

Proxy resources - Connection table size This is the number of connection table slots in the proxy. This measures the number of simultaneously active requests that the proxy has handled, and so gives an idea about the peak number of clients that the proxy has handled simultaneously.

Proxy resources - Buffer space This is the number of KB memory allocated for network buffers.

Timeout details - reading from client The number of messages dropped because reading the message from the client timed out.

Timeout details - connecting to server The number of messages dropped, because a connection to the xymon server could not be established.

Timeout details - sending to server The number of messages dropped because the communication to the xymon server timed out after a connection was established.

Timeout details - recovered When a timeout happens while sending the status message to the server, bbproxy will attempt to recover the message and retry sending it to the server after waiting a few seconds. This number is the number of messages that were recovered, and so were not lost.

Timeout details - reading from server The number of response messages that timed out while attempting to read them from the server. Note that this applies to the “config” and “query” messages only, since all other message types do not get any response from the servers.

Timeout details - sending to client The number of response messages that timed out while attempting to send them to the client. Note that this applies to the “config” and “query” messages only, since all other message types do not get any response from the servers.

Average queue time The average time it took the proxy to process a message, calculated from the messages that have passed through the proxy during the past 5 minutes. This number is computed from the messages that actually end up establishing a connection to the xymon server, i.e. status messages that were combined into combo-messages do not go into the calculation - if they did, it would reduce the average time, since it is faster to merge messages than send them out over the network.

18.3.5

If you think the numbers do not add up, here is how they relate.

The “Incoming messages” should be equal to the sum of the “Incoming Combo/Status/Page/Other messages”, or slightly more because messages in transit are not included in the per-type message counts.

The “Outbound messages” should be equal to sum of the “Incoming Combo/Page/Other messages”, plus the “Resulting combos” count, plus “Incoming Status messages” minus “Messages merged” (this latter number is the number of status messages that were NOT merged into combos, but sent directly). The “Outbound messages” may be slightly lower than that, because messages in transit are not included in the “Outbound messages” count until they have been fully sent.

18.3.6 SIGNALS

SIGHUP Re-opens the logfile, e.g. after it has been rotated.

SIGTERM Shut down the proxy.

SIGUSR1 Toggles logging of individual messages.

18.3.7 SEE ALSO

bb(1), xymond(8), hobbit(7)

18.4 HOBBITD_CLIENT

xymond_client - hobbitd worker module for client data

18.4.1 SYNOPSIS

xymond_channel –channel=client hobbitd_client [options]

18.4.2 DESCRIPTION

xymond_client is a worker module for hobbitd, and as such it is normally run via the *hobbitd_channel(8)* program. It receives xymond client messages sent from systems that have the xymon client installed, and use the client data to generate the Xymon status messages for the cpu-, disk-, memory- and procs-columns. It also feeds Hobbit data messages with the netstat- and vmstat-data collected by the client.

When generating these status messages from the client data, xymond_client will use the configuration rules defined in the *hobbit-clients.cfg(5)* file to determine the color of each status message.

18.4.3 OPTIONS

- clear-color=COLOR** Define the color used when sending “msgs”, “files” or “ports” reports and there are no rules to check for these statuses. The default is to show a “clear” status, but some people prefer to have it “green”. If you would rather prefer not to see these status columns at all, then you can use the “–no-clear-msgs”, “–no-clear-files” and “–no-clear-ports” options instead.
- no-clear-msgs** If there are no logfile checks, the “msgs” column will show a “clear” status. If you would rather avoid having a “msgs” column, this option causes xymond_client to not send in a clear “msgs” status.
- no-clear-files** If there are no file checks, the “files” column will show a “clear” status. If you would rather avoid having a “files” column, this option causes xymond_client to not send in a clear “files” status.
- no-clear-ports** If there are no port checks, the “ports” column will show a “clear” status. If you would rather avoid having a “ports” column, this option causes xymond_client to not send in a clear “ports” status.
- no-ps-listing** Normally the “procs” status message includes the full process-listing received from the client. If you prefer to just have the monitored processes shown, this option will turn off the full ps-listing.
- no-port-listing** Normally the “ports” status message includes the full netstat-listing received from the client. If you prefer to just have the monitored ports shown, this option will turn off the full netstat-listing.
- config=FILENAME** Sets the filename for the *xymon-clients.cfg* file. The default value is “etc/xymon-clients.cfg” below the xymon server directory.
- dump-config** Dumps the configuration after parsing it. May be useful to track down problems with configuration file errors.
- test** Starts an interactive session where you can test the *xymon-clients.cfg* configuration.
- debug** Enable debugging output.

18.4.4 FILES

xymon/server/etc/hobbit-clients.cfg

18.4.5 SEE ALSO

xymon-clients.cfg(5), *hobbitd(8)*, *hobbitd_channel(8)*, *hobbit(7)*

18.5 HOBBITWEB

xymon web page headers, footers and forms.

18.5.1 DESCRIPTION

The xymon webpages are somewhat customizable, by modifying the header- and footer-templates found in the `xymon/server/web/` directory. There are usually two or more files for a webpage: A **template_header** file which is the header for this webpage, and a **template_footer** file which is the footer. Webpages where entry forms are used have a **template_form** file which is the data-entry form.

With the exception of the **bulletin** files, the header files are inserted into the HTML code at the very beginning and the footer files are inserted at the bottom.

The following templates are available:

bulletin A **bulletin_header** and **bulletin_footer** is not shipped with xymon, but if they exist then the content of these files will be inserted in all HTML documents generated by Xymon. The “bulletin_header” contents will appear after the normal header for the webpage, and the “bulletin_footer” will appear just before the normal footer for the webpage. These files can be used to post important information about the Hobbit system, e.g. to notify users of current operational or monitoring problems.

acknowledge Header, footer and form template for the xymon **acknowledge alert** webpage generated by *bb-ack.cgi(1)*

bb Header and footer for the xymon **Main view** webpages, generated by *bbgen(1)*

bb2 Header and footer for the xymon **All non-green view** webpage, generated by *bbgen(1)*

bbnk Header and footer for the now deprecated **BBNK** webpage, generated by *bbgen*. You should use the newer *xymon-nkview.cgi(1)* utility instead, which uses the **xymonnk** templates.

bbrep Header and footer for the xymon **Main view** availability report webpages, generated by *bbgen(1)* when running in availability report mode.

bbsnap Header and footer for the xymon **Main view** snapshot webpages, generated by *bbgen(1)* when running in snapshot report mode.

bbsnap2 Header and footer for the xymon **All non-green view** snapshot webpage, generated by *bbgen(1)* when running in snapshot report mode.

columndoc Header and footer for the xymon **Column documentation** webpages, generated by the *bb-csvinfo.cgi(1)* utility in the default xymon configuration.

confreport Header and footer for the xymon **Configuration report** webpage, generated by the *xymon-confreport.cgi(1)* utility. Note that there are also “*confreport_front*” and “*confreport_back*” templates, these are inserted into the generated report before the hostlist, and before the column documentation, respectively.

event Header, footer and form for the xymon **Eventlog report**, generated by *xymon-eventlog.cgi(1)*

findhost Header, footer and form for the xymon **Find host** webpage, generated by *bb-findhost.cgi(1)*

graphs Header and footer for the xymon **Graph details** webpages, generated by *xymongraph.cgi(1)*

hist Header and footer for the xymon **History** webpage, generated by *bb-hist.cgi(1)*

histlog Header and footer for the xymon **Historical status-log** webpage, generated by *xymonsvc.cgi(1)* utility when used to show a historical (non-current) status log.

xymonnk Header and footer for the xymon **Critical Systems view** webpage, generated by *xymon-nkview.cgi(1)*

hostsvc Header and footer for the xymon **Status-log** webpage, generated by *xymonsvc.cgi(1)* utility when used to show a current status log.

info Header and footer for the xymon **Info column** webpage, generated by *xymonsvc.cgi(1)* utility when used to show the host configuration page.

maintact Header and footer for the xymon webpage, generated by *xymon-enadis.cgi(1)* utility when using the Enable/Disable “preview” mode.

maint Header, footer and form for the xymon **Enable/disable** webpage, generated by *xymon-enadis.cgi(1)*

nkack Form show on the **status-log** webpage when viewed from the “Critical Systems” overview. This form is used to acknowledge a critical status by the operators monitoring the Critical Systems view.

nkedit Header, footer and form for the **Critical Systems Editor**, the *xymon-nkedit.cgi(1)* utility.

replog Header and footer for the xymon **Report status-log** webpage, generated by *xymonsvc.cgi(1)* utility when used to show a status log for an availability report.

report Header, footer and forms for selecting a pre-generated **Availability Report**. Handled by the *bb-datepage.cgi(1)* utility.

snapshot Header and footer for the xymon **Snapshot report** selection webpage, generated by *bb-snapshot.cgi(1)*

18.5.2 SEE ALSO

bbgen(1), *xymonsvc.cgi(1)*, *hobbit(7)*

Chapter 19

xymon Client

19.1 “/opt/bin/logfetch” - xymon client data collector

19.1.1 SYNOPSIS

logfetch CONFIGFILE STATUSFILE

19.1.2 DESCRIPTION

logfetch is part of the xymon client. It is responsible for collecting data from logfiles, and other file-related data, which is then sent to the Xymon server for analysis.

logfetch uses a configuration file, which is automatically retrieved from the xymon server. There is no configuration done locally. The configuration file is usually stored in the **\$BBHOME/tmp/logfetch.cfg** file, but editing this file has no effect since it is re-written with data from the xymon server each time the client runs.

logfetch stores information about what parts of the monitored logfiles have been processed already in the **\$BBHOME/tmp/logfetch.status** file. This file is an internal file used by **logfetch**, and should not be edited. If deleted, it will be re-created automatically.

19.1.3 SECURITY

logfetch needs read access to the logfiles it should monitor. If you configure monitoring of files or directories through the “file:” and “dir:” entries in *client-local.cfg(5)* then **logfetch** will require at least read-access to the directory where the file is located. If you request checksum calculation for a file, then it must be readable by the xymon client user.

Do **NOT** install **logfetch** as uid-root. There is no way that **logfetch** can check whether the configuration file it uses has been tampered with, so installing **logfetch** with uid-root privileges could allow an attacker to read any file on the system by using a hand-crafted configuration file. In fact, **logfetch** will attempt to remove its own uid-root setup if it detects that it has been installed uid-root.

19.1.4 ENVIRONMENT VARIABLES

DU Command used to collect information about the size of directories. By default, this is the command **du -k**. If the local du-command on the client does not recognize the “-k” option, you should set the DU environment variable in the **\$BBHOME/etc/xymonclient.cfg** file to a command that does report directory sizes in kilobytes.

19.1.5 FILES

\$BBHOME/tmp/logfetch.cfg

\$BBHOME/tmp/logfetch.status

19.1.6 SEE ALSO

xymon(7), hobbit-clients.cfg(5)

19.2 CLIENTUPDATE

19.2.1 NAME

`clientupdate` - xymon client update utility

19.2.2 SYNOPSIS

`clientupdate [options]`

19.2.3 DESCRIPTION

clientupdate is part of the xymon client. It is responsible for updating an existing client installation from a central repository of client packages stored on the Xymon server.

When the xymon client sends a normal client report to the Xymon server, the server responds with the section of the *client-local.cfg(5)* file that is relevant to this client. Included in this may be a “clientversion” value. The clientversion received from the server is compared against the current clientversion installed on the client, as determined by the contents of the \$BBHOME/etc/clientversion.cfg file. If the two versions are not identical, clientupdate is launched to update the client installation.

19.2.4 OPTIONS

-level Report the current clientversion.

-update=NEWVERSION Attempt to update the client to NEWVERSION by fetching this version of the client software from the xymon server.

-reexec Used internally during the update process, see **OPERATION** below.

-remove-self Used internally during the update process. This option causes the running clientupdate utility to delete itself - it is used during the update to purge a temporary copy of the clientupdate utility that is installed in \$BBTMP.

19.2.5 USING CLIENTUPDATE IN HOBBIT

To manage updating clients without having to logon to each server, you can use the clientupdate utility. This is how you setup the release of a new client version.

Create the new client Setup the new client \$BBHOME directory, e.g. by copying an existing client installation to an empty directory and modifying it for your needs. It is a good idea to delete all files in the tmp/ and logs/ directories, since there is no need to copy these over to all of the clients. Pay attention to the etc/ files, and make sure that they are suitable for the systems where you want to deploy this new client. You can add files - e.g. extension scripts in the ext/ directory - but the clientupdate utility cannot delete or rename files.

Package the client When your new client software is ready, create a tar-file of the new client. All files in the tar archive must have filenames relative to the clients' \$BBHOME (usually, xymon/client/). Save the tarfile on the xymon server in `hobbit/server/download/somefile.tar`. Dont compress it. It is recommended that you use some sort of operating-system and version-numbering scheme for the filename, but you can choose whatever filename suits you - the only requirement is that it must end with “.tar”. The part of the filename preceding “.tar” is what Xymon will use as the “clientversion” ID.

Configure which hosts receive the new client In the *client-local.cfg(5)* file, you must now setup a **clientversion:ID** line where the **ID** matches the filename you used for the tar-file. So if you have packaged the new client into the file **linux.v2.tar**, then the corresponding entry in client-local.cfg would be **clientversion:linux.v2**.

Wait for xymond to reload client-local.cfg hobbtd will automatically reload the client-local.cfg file after at most 10 minutes. If you want to force an immediate reload, send a SIGHUP signal to the hobbtd process.

Wait for the client to update The next time the client contacts the xymon server to send the client data, it will notice the new clientversion setting in client-local.cfg, and will run **clientupdate** to install the new client software. So when the client runs the next time, it will use the new client software.

19.2.6 OPERATION

clientupdate runs in two steps:

Re-exec step The first step is when clientupdate is first invoked from the xymonclient.sh script with the “–re-exec” option. This step copies the clientupdate program from \$BBHOME/bin/ to a temporary file in the \$BBTMP directory. This is to avoid conflicts when the update procedure installs a new version of the clientupdate utility itself. Upon completion of this step, the clientupdate utility automatically launches the next step by running the program from the file in \$BBTMP.

Update step The second step downloads the new client software from the xymon server. The new software must be packed into a tar file, which clientupdate then unpacks into the \$BBHOME directory.

19.2.7 ENVIRONMENT VARIABLES

clientupdate uses several of the standard xymon environment variables, including **BBHOME** and **BBTMP**.

19.2.8 SEE ALSO

xymon(7), bb(1), client-local.cfg(5)

19.3 BB-REPLOG.CGI

bb-replog.cgi - CGI program to report service availability log

19.3.1 SYNOPSIS

bb-replog.cgi

19.3.2 DESCRIPTION

bb-replog.cgi is invoked as a CGI script via the bb-replog.sh CGI wrapper. Based on the parameters it receives, it generates an availability report for a specific host-service combination for the requested time-period. The availability report includes a calculation of the availability percentage (split out on percent green, yellow, red time), and an eventlog for the period listing the status changes that have occurred to allow for drill-down to the test reports that indicate a problem. Access to the individual historical status logs go via the *xymonsvc.cgi(1)* CGI script.

bb-replog.cgi is passed a QUERY_STRING environment variable with the following parameters:

HOSTSVC(thehostandservicetoreporton) STYLE(reportstyle:"crit","non-crit","all") ST(starttimeinsecondssince1-1-197000:00UTC) END(endtimeinsecondssince1-1-197000:00UTC)

The following non-standard parameters are handled by the bbgem version of bb-hist.cgi:

IP(IPaddressofhost-fordisplaypurposesonly) REPORTTIME(theREPORTTIME:settingforthishost) WARNPCT(theWARNPCT:setting)

The REPORTTIME and WARNPCT options are taken from the *bb-hosts(5)* definition for the host, or the defaults are used. These modify the availability calculation to handle reporting against agreed Service Level Agreements re. the time of day when the service must be available, and the agreed availability level.

19.3.3 OPTIONS

-env=FILENAME Loads environment from FILENAME before executing the CGI.

19.3.4 SEE ALSO

bb-hosts(5), *xymonserver.cfg(5)*, *hobbitsvc.cgi(1)*

19.4 HOBBITD_HISTORY

xymond_history - hobbitd worker module for logging status changes

19.4.1 SYNOPSIS

xymond_channel –channel=stachg hobbitd_history [options]

19.4.2 DESCRIPTION

xymond_history is a worker module for hobbitd, and as such it is normally run via the *xymond_channel(8)* program. It receives xymond status-change messages from the “stachg” channel via stdin, and uses these to update the history logfiles in a manner that is compatible with the standard Big Brother daemon, bbd.

19.4.3 OPTIONS

- histdir=DIRECTORY** The directory for the history files. If not specified, the directory given by the BBHIST environment is used.
- histlogdir=DIRECTORY** The directory for the historical status-logs. If not specified, the directory given by the BBHISTLOGS environment is used.
- debug** Enable debugging output.

19.4.4 ENVIRONMENT

BBALLHISTLOG This environment variable controls if the \$BBHIST/allevents logfile is updated. This file is used by the event-log display on the BB2 html page. If you do not use this, you can disable it by setting BBALLHISTLOGS=False.

BBHOSTHISTLOG This environment variable controls if the \$BBHIST/HOSTNAME logfile is updated. This file holds a list of all status changes seen for a single host, but is not used by any of the standard xymon tools. If you do not want to save this, you can disable it by setting BBHOSTHISTLOG=False.

SAVESTATUSLOG This environment variable controls if the historical status-logs are saved whenever a status change occurs. These logfiles are stored in the \$BBHISTLOGS directory, and are used for the detailed log-display of a status from the xymon “History” page. If you do not want to save these, you can disable it by setting SAVESTATUSLOG=False.

19.4.5 FILES

This module does not rely on any configuration files.

19.4.6 SEE ALSO

xymond_channel(8), *hobbitd(8)*, *hobbit(7)*

19.5 MSGCACHE

msgcache - Cache client messages for later pickup by xymonfetch

19.5.1 SYNOPSIS

msgcache [options]

19.5.2 DESCRIPTION

msgcache implements a xymon message cache. It is intended for use with clients which cannot deliver their data to the Xymon server in the normal way. Instead of having the client tools connect to the Hobbit server, msgcache runs locally and the client tools then deliver their data to the msgcache daemon. The msgcache daemon is then polled regularly by the *xymonfetch(8)* utility, which collects the client messages stored by msgcache and forwards them to the xymon server.

NOTE: When using msgcache, the **BBDISP** setting for the clients should be **BBDISP=127.0.0.1** instead of pointing at the real xymon server.

19.5.3 RESTRICTIONS

Clients delivering their data to msgcache instead of the real xymon server will in general not notice this. Specifically, the client configuration data provided by the Xymon server when a client delivers its data is forwarded through the xymonfetch / msgcache chain, so the normal centralized client configuration works.

However, other commands which rely on clients communicating directly with the xymon server will not work. This includes the **config** and **query** commands which clients may use to fetch configuration files and query the xymon server for a current status.

The **download** command also does not work with msgcache. This means that the automatic client update facility will not work for clients communicating via msgcache.

19.5.4 OPTIONS

-listen=IPADDRESS[:PORT] Defines the IP-address and portnumber where msgcache listens for incoming connections. By default, msgcache listens for connections on all network interfaces, port 1984.

-server=IPADDRESS[,IPADDRESS] Restricts which servers are allowed to pick up the cached messages. By default anyone can contact the msgcache utility and request all of the cached messages. This option allows only the listed servers to request the cached messages.

-max-age=N Defines how long cached messages are kept. If the message has not been picked up with N seconds after being delivered to msgcache, it is silently discarded. Default: N=600 seconds (10 minutes).

-daemon Run as a daemon, i.e. msgcache will detach from the terminal and run as a background task

-no-daemon Run as a foreground task. This option must be used when msgcache is started by *xymonlaunch(8)* which is the normal way of running msgcache.

-pidfile=FILENAME Store the process ID of the msgcache task in FILENAME.

-logfile=FILENAME Log msgcache output to FILENAME.

-debug Enable debugging output.

19.5.5 SEE ALSO

xymonfetch(8), *hobbit(7)*

19.6 BBRETEST-NET.SH

`bbretest-net.sh` - xymon network re-test tool

19.6.1 SYNOPSIS

`bbretest-net.sh`

19.6.2 DESCRIPTION

bbretest-net.sh is an extension script for xymon that runs on the network test server. It picks up the failing network tests executed by the *bbtest-net(1)* program, and repeats these tests with a faster test cycle than the normal bbtest-net schedule. This means that when the server recovers and the network service becomes available again, this is detected quicker resulting in less reported downtime.

Only tests whose first failure occurred within 30 minutes are included in the tests that are run by bbretest-net.sh. The 30 minute limit is there to avoid hosts that are down for longer periods of time to bog down bbretest-net.sh. You can change this limit with the “`--frequenttestlimit=SECONDS`” when you run bbtest-net.

19.6.3 INSTALLATION

This script runs by default from your *xymonlaunch.cfg(5)* file.

19.6.4 FILES

\$BBTMP/TESTNAME.LOCATION.status Temporary status file managed by bbtest-net with status of tests that have currently failed.

\$BBTMP/frequenttests.LOCATION Temporary file managed by bbtest-net with the hostnames that bbretest-net.sh should test.

19.6.5 SEE ALSO

`bbtest-net(1)`, `xymon(7)`, `hobbitlaunch.cfg(5)`

19.7 HOBBITD_HOSTDATA

xymond_hostdata - hobbitd worker module for storing historical client data

19.7.1 SYNOPSIS

xymond_channel –channel=clichg hobbitd_hostdata

19.7.2 DESCRIPTION

xymond_hostdata is a worker module for hobbitd, and as such it is normally run via the *hobbitd_channel(8)* program. Whenever a status column in xymon changes to an alert state (usually red, yellow or purple), this module receives a copy of the latest Xymon client data sent by the host, and stores it on disk. This allows you to review all of the data collected by the Hobbit client on the server around the time that a problem occurred. This can make troubleshooting incidents easier by providing a snapshot of the host status shortly before a problem became apparent.

Note: This module requires that *xymond(8)* is launched with the “–store-clientlogs” option enabled.

19.7.3 FILES

All of the host data are stored in the \$CLIENTLOGS directory, by default this is the \$BBVAR/hostdata/ directory.

19.7.4 SEE ALSO

xymond(8), *hobbitd_channel(8)*, *hobbit(7)*

19.8 ORCAHOBBIT

hgcmdorcaxymon - xymon client utility to grab data from ORCA

19.8.1 SYNOPSIS

orcaxymon **-orca=PREFIX** [options]

19.8.2 NOTICE

This utility is included in the client distribution for xymon 4.2. However, the backend module to parse the data it sends it **NOT** included in xymon 4.2. It is possible to use the generic Xymon NCV data handler in *xymond_rrd(8)* to process ORCA data, if you have an urgent need to do so.

19.8.3 DESCRIPTION

orcaxymon is an add-on tool for the xymon client. It is used to grab data collected by the ORCA data collection tool (orcallator.se), and send it to the Xymon server in NCV format.

orcaxymon should run from the client *hobbitlaunch(8)* utility, i.e. there must be an entry in the *clientlaunch.cfg(5)* file for orcaxymon.

19.8.4 OPTIONS

-orca=PREFIX The filename prefix for the ORCA data log. Typically this is the directory for the ORCA logs, followed by “orcallator”. The actual filename for the ORCA logs include a timestamp and sequence number, e.g. “orcallator-2006-06-20-000”. This option is required.

-debug Enable debugging output.

19.8.5 SEE ALSO

xymon(7), clientlaunch.cfg(5)

Chapter 20

Data Backend

20.1 HOBBITD_RRD

xymond_rrd - hobbitd worker module for updating xymon RRD files

20.1.1 SYNOPSIS

```
xymond_channel --channel=status hobbitd_rrd [options]
xymond_channel --channel=data hobbitd_rrd [options]
```

20.1.2 DESCRIPTION

xymond_rrd is a worker module for hobbitd, and as such it is normally run via the *hobbitd_channel(8)* program. It receives “status” and “data” messages from xymond via stdin, and updates the RRD databases used to generate trend-graphs.

Clients can send data to xymon using both status- and data- messages. So you will normally run two instances of this module, once for the “status” channel and once for the “data” channel.

xymond_rrd understands data sent by the LARRD 0.43c client-side scripts (the so-called “bottom-feeder” scripts). So you still want to install the LARRD bottom-feeders on the clients you monitor.

Note: For certain types of data, the RRD files used by xymon are incompatible with those generated by the Big Brother LARRD add-on. See the COMPATIBILITY section below.

20.1.3 OPTIONS

-debug Enable debugging output.

-rrddir= DIRECTORY Defines the directory where the RRD-files are stored. xymond_rrd will use the location pointed to by the BBRRDS environment if this option is not present.

-extra-script= FILENAME Defines the script that is run to get the RRD data for tests that are not built into xymond_rrd. You must also specify which tests are handled by the external script in the **-extra-tests** option. This option can only be given once, so the script must handle all of the external test-data. See the CUSTOM RRD DATA section below. Note that this is NOT needed if your custom graphs are generated by the NCV (Name Colon Value) module described below, it is only required for data where you have a custom script to parse the status message and extract the data that is put into the graph.

-extra-tests= TEST[,TEST] List of testnames that are handled by the external script. See the CUSTOM RRD DATA section below. Note that NCV graphs should NOT be listed here, but in the TEST2RRD environment variable - see below.

-processor=COMMAND Feed the raw RRD data into COMMAND via standard input. COMMAND must be a simple command with no options; if necessary, COMMAND can be a script that runs the real command with the necessary options. The data sent to COMMAND consists of lines of text of the form

dsname1[:dsname2]time:value1[:value2]rrdtype[key1][key2] First the dataset names, corresponding to the DS definitions in the RRD files. Next the timestamp of the update, followed by the values in the same order as the dsname-list. Third item is the hostname, fourth is the type of RRD file. After that there may be between 0-2 additional keys: For “disk” this is the filesystem mountpoint, for “tcp” tests it is the service name, for “tcp” “http” tests there is also the URL checked. Note that filesystem names and URLs have forward slash substituted by comma. See the xymond/rrd/*.c sourcecode for details, look for the “setupfn” function calls. An example:

```
la 1196115478:7 myhost la
la 1196115478:169 myhost procs
la 1196115478:1 myhost users
la 1196115478:244 myhost clock
pct:used 1196115478:63:62338144 myhost disk ,root
realmempct 1196115478:76 myhost memory real
realmempct 1196115478:0 myhost memory swap
realmempct 1196115478:20 myhost memory actual
sec 1196115478:0.03 tcp myhost conn
sec 1196115478:1.38 myhost tcp http www.xymon.com,
```

20.1.4 ENVIRONMENT

TEST2RRD Defines the mapping between a status-log columnname and the corresponding RRD database format. This is normally defined in the *xymonserver:cfg(5)* file.

BRRRDS Default directory where RRD files are stored.

NCV_testname Defines the types of data collected by the “ncv” module in xymond_rrd. See below for more information.

20.1.5 COLLECTED DATA

The following RRD-file datasets are generated by xymond_rrd:

la Records the CPU load average. Data is collected from the “cpu” status report. Requires that a xymon client is running on the monitored server.

disk Records the disk utilization. Data is collected from the “disk” status report. Requires that a xymon-compatible client is running on the monitored server.

memory Records memory- and swap-utilization. Data is collected from the “memory” status report. If no “memory” status is reported, it will use the data from the Win32 client “cpu” status report to generate this dataset. Requires that a xymon-compatible client is running on the monitored server.

netstat Records TCP and UDP statistics. Data is collected from the “netstat” status report; however, this data is often sent via the xymon “data” protocol, so there need not be a “netstat” column visible on the Xymon display. To get these data, the LARRD netstat bottom-feeder script must be running on the monitored server.

vmstat Records system performance metrics from the “vmstat” command. Data is collected from the “vmstat” status report; however, this data is often sent via the xymon “data” protocol, so there need not be a “vmstat” column visible on the Xymon display. To get these data, the LARRD vmstat bottom-feeder script must be running on the monitored server.

tcp Response-time metrics from all of the xymon network tests are recorded in the “tcp” RRD.

apache Apache server performance metrics, taken from the “apache” data report. See the description of the **apache** keyword in *bb-hosts(5)* for details.

sendmail Sendmail server performance metrics, taken from the “mailstats” output. To get these data, the LARRD sendmail bottom-feeder script must be running on the monitored server.

mailq Mail queue size. To get these data, the LARRD nmailq bottom-feeder script must be running on the monitored server.

bea BEA Weblogic performance data. This is an experimental set of data collected from BEA Weblogic servers via SNMP, by the “beastats” tool included with xymon.

iishealth IIS webserver performance data, collected by the “iishealth” script. This script is a client-side add-on available from the www.deadcat.net archive.

temperature Temperature data, collected with the temperature script from www.deadcat.net. To get these data, the temperature script must be running on the monitored server.

ntpstat Tracks the deviation between the local system time and an NTP server, using the output from the “ntpq -c rv” command. A simple script to collect these data is included in the xymon contrib/ directory.

citrix Tracks the number of active sessions on a Citrix server using the “query session” command. An extension for the BBNT client that generates data for this graph is in the xymon contrib/ directory.

20.1.6 CUSTOM RRD DATA IN NAME-COLON-VALUE (NCV) FORMAT

Many data-collection scripts report data in the form “NAME : value” or “NAME = value”. So a generic module in xymond_rrd allows for easy tracking of this type of data.

The “ncv” module will automatically detect all occurrences of a “NAME : value” or “NAME = value” string in a status message, and generate an RRD file holding all of the name/value data found in the message. The colon- or equal-sign must be present - if there is only whitespace, this module will fail.

Only the valid letters (A-Z, a-z) and digits (0-9) are used in the dataset names; whitespace and other characters are stripped off automatically. Only the first 19 characters of a dataset name are used (this is an RRD limitation). Underscore ‘_’ is not allowed, even though RRDtool permits this, and will be stripped from the name.

Note that each “NAME : value” must be on a line by itself. If you have a custom script generating the status- or data-message that is fed into the NCV handler, make sure it inserts a newline before each of the data-items you want to track.

To enable the ncv module for a status, add a “COLUMNNAME=ncv” to the TEST2RRD setting and the COLUMNNAME to the GRAPHS setting in *xymonserver.cfg(5)*, then restart xymon. Xymon will now send all status-messages for the column COLUMNNAME through the xymond_rrd ncv-handler.

The name of the RRD file will be COLUMNNAME.rrd.

By default, all of the datasets are generated as the RRD type “DERIVE” which works for all types of monotonically increasing counters. If you have data that are of the type GAUGE, you can override the default via an environment variable NCV_COLUMNNAME.

E.g. if you are using the bb-mysqlstatus script from www.deadcat.net to collect data about your MySQL server, it generates a report in the column called “mysql”. One data item is the average number of queries/second, which must be logged in the RRD file as type “GAUGE”. To do that, add the following to xymonserver.cfg:

NCV_mysql=”Queriespersecondavg:GAUGE” If you have multiple datasets that you myst define, add them to the environment variable separated by commas, e.g.

NCV_mysql=”Uptime:NONE,Queriespersecondavg:GAUGE”

The dataset type “NONE” used above causes xymond_rrd to ignore this data, it is not included in the RRD file.

You can use “*” as the dataset name to match all datasets not listed. E.g.

`NCV_weather=“Rain:DERIVE,:GAUGE”` will cause the “Rainfall” dataset to be of type DERIVE, and all others of type GAUGE. If you want to track only a few of the variables in your data, you can use “*:NONE” to drop any dataset not explicitly listed.

For a more detailed “how to” description, see the on-line HTML documentation of “How to create graph custom data” available in the Help menu section on your xymon server.

20.1.7 CUSTOM RRD DATA VIA SCRIPTS

xymond_rrd provides a simple mechanism for adding custom graphs to the set of data collected on your xymon server. By adding the “–extra-script” and “–extra-tests” options, data reported to Xymon from selected tests are passed to an external script, which can define the RRD data-sets to store in an RRD file.

NOTE: For performance reasons, you should not use this mechanism for large amounts of data. The overhead involved in storing the received message to disk and launching the script is significantly larger than the normal xymond_rrd overhead. So if you have a large number of reports for a given test, you should consider implementing it in C and including it in the hobbitd_rrd tool.

Apart from writing the script, You must also add a section to *xymongraph.cfg(5)* so that *xymongraph.cgi(1)* knows how to generate the graph from the data stored in the RRD file. To make the graphs actually show up on the status-page and/or the “trends” page, add the name of the new graph to the TEST2RRD and/or GRAPHS setting in *xymon-server.cfg(5)*.

The script is invoked for each message that arrives, where the test-name matches one of the testnames given in the “–extra-tests” option. The script receives three commandline parameters:

Hostname The name of the host reporting the data.

Testname The name of the test being reported.

Filename File containing the data that was reported. This file is generated for you by xymond_rrd, and is also deleted automatically after your script is finished with it.

The script must process the data that is reported, and generate the following output:

RRD data-set definitions For each dataset that the RRD file holds, a line beginning with “DS:” must be output.

If multiple data-sets are used, print one line for each dataset. Data-set definitions are described in the *rrd-create(1)* documentation, but a common definition for e.g. tracking the number of users logged on would be “DS:users:GAUGE:600:0:U”. “users” is the name of the dataset, “GAUGE” is the datatype, “600” is the longest time allowed between updates for the data to be valid, “0” is the minimum value, and “U” is the maximum value (a “U” means “unknown”).

RRD filename The name of the RRD file where the data is stored. Note that xymon stores all RRD files in host-specific directories, so unlike LARRD you should not include the hostname in the name of the RRD file.

RRD values One line, with all of the data values collected by the script. Data-items are colon-delimited and must appear in the same sequence as your data-set definitions, e.g. if your RRD has two datasets with the values “5” and “0.4” respectively, then the script must output “5:0.4” as the RRD values. In some cases it may be useful to define a dataset even though you will not always have data for it. In that case, use “U” (unknown) for the value.

If you want to store the data in multiple RRD files, the script can just print out more sequences of data-set definitions, RRD filenames and RRD values. If the data-set definitions are identical to the previous definition, you need not print the data-set definitions again - just print a new RRD filename and value.

The following sample script for tracking weather data shows how to use this mechanism. It assumes the status message include lines like these:

```
green Weather in Copenhagen is FAIR
```

```
Temperature: 21 degrees Celsius
Wind: 4 m/s
Humidity: 72 %
Rainfall: 5 mm since 6:00 AM
```

A shell-script to track all of these variables could be written like this:

```
#!/bin/sh

# Input parameters: Hostname, testname (column), and messagefile
HOSTNAME="$1"
TESTNAME="$2"
FNAME="$3"

if [ "$TESTNAME" = "weather" ]
then
    # Analyze the message we got
    TEMP='grep "^\$Temperature:" $FNAME | awk '{print $2}''
    WIND='grep "^\$Wind:" $FNAME | awk '{print $2}''
    HMTY='grep "^\$Humidity:" $FNAME | awk '{print $2}''
    RAIN='grep "^\$Rainfall:" $FNAME | awk '{print $2}''

    # The RRD dataset definitions
    echo "DS:temperature:GAUGE:600:-30:50"
    echo "DS:wind:GAUGE:600:0:U"
    echo "DS:humidity:GAUGE:600:0:100"
    echo "DS:rainfall:DERIVE:600:0:100"

    # The filename
    echo "weather.rrd"

    # The data
    echo "$TEMP:$WIND:$HMTY:$RAIN"
fi

exit 0
```

20.1.8 COMPATIBILITY

Some of the RRD files generated by xymond_rrd are incompatible with the files generated by the Big Brother LARRD add-on:

vmstat The vmstat files with data from Linux based systems are incompatible due to the addition of a number of new data-items that LARRD 0.43 do not collect, but xymond_rrd does. This is due to changes in the output from the Linux vmstat command, and changes in the way e.g. system load metrics are reported.

netstat All netstat files from LARRD 0.43 are incompatible with xymond_rrd. The netstat data collected by LARRD is quite confusing: For some types of systems LARRD collects packet-counts, for others it collects byte- counts.

xymond_rrd uses a different RRD file-format with separate counters for packets and bytes and tracks whatever data the system is reporting.

20.1.9 SEE ALSO

xymond_channel(8), hobbitd(8), hobbitserver.cfg(5), hobbit(7)

Chapter 21

Miscellaneous programs

21.1 HOBBITLAUNCH

`xymonlaunch` - Master program to launch other xymon programs

21.1.1 SYNOPSIS

`xymonlaunch [options]`

21.1.2 DESCRIPTION

`xymonlaunch(8)` is the main program that controls the execution and scheduling of all of the components in the xymon system.

`xymonlaunch` allows the administrator to add, remove or change the set of xymon applications and extensions without restarting Xymon - `xymonlaunch` will automatically notice any changes in the set of tasks, and change the scheduling of activities accordingly.

`xymonlaunch` also allows the administrator to setup specific logfiles for each component of the xymon system, instead of getting output from all components logged to a single file.

21.1.3 OPTIONS

-env=FILENAME Loads the environment from FILENAME before starting other tools. The environment defined by FILENAME is the default, it can be overridden by the ENVFILE option in `xymonlaunch.cfg(5)`

-config=FILENAME This option defines the file that `xymonlaunch` scans for tasks it must launch. A description of this file is in `xymonlaunch.cfg(5)`

The default tasklist is `/etc/xymonlaunch.cfg`

-log=FILENAME Defines the logfile where `xymonlaunch` logs information about failures to launch tasks and other data about the operation of `xymonlaunch`. Logs from individual tasks are defined in the `xymonlaunch.cfg` file. By default this is logged to stdout.

-pidfile=FILENAME Filename which `xymonlaunch` saves its own process-ID to. Commonly used by automated start/stop scripts.

-verbose Logs the launch of all tasks to the logfile. Note that the logfile may become quite large if you enable this.

-dump Just dump the contents of the `xymonlaunch.cfg` file after parsing it. Used for debugging.

-debug Enable debugging output while running.

-no-daemon `xymonlaunch` normally detaches from the controlling tty and runs as a background task. This option keeps it running in the foreground.

21.1.4 STARTING TASKS

`xymonlaunch` will read the configuration file and start all of the tasks listed there.

If a task completes abnormally (i.e. terminated by a signal or with a non-zero exit status), then `xymonlaunch` will attempt to restart it 5 times. If it still will not run, then the task is disabled for 10 minutes. This will be logged to the `xymonlaunch` logfile.

If the configuration file changes, `xymonlaunch` will re-read it and notice any changes. If a running task was removed from the configuration, then the task is stopped. If a new task was added, it will be started. If the command used for a task changed, or it was given a new environment definition file, or the logfile was changed, then the task is stopped and restarted with the new definition.

21.1.5 SEE ALSO

xymonlaunch.cfg(5), hobbit(7)

Appendix A

Making of this book

A.1 Using Mercurial's book template to build our bookor

This book is converted into latex version from troff version xymon manpages.

It has a Makefile to automate the process book making.

1. Get xymon manpages.
2. rtf2latex
3. fix latex syntax issue.
4. Prepare Diagram in InkScape.
5. Indexes
6. Table of contents
7. References
8. List of figures
9. List of Tables
10. Indexes
11. Download a recent source tarball from <http://www.selenic.com/mercurial/download>.
12. Unpack the tarball:

```
1 gzip -dc mercurial-version.tar.gz | tar xf -
```

13. Go into the source directory and run the installer script. This will build Mercurial and install it in your home directory.

```
1 cd mercurial-version
2 python setup.py install --force --home=$HOME
```

Once the install finishes, Mercurial will be in the `bin` subdirectory of your home directory. Don't forget to make sure that this directory is present in your shell's search path.

You will probably need to set the `PYTHONPATH` environment variable so that the Mercurial executable can find the rest of the Mercurial packages. For example, on my laptop, I have set it to `/home/bos/lib/python`. The exact path that you will need to use depends on how Python was built for your system, but should be easy to figure out. If you're uncertain, look through the output of the installer script above, and see where the contents of the `mercurial` directory were installed to.

A.2 On Windows

Building and installing Mercurial on Windows requires a variety of tools, a fair amount of technical knowledge, and considerable patience. I very much *do not recommend* this route if you are a “casual user”. Unless you intend to hack on Mercurial, I strongly suggest that you use a binary package instead.

If you are intent on building Mercurial from source on Windows, follow the “hard way” directions on the Mercurial wiki at <http://www.selenic.com/mercurial/wiki/index.cgi/WindowsInstall>, and expect the process to involve a lot of fiddly work.

Appendix B

Open Publication License

Version 1.0, 8 June 1999

B.1 Requirements on both unmodified and modified versions

The Open Publication works may be reproduced and distributed in whole or in part, in any medium physical or electronic, provided that the terms of this license are adhered to, and that this license or an incorporation of it by reference (with any options elected by the author(s) and/or publisher) is displayed in the reproduction.

Proper form for an incorporation by reference is as follows:

Copyright (c) year by *author's name or designee*. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, vx.y or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

The reference must be immediately followed with any options elected by the author(s) and/or publisher of the document (see section [B.6](#)).

Commercial redistribution of Open Publication-licensed material is permitted.

Any publication in standard (paper) book form shall require the citation of the original publisher and author. The publisher and author's names shall appear on all outer surfaces of the book. On all outer surfaces of the book the original publisher's name shall be as large as the title of the work and cited as possessive with respect to the title.

B.2 Copyright

The copyright to each Open Publication is owned by its author(s) or designee.

B.3 Scope of license

The following license terms apply to all Open Publication works, unless otherwise explicitly stated in the document.

Mere aggregation of Open Publication works or a portion of an Open Publication work with other works or programs on the same media shall not cause this license to apply to those other works. The aggregate work shall contain a notice specifying the inclusion of the Open Publication material and appropriate copyright notice.

Severability. If any part of this license is found to be unenforceable in any jurisdiction, the remaining portions of the license remain in force.

No warranty. Open Publication works are licensed and provided "as is" without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement.

B.4 Requirements on modified works

All modified versions of documents covered by this license, including translations, anthologies, compilations and partial documents, must meet the following requirements:

1. The modified version must be labeled as such.
2. The person making the modifications must be identified and the modifications dated.
3. Acknowledgement of the original author and publisher if applicable must be retained according to normal academic citation practices.
4. The location of the original unmodified document must be identified.
5. The original author's (or authors') name(s) may not be used to assert or imply endorsement of the resulting document without the original author's (or authors') permission.

B.5 Good-practice recommendations

In addition to the requirements of this license, it is requested from and strongly recommended of redistributors that:

1. If you are distributing Open Publication works on hardcopy or CD-ROM, you provide email notification to the authors of your intent to redistribute at least thirty days before your manuscript or media freeze, to give the authors time to provide updated documents. This notification should describe modifications, if any, made to the document.
2. All substantive modifications (including deletions) be either clearly marked up in the document or else described in an attachment to the document.
3. Finally, while it is not mandatory under this license, it is considered good form to offer a free copy of any hardcopy and CD-ROM expression of an Open Publication-licensed work to its author(s).

B.6 License options

The author(s) and/or publisher of an Open Publication-licensed document may elect certain options by appending language to the reference to or copy of the license. These options are considered part of the license instance and must be included with the license (or its incorporation by reference) in derived works.

- A To prohibit distribution of substantively modified versions without the explicit permission of the author(s). “Substantive modification” is defined as a change to the semantic content of the document, and excludes mere changes in format or typographical corrections.

To accomplish this, add the phrase “Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.” to the license reference or copy.

- B To prohibit any publication of this work or derivative works in whole or in part in standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

To accomplish this, add the phrase “Distribution of the work or derivative of the work in any standard (paper) book form is prohibited unless prior permission is obtained from the copyright holder.” to the license reference or copy.

Bibliography

[Wau] Tim Waugh. patchutils—programs that operate on patch files. <http://cyberelk.net/tim/patchutils/>.

[Yan] T.J Yang. System monitoring with hobbit. http://en.wikibooks.org/wiki/System_Monitoring_with_Hobbit.

Index

-area=NAME, 86
-debug, 86
-env=FILENAME, 86
-nacklevel=NUMBER, 86
PYTHONPATH environment variable, 200
\$BBHOME environment variable, 183
BBHOME environment variable, 183
BBTMR environment variable, 183
bb-hosts command, 30
bbcmand command, 141
bbcbotest.cfg command, 143, 146, 147
bb command, 165
hostinfo.csv command, 152
logfetch command, 180
trimhistory command, 128
xymon-alerts.cfg command, 30
xymon-clients.cfg command, 30
xymongraph.cfg command, 30, 144, 148
xymonlaunch.cfg command, 30, 143, 157
xymonserver.cfg command, 30

ACKUNTILMSG, 57
ALERTCOLORS, 54
ALERTREPEAT, 54

BB, 53
bb-hosts, 58
BBACKS, 53
BBALLHISTLOG, 55
bbd, 67
BBDATA, 53
BBDATEFORMAT, 57
BBDISABLED, 53
BBDISP, 54
BBDISPLAYS, 54
BBGEN, 53
BBGENREOPTS, 58
BBGENSNAPOPTS, 58
BBGHOSTS, 54
BBHELPSSKIN, 53
BBHIST, 53
BBHISTTEXT, 58
BBHISTLOGS, 53

BBHOME, 52, 86
BBHOSTHISTLOG, 55
BBHOSTS, 53
bbhostshow, 111
BBHTML, 53
BBLOCATION, 56
BBLOGS, 53
BBLOGSTATUS, 52
BBMAXMSGSPERCOMBO, 54
BBMENUSKIN, 53
BBMKBB2EXT, 58
BBNOTES, 52
BBNOTESSIONSKIN, 53
BBPAGE, 54
BBPAGERS, 54
BBPORT, 54
BBREP, 52
BBREPURL, 53
BBREPWARN, 58
BBROUTERTEXT, 56
BBRRDS, 53
BBSSTITLE, 57
BBSERVERCGIURL, 53
BBSERVERHOSTNAME, 51
BBSERVERIP, 51
BBSERVERLOGS, 52
BBSERVERROS, 51
BBSERVERROOT, 52
BBSERVERWWWNAME, 51
BBSERVERWWWURL, 53
BBSKIN, 53
BBSLEEPBETWEENMSGS, 54
BBSNAP, 52
BBSNAPURL, 54
BBTMR, 52
BBVAR, 52
BBWAP, 54
BBWEB, 53
BBWEBHOST, 53
BBWEBHOSTURL, 53
BBWEBHTMLLOGS, 53
BBWWW, 52

CGI_ACK_OPTS, 84
 CGI_ACKINFO_OPTS, 84
 CGI_CSVINFO_OPTS, 84
 CGI_DATEPAGE_OPTS, 84
 CGI_ENADIS_OPTS, 84
 CGI_EVENTLOG_OPTS, 84
 CGI_FINDHOST_OPTS, 84
 CGI_HIST_OPTS, 84
 CGI_HOBBITCOLUMN_OPTS, 84
 CGI_HOBBITCONFREPORT_OPTS, 84
 CGI_HOBBITGRAPH_OPTS, 84
 CGI_HOSTGRAPHS_OPTS, 84
 CGI_LNKEDIT_OPTS, 84
 CGI_LNKVIEW_OPTS, 84
 CGI REP_OPTS, 84
 CGI_REPLOG_OPTS, 84
 CGI_SNAPSHOT_OPTS, 84
 CGI_SVC_OPTS, 84
 CGI_SVCHIST_OPTS, 84
 CGIBINURL, 54
 CLIENTSVCS, 57
 COLUMNDOCURL, 54
 conn, 65
 CONNTEST, 56

 dialup, 61
 dig, 67
 dispinclude filename, 59
 dns, 67
 DOCOMBO, 54
 DOTHEIGHT, 57
 DOTWIDTH, 57

 environment variables
 PYTHONPATH, 200
 \$BBHOME, 183
 BBHOME, 183
 BBTMP, 183
 Example, 78

 FPING, 56
 FQDN, 51

 GRAPHS, 56

 HIDEHTTP, 70
 HOBBITLOGO, 57
 HOLIDAYFORMAT, 57
 HOLIDAYS, 52
 HTMLCONTENTTYPE, 52

 include filename, 59
 INFOCOLUMN, 52

IPTEST_2_CLEAR_ON_FAILED_CONN, 56

 ldap, 71
 ldaps, 71
 ldapyellowfail, 71

 MAIL, 55
 MAILC, 55
 MAXMSG_CLICHG, 55
 MAXMSG_CLIENT, 55
 MAXMSG_DATA, 55
 MAXMSG_ENADIS, 55
 MAXMSG_NOTES, 55
 MAXMSG_PAGE, 55
 MAXMSG_STACHG, 55
 MAXMSG_STATUS, 55
 MKBB2COLREPEAT, 57
 MKBACKFONT, 57
 MKBCOLFONT, 57
 MKBLOCAL, 57
 MKBREMOTE, 57
 MKBBROWFONT, 57
 MKBBSUBLOCAL, 57
 MKBTITLE, 57

 NETFAILTEXT, 56
 netinclude filename, 59
 nobb2, 61
 noclear, 59, 64
 noconn, 65
 nodisp, 61
 NONETPAGE, 56
 noping, 65
 NORRDDISKS, 56
 nosslcert, 64
 notrace, 66
 ntp, 67
 NTPDATE, 57

 OKCOLORS, 54

 PAGELEVELS, 54
 PINGCOLUMN, 52
 prefer, 59

 QUERY_STRING, 86

 RPCINFO, 57
 RRDDISKS, 56
 RRDHEIGHT, 52
 RRDWIDTH, 52

 SAVESTATUSLOG, 55

Service Level Agreement, [96](#)

SLA, [96](#)

SUMMARY_SET_BKG, [57](#)

SVCCODES, [55](#)

TEST2RRD, [56](#)

testip, [64](#)

trace, [66](#)

TRACEROUTE, [56](#)

TRENDSCOLUMN, [52](#)

TRENDSECONDS, [52](#)

WEEKSTART, [52](#)

WMLMAXCHARS, [57](#)