CODE MAGUS

Serfboard Configuration Guide and Reference
Version 1

CML00023-01

Code Magus Limited (England reg. no. 4024745)
Number 6, 69 Woodstock Road
Oxford, OX2 6EY, United Kingdom
www.codemagus.com

Business Partner IBM™

August 16, 2016

Contents

1 Introduction **3**
1.1 *Serfboard* overview . . . 3
1.2 About this Manual . . . 3
1.3 Execution Overview . . . 4

2 Starting the `serfboard` server **5**
2.1 Synopsis . . . 5
2.2 Parameters . . . 5
2.3 Environmental Variables . . . 6

3 Common Syntactical Elements **7**
3.1 Comments . . . 7
3.2 *Integer* . . . 7
3.3 *Number* . . . 7
3.4 *Identifier* . . . 8
3.5 *String* . . . 8
3.6 Filename . . . 8

4 Configuration **10**
4.1 Command interface . . . 10
4.2 Reserved Words . . . 11
4.3 Environment Variables . . . 11
4.4 Commands . . . 11
4.4.1 Background Command . . . 12
4.4.2 Close Command . . . 12
4.4.3 Display Command . . . 13
4.4.4 Exit Command . . . 15
4.4.5 Help Command . . . 15
4.4.6 Load Command . . . 15
4.4.7 Open Command . . . 16
4.4.8 Set Command . . . 20
4.4.9 Shutdown Command . . . 20
4.4.10 Start Command . . . 20

5 Dashboard Configuration **21**
5.1 Overview . . . 21
5.2 Reserved Words . . . 21
5.3 Include Directive . . . 21
5.4 Dashboard Definition . . . 21
5.5 *DashboardObjects* . . . 22
5.6 *Image* . . . 22
5.7 *Link* . . . 23
5.8 Panel . . . 24
5.8.1 Control . . . 25
5.8.2 Delay . . . 25

5.8.3 Description 26
5.8.4 Height 26
5.8.5 Origin 26
5.8.6 Options 26
5.8.7 Title 27
5.8.8 Width 27
5.8.9 Metrics 28
5.8.10 Dashboard 29
5.8.11 Example Panel Configuration 29
5.9 Embedded Dashboards 29

6 Metric Configuration **31**
6.1 Overview 31
6.2 Reserved Words 31
6.3 Include Directive 31
6.4 *Metrics definition* 31
6.4.1 *MetricStatement* 32
6.4.2 Example Metric Definition 33

7 Accessing Dashboards **34**

A *Serfboard* Complete example **35**
A.1 Introduction 35
A.2 Dashboard Specification 36
A.3 Dashboard Configuration 37
A.4 Metrics Configuration 39
A.5 Dashboard View 42

B Invoking Web Page Example **43**

1 Introduction

1.1 *Serfboard* overview

The *Serfboard* system enables a user to monitor, and possibly interact with, a running system or systems via a web based graphical dashboard. Dashboards of this nature that show a unified view of key performance indicators of different systems are extremely effective in the management of networked systems testing or systems stress testing. In stress testing, being interactive, the user can change various thresholds and key values in order to watch the effect of different loads on the system under stress.

The *Serfboard* system comprises the following parts:

- The core server (`serfboard`) which is a daemon process (executes in the background) that receives metric data, stores it and supplies it to the web application on request.
- The display instrument plug-in programs and web application which are responsible for rendering the final dashboard as viewed by the user. The web application continuously requests updates of data from the server.
- Metric feed probes and feed library. The metric feed probes are responsible for querying a specific system that supplies metrics (for example SNMP), extracting the relevant data, converting it to the *Serfboard* format and forwarding the metric data on the the server. There is also a C programming library interface available for development of user written metric feeder programs. See the following for more information about supplied probes:
  - cmlxaixp: AIX Performance Metric Probe [6]
  - cmlxsolp: Solaris Performance Metric Probe [9]
  - cmlxsnmp: SNMP Metric Probe [5]
  - cmlxwinp: Windows Performance Metric Probe [7]
  - cmlxwasp: Websphere Application Server Performance Metric Probe [8]

1.2 About this Manual

This manual explains how to run and configure a Dashboard Server (`serfboard`) in order to be ready to accept metric data from various sources and present the modified metric state data and dashboard layout to a requesting web enabled dashboard. It should be used together with the following manuals:

- CML00024-01 Serfboard Instruments Guide and Reference Version 1 [2]

Helps the dashboard designer choose the rendering method for the particular metric data that is to be displayed.

- CML00027-01 Serfboard User Guide Version 1 [4]

  This explains the web application and how to navigate around a dashboard.

It is assumed that *Serfboard*, which includes the `serfboard` server and web application, have been correctly installed as per the installation manual, Serfboard Installation Guide and Reference Version 1 [3].

1.3 Execution Overview

A `serfboard` instance is started with out any configuration. It is configured through its command line interface, see section 4 on page 10. Dashboards are loaded via the command interface from dashboard configuration files and each dashboard loaded has a unique name to identify the dashboard. The definition of the metrics to render on the dashboards are loaded from metric configuration files and are grouped together with a group name. *Serfboard* will only accept incoming metrics that are defined in these metric configuration files. The user, requests a dashboard by a dashboard name. This may be by clicking a configured link or filling in a web form. This dashboard name as configured in the *Serfboard* system defines the dashboard layout and the metrics it will render to the user.

A dashboard comprises rectangular panels defined in the `serfboard` dashboard configuration (see section 5 on page 21). Each panel continuously requests an update of metric data from `serfboard`. The `serfboard` types configuration (see section 6 on page 31) defines the metric data that is available for rendering within a dashboard defined by the group. In other words the incoming metric source data is mapped within a group to a panel on a dashboard via the the types and panel configuration files as explained in this manual.

Furthermore each panel has an associated rendering instrument, which is responsible for rendering the metrics graphically. Examples of a rendering instrument are speedometers; graphs; and bar charts. Instruments accept only certain types of metrics on one or more channels. For example a graph instrument can render one or more curves on the same axis, each one from a different channel, and a bar graph accepts a metric that includes the name and value of each bar that it should render. They are explained in more detail in Serfboard Instruments Guide and Reference Version 1 [2].

When configuring `serfboard` using the panel and type configuration files a dashboard designer must understand the type of each metric received by `serfboard` and what type of metric and how many a display instrument can render. The incoming metrics, the panel configuration and the types configuration are all identified by the group name.

2 Starting the `serfboard` server

2.1 Synopsis

The `serfboard` server is started from the command line and if invoked with the `--help` parameter will show the available options:

```
Code Magus Limited SerfBoard V1.0: build 2011-01-05-20.22.00
[./bin/serfboard] $Id: serfboard.c,v 1.99 2010/12/07 10:22:47 janvlok Exp $
Copyright (c) 2009 by Code Magus Limited. All rights reserved.
 [Contact: stephen@codemagus.com].
Usage: serfboard [OPTION...]
  -c, --command=<command>     Command to pass to command process
  -v, --verbose               Verbose output
  -t, --trace                 Trace output

Help options:
  -?, --help                  Show this help message
  --usage                     Display brief usage message
```

2.2 Parameters

The parameters are explained below:

- `-c, --command=<command>`

  This optional parameter is a command to be passed to the command interface at startup.

- `-l, --log-file={|<filename>}`

  This optional parameter names the output replay file. All incoming metrics are written to this file and can be replayed back to `serfboard` with the program `sbreplay`. This is used in reviewing a dashboard that has been run or for testing. If not specified then no replay file is created.

- `-v, --verbose`

  This optional parameter causes verbose program diagnostic trace to be output.

- `-t, --trace`

  This optional parameter causes very verbose program diagnostic trace to be output.

- `-?, --help`

  This optional parameter prints the program's options and description to stderr.

- `--usage`

  This optional parameter prints a short list of the program's options to stderr.

2.3 Environmental Variables

The following environmental variables are required by *Serfboard*:

- `CODEMAGUS_HOME`

  Specify the path for the Codemagus software installation..

3 Common Syntactical Elements

The syntactical elements of the grammars that make up the commands and configuration files used in `serfboard` comprise reserved words, identifiers, string literals, comments, integers and numbers. All grammar is free format and white spaces have no grammatical meaning except where they might appear within string literals.

The elements common to the command and configuration grammars are explained in the following sub-sections.

3.1 Comments

Comments are introduced by using a double minus ("`--`") and continue up to the end of the current input line.

Examples:

```
-- File: atmams.cmd
-- Commands for Serfboard Dashboard atmams
--
```

3.2 *Integer*

An *Integer* consists of a nonempty sequence of decimal digits.

Examples:

```
1234
0
```

3.3 *Number*

A *Number* consists of a nonempty sequence of decimal digits that

- possibly contains a radix character (decimal point '.').
- is optionally followed by a decimal exponent; consisting of an 'E' or 'e' followed by an optional plus or minus sign followed by a nonempty sequence of decimal digits that indicates multiplication by a power of 10.

Examples:

```
1234
0.001
1.2
123.45E-12
```

3.4 *Identifier*

An *Identifier* is case sensitive and starts with a letter which can be followed by any number of letters, digits, or the under-score character.

Examples:

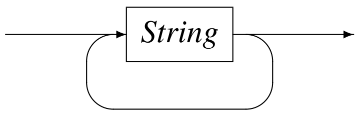
```
atm_ops  router_metrics_2 atmams
```

3.5 *String*

A *String* is:

- any sequence of characters (except double quotes and the newline character) enclosed by double quotes.
- any sequence of characters (except single quotes and the newline character) enclosed by single quotes.

Strings cannot span source text or configuration file lines, but they may be concatenated.



Examples:

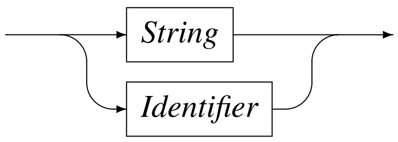
```
"GigabitEthernet0/0 In Octets"
'$Revision: 1.75 $'
"String 1 and " "String 2"
```

Although a *String* can not span across two lines it may be split into two or more strings where each distinct portion is on a separate line. In the following example the two strings are concatenated together to form one string used as the description.

```
description( "This is an example of two strings across two lines "
             "that will be concatenated together to form one string."
           )
```

3.6 Filename

Filename



A *Filename* is usually written as a *String* but may also be an *Identifier*. Most importantly a *Filename* must conform to any constraints of the underlying file system.

4 Configuration

The configuration of dashboards, input metrics and parameters are done via commands to `serfboard`. This section describes the `serfboard` command interface and commands.

The dashboard and metric configuration files are explained fully in the following sections, dashboards in section 5 on page 21 and metrics in section 6 on page 31. An explanation of common components and configuration syntax is explained in the previous section 3 on page 7.

4.1 Command interface

`serfboard` is configured from commands via its command interface. There are various ways to present commands to the command interface. They are:

- Standard input

  When `serfboard` is run in the foreground, the user directly communicates with the `serfboard` command interface via the terminal. Note that if `serfboard` is started in the background immediately, this channel is no longer available even if `serfboard` is subsequently brought back to the foreground.

- Network

  A TCP/IP interface to the `serfboard` command port. Note the TCP/IP port must have been configured, see (section 4.4.7 on page 16).

- Command File.

  A file containing commands. If an unrecognised input is passed to `serfboard`, it will assume that this could be the name of a file containing commands. It will attempt to open this file and process it. `serfboard` uses the CML library `cmdname` for validating the command file name and path resolution; see cmdname: Command Name Resolver Library Version 1 [10] for more information on how command files are named and resolved to an actual path name.

- Command Line Parameter

  A single command at start up, for example:

  ```
  serfboard -c atmams.cmd
  ```

  `atmams.cmd` is a file containing commands for `serfboard`.

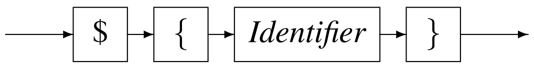
4.2 Reserved Words

Reserved words have a special meaning in terms of directing the parsing of commands. The reserved words are:

`all`	`exit`	`legacy`	`probe`
`background`	`feeder`	`msglevel`	`status`
`bind`	`group`	`metric`	`set`
`cgi`	`help`	`metrics`	`shutdown`
`close`	`host`	`notset`	`start`
`command`	`interface`	`open`	`trace`
`dashboard`	`load`	`output`	`verbose`
`display`	`log`	`port`	

4.3 Environment Variables

EnvironmentVariable

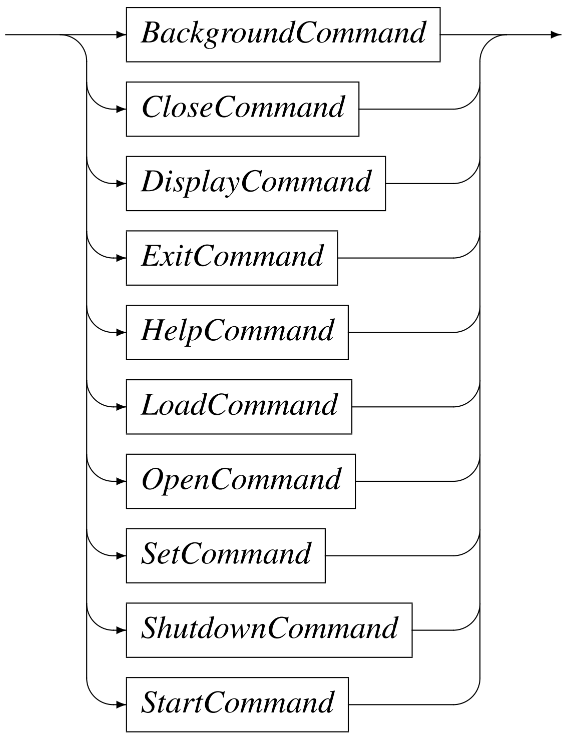


Environment variables are expanded to their value when encountered in command input text.

4.4 Commands

`serfboard` is configured from commands presented to its command interface. Input is either a single command or the name of a command file. Commands are explained in detail in the following sections. A command file is a text based file that consists of one or more commands, where each command is on a separate line. Typically `serfboard` requires multiple commands to be effectively configured so commands are often written as a logical group in a command file. A command file name is validated using the library cmdname [10].

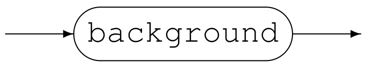
command



4.4.1 Background Command

This command informs `serfboard` that it is running in the back ground.

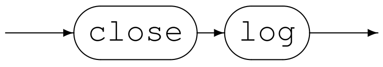
BackgroundCommand



4.4.2 Close Command

This command closes an open log file.

CloseCommand



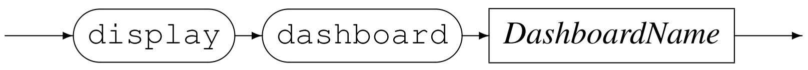
Example:

```
serfboard>close log
Log "text(/home/jan/replay/atmams.log,mode=a)" closed, record count = 11
```

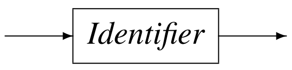
4.4.3 Display Command

This command is used to display:

- dashboards:



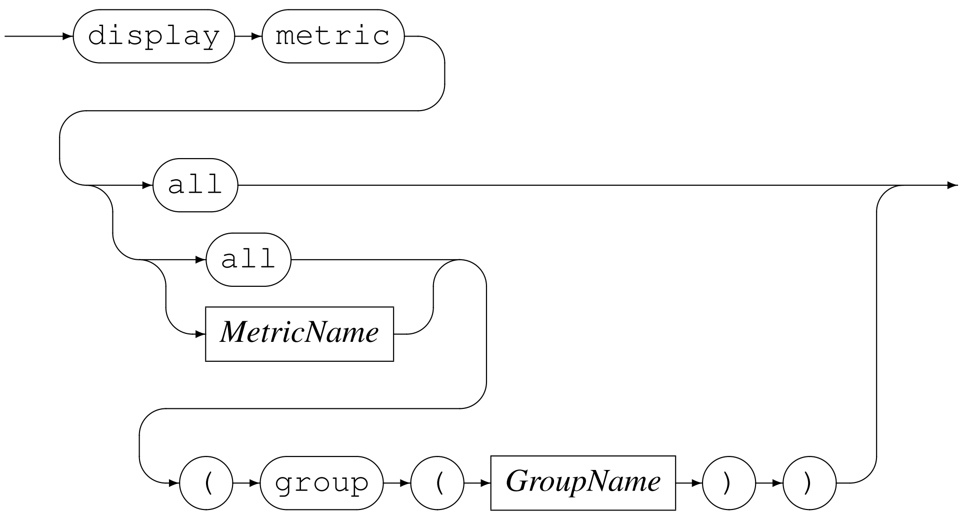
DashboardName



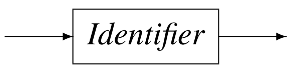
Example:

```
serfboard>display dashboard serfboard_demo01
dashboard serfboard_demo01 (
description("SerfBoard Demo 01")
image (origin(0,1) width(281) height(47)
    url("/serfboard/html/images/codemagus_logo.jpg"))
.
.
.
)
```

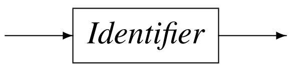
- metrics:



GroupName



MetricName



Examples:

- Display all the metrics defined:

```
serfboard>display metric all
metric_group serfboard_demo01 (
metric Sessions (
   title("Sessions Offered")
   description("Sessions offered"))
.
.
.
)
metric_group serfboard_demo02 (
metric Sessions (
   average_window(120)
   title("Sessions Offered")
   description("Sessions Offered"))
.
.
.
)
metric_group serfboard_demo03 (
metric cpu_tot_idle (
   average_window(180)
   title("Idle")
   description("cpu idle"))
.
.
.
)
```

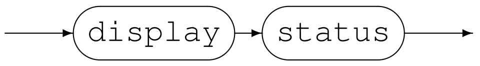
- Display all the metrics defined in group `serfboard_demo02`:

```
serfboard>display metric all (group(serfboard_demo02))
metric_group serfboard_demo02 (
metric Sessions (
   average_window(120)
   title("Sessions Offered")
   description("Sessions Offered"))
.
.
.
)
```

- Display metric `Sessions` in group `serfboard_demo01`:

```
serfboard>display metric Sessions (group(serfboard_demo01))
metric Sessions (
   title("Sessions Offered")
   description("Sessions offered"))
```

- `serfboard` status:



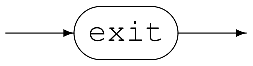
Example:

```
serfboard>display status
Feeder interface: 0.0.0.0:41000.
CGI interface:    127.0.0.1:9000.
Probe interface:
      router_metrics connected
Metrics:
   Group serfboard_demo01     16 metrics loaded.
   Group serfboard_demo02     10 metrics loaded.
   Group serfboard_demo03     14 metrics loaded.
Dashboards:
   serfboard_demo01     Active
   serfboard_demo02     Active
   serfboard_demo03     Active
```

4.4.4 Exit Command

This command terminates the user session to the command interface of `serfboard`.

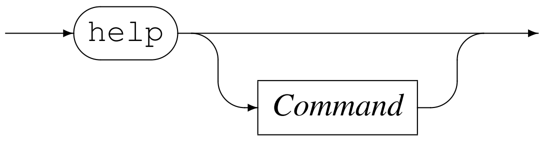
ExitCommand



4.4.5 Help Command

Help on `serfboard` commands.

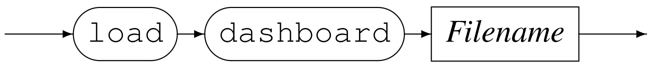
HelpCommand



4.4.6 Load Command

This command is used to load

- a dashboard configuration file (see section 5 on page 21 for how to code a dashboard configuration).

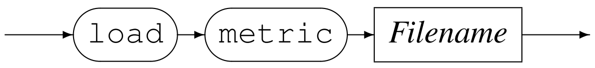


Example:

Load the dashboard defined in `serfboard_demo01.dboard`:

```
serfboard>load dashboard "${CONFIGS}/${DB}01.dboard"
Loaded dashboard file /usr/local/.../serfboard_demo01.dboard
```

- a metric configuration file (see section 6 on page 31 for how to code a metric configuration).



Example:

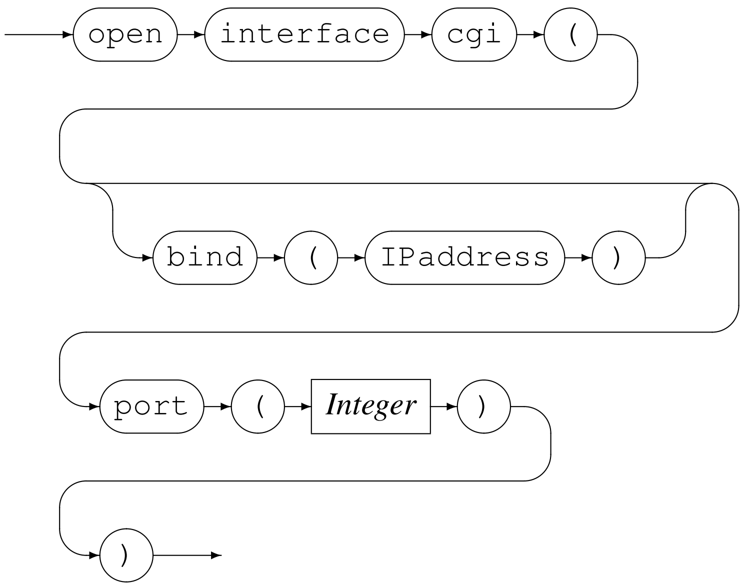
Load the metrics defined in `serfboard_demo01.metric`:

```
serfboard>load metric "${CONFIGS}/${DB}01.metric"
Loaded metric file /usr/local/.../serfboard_demo01.metric
```

4.4.7 Open Command

This command is used to open:

- a listener interface for communication with and initiated by CGI web clients which in turn serve HTML to multiple web browsers:



The optional parameter `bind` is to bind the socket connection to the `IPAddress` specified - `serfboard` will only listen for TCP/IP connections from this interface. The default binding is to `localhost`.

IPAddress can be specified as a host name or by using the Internet notation of dots and numbers.

Example:

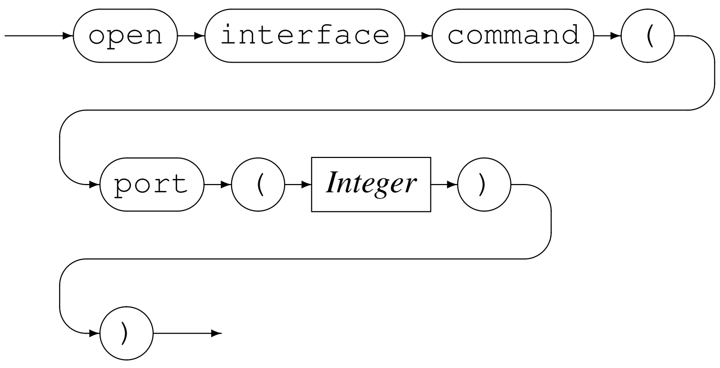
Open a `CGI` interface to accept connections on port 9001, only from `localhost`:

```
serfboard>open interface cgi (port(9001))
Opened CGI interface
```

Using `netstat` the server can be seen to be listening on the localhost and correct port:

```
jan@bloubos:˜> netstat -tanp|grep 9001
tcp        0      0 127.0.0.1:9001          0.0.0.0:*               LISTEN
                                                                23606/serfboard
```

- a `serfboard` command interface:



Example:

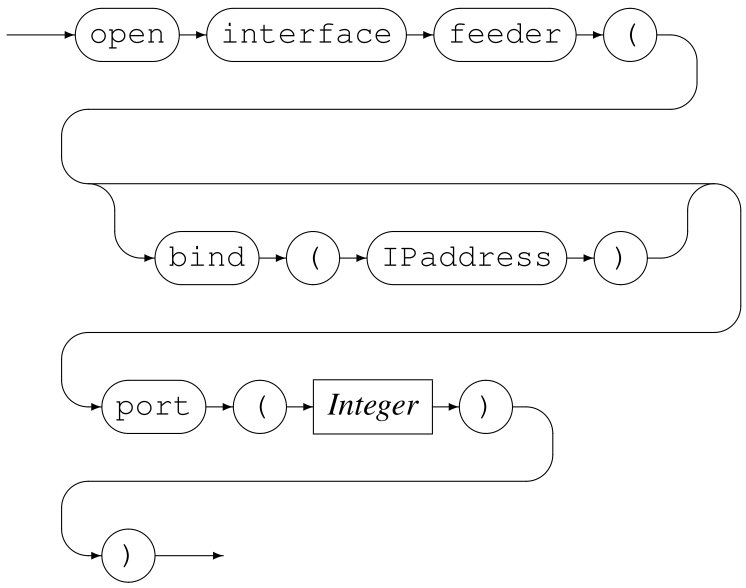
Open a `serfboard` command interface to accept connections on port 10001:

```
serfboard>open interface command(port(10001))
Opened command Port 10001
```

Using `netstat` the server can be seen to be listening on any interface and correct port:

```
jan@bloubos:˜> netstat -tanp|grep 10001
tcp        0      0 0.0.0.0:10001           0.0.0.0:*               LISTEN
                                                                30511/serfboard
```

- a listen interface for communication with metric feed probes:



The optional parameter `bind` is to bind the socket connection to the `IPAddress` specified - `serfboard` will only listen for connections from this interface. The default binding is to listen on all interfaces.

IPAddress can be specified as a host name or by using the Internet notation of dots and numbers.

`serfboard` listens on both TCP/IP and UDP for connections. UDP is a connectionless transport without guarantee of delivery.

Example:

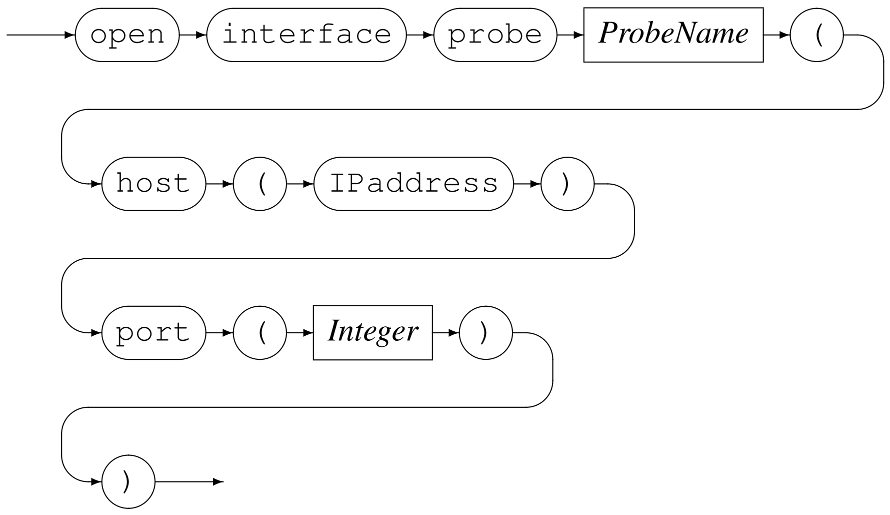
Open a metric feed interface to accept connections on port 41001, from all network interfaces:

```
serfboard>open interface feeder(port(41001))
Opened Feeder interface
```

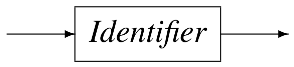
Using `netstat` it can be seen that the server is is listening on port 41001 for both TCP/IP and UDP.

```
jan@bloubos:~> netstat -anp|grep 41001
tcp        0      0 0.0.0.0:41001           0.0.0.0:*               LISTEN
                                                                29784/serfboard
udp        0      0 0.0.0.0:41001           0.0.0.0:*           29784/serfboard
```

- a command interface to a metric feeder probe:



ProbeName



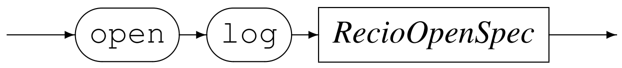
IPAddress can be specified as a host name or by using the Internet notation of dots and numbers.

Example:

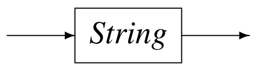
Open a command interface to a `SNMP` feeder probe on `codemagus.it.nednet.co.za` port 60050, giving it the name of `router_metrics` for identification:

```
serfboard>open interface probe router_metrics(host(codemagus.it.nednet.co.za)
           port(60050))
Opened probe interface
```

- a log file for recording metrics received:



RecioOpenSpec



RecioOpenSpec is a `recio` [1] open specification string.

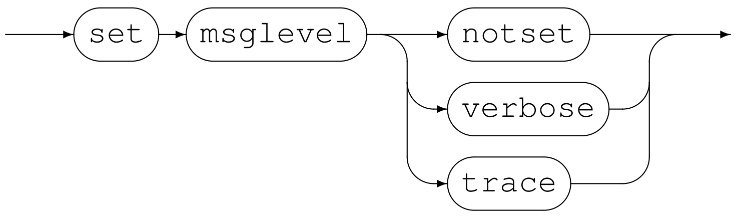
Example:

```
serfboard>open log "text(${HOME}/replay/atmams.log,mode=a)"
Log "text(/home/jan/replay/atmams.log,mode=a)" opened
```

4.4.8 Set Command

This command is used for setting the message level which determines the amount of diagnostic output `serfboard` writes. This option changes the settings active at `serfboard` startup as set by the parameters '`--verbose`' and '`--trace`' or the environment variable `CODEMAGUS_MSGLEVEL`.

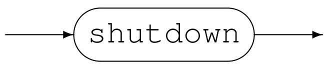
SetCommand



4.4.9 Shutdown Command

This command terminates `serfboard`.

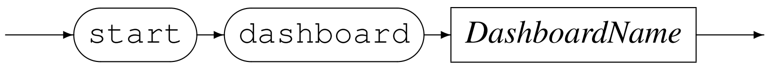
ShutdownCommand



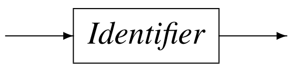
4.4.10 Start Command

This command is used to start a dashboard.

StartCommand



DashboardName



5 Dashboard Configuration

5.1 Overview

The dashboard configuration defines all the panels that make up a dashboard as the user would view it. For each panel the size, position, display instrument, title, description and the metrics to be used are specified. The metrics for a panel must be defined in a loaded metrics configuration file (see section 6 on page 31).

`serfboard` can load more than one dashboard configuration file via multiple `load` commands, as explained in section 4.4.6 on page 15.

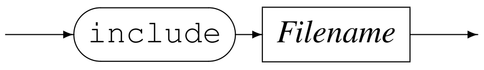
5.2 Reserved Words

Reserved words have a special meaning in terms of directing the parsing of the configuration file. The reserved words are:

`control`	`group`	`link`	`panel`
`dashboard`	`height`	`metric`	`text`
`delay`	`include`	`options`	`title`
`description`	`image`	`origin`	`url`
`width`			

5.3 Include Directive

The `include` directive, which must be on its own line, instructs the parser to treat the contents of a specified file as if those contents had appeared in the configuration at the point where the directive appears.



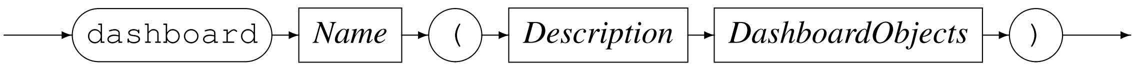
Example:

```
include ${CONFIGS}/serfboard_demo01.dboard
```

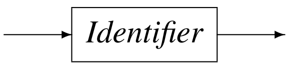
5.4 Dashboard Definition

The dashboard definition starts by defining the name of the dashboard. This identifies the dashboard once loaded by `serfboard`. Following the name is the description of the dashboard and then the panel definitions.

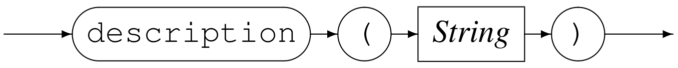
See appendix A.3 on page 37 for an example dashboard configuration file.



Name



Description



Note that the value of the description is a string and must be quoted as in the following example:

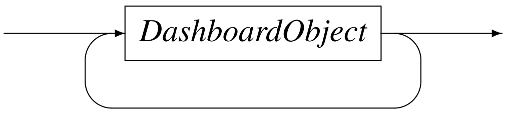
```
description("Serfboard Demo 02")
```

5.5 *DashboardObjects*

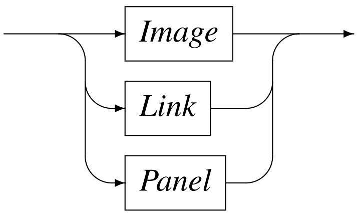
Each dashboard comprises objects as follows:

- optional hyper links.
- optional images.
- one or more panels in which a display instrument renders one or more metrics.

DashboardObjects



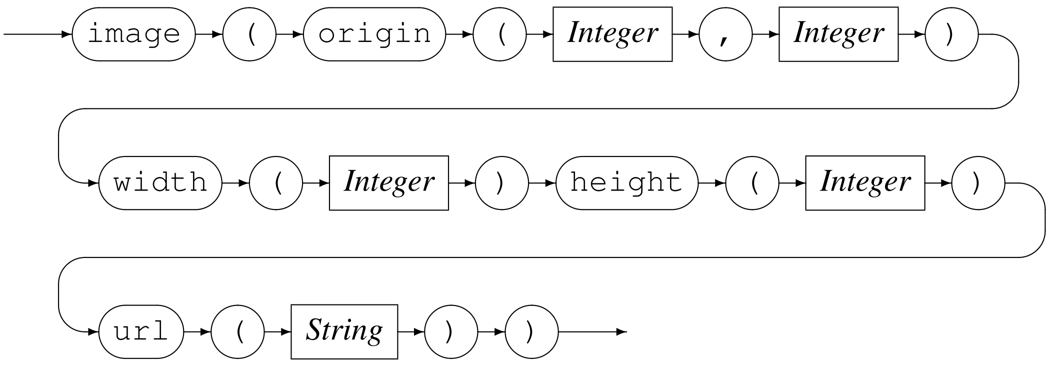
DashboardObject



5.6 *Image*

This is used to include images on the dashboard page.

Image



The attributes of the image statement are as follows:

- The `origin` in pixels.
- The `width` in pixels.
- The `height` in pixels.
- The actual URL.

Example:

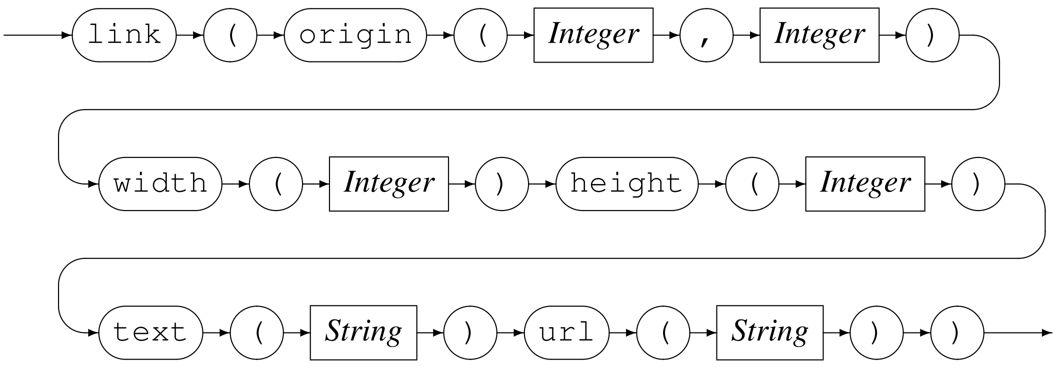
Include the Codemagus logo on the dashboard page:

```
image (origin(0,1) width(281) height(47)
    url("/serfboard/html/images/codemagus_logo.jpg"))
```

5.7 *Link*

This is used to add hyperlinked text to the page. The nature of a dashboard is that each panel refreshes at regular intervals as they request an update of data from `serfboard` causing the browser history to be lost. One function of a customisable link is the ability to return to the invoking web page.

Link



The attributes of the link statement are as follows:

- The `origin` in pixels.
- The `width` in pixels.
- The `height` in pixels.
- The `text` string that is displayed on the page
- The actual URL.

Example:

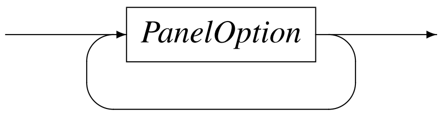
Define a link for returning to the main index web page:

```
link(origin(0,50) width(100) height(50) text("Home") url("/index.html"))
```

5.8 Panel

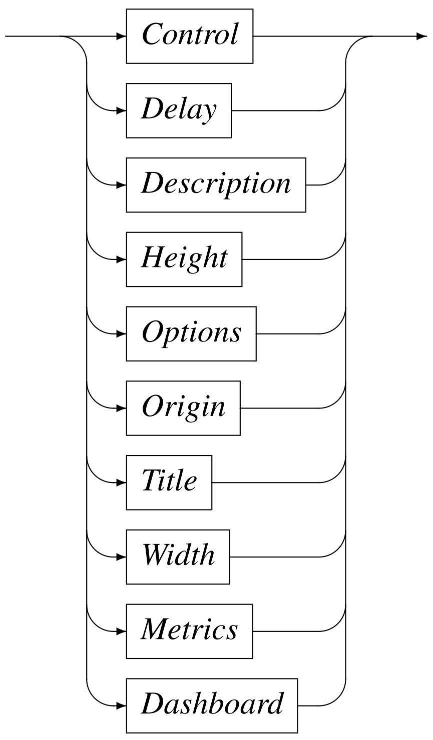
This is used to define one or more dashboard panels in which a display instrument renders one or more metrics.

Panel



Each panel consists of the following sub-options.

PanelOption

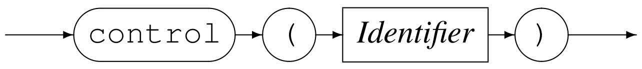


All the sub-options except *Options* and *Metrics* are required.

5.8.1 Control

This defines the display instrument used to render the panel. This value is case sensitive.

Control



Refer to Serfboard Instruments Guide and Reference Version 1 [2] for a complete list of valid display instrument names and more information on configuring them.

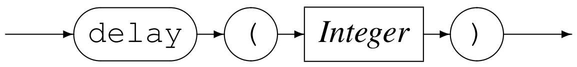
Examples are:

```
control(Graph)
control(Dials)
```

5.8.2 Delay

This defines the interval, in milliseconds, between updates for a panel.

Delay

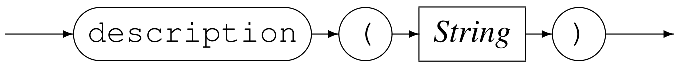


For example `delay(8000)` means that the panel is updated every 8 seconds.

5.8.3 Description

This defines the panel description.

Description



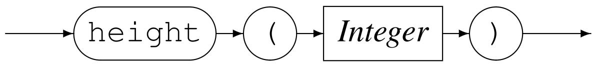
The description of the panel is used to generate a help web page that gives more detail of the data used by the instrument. If it is not specified then the title will be used as the help text. It must be enclosed in double quotes as in this example:

```
description("This shows invalid responses received for logon requests")
```

5.8.4 Height

This defines the height of panel in pixels.

Height

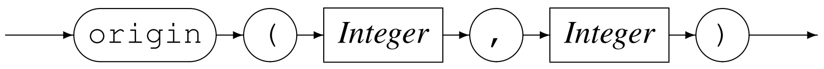


For example `height(300)` defines a panel 300 pixels high.

5.8.5 Origin

This defines the position of the panel in terms of x and y co-ordinates expressed in pixels with (0,0) being the top left corner.

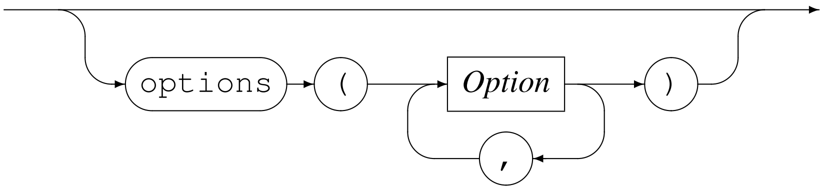
Origin



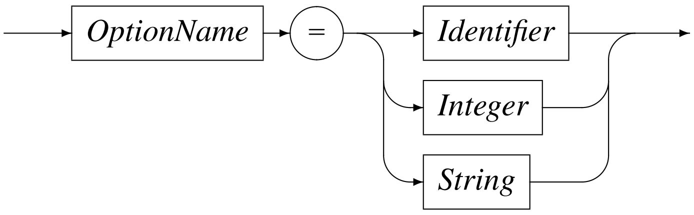
5.8.6 Options

This defines the options to be passed to the display instrument. These options refer to the panel including all metrics rendered inside it.

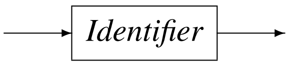
Options



Option



OptionName



Options are name-value pairs. An example of such a string is

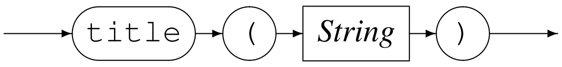
```
type=RATE,scaling=NONE,average_window=60,y1max=40,legend_columns=2
```

Refer to the documentation for each instrument in Serfboard Instruments Guide and Reference Version 1 [2] for more information.

5.8.7 Title

This defines the title for the panel.

Title



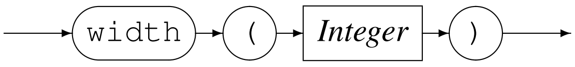
The title of the panel is used by some instruments as a heading. It must be enclosed in double quotes as in this example:

```
title("Response Times")
```

5.8.8 Width

This defines the width of the panel in pixels.

Width

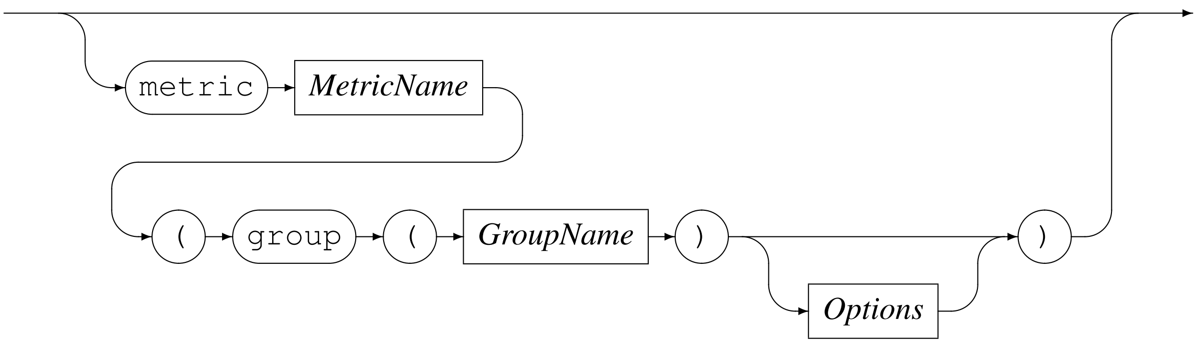


For example `width(150)` defines a panel 150 pixels wide.

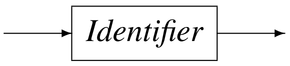
5.8.9 Metrics

This defines the metrics for the panel.

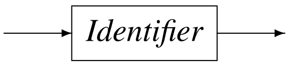
Metrics



MetricName



GroupName



For display instruments that accept metrics, the metrics must be of a specific type of metric data. The number of metrics a display instrument can process determines the maximum number of metrics it may accept for rendering. A dashboard designer must be aware of what incoming metrics are available in order to configure them correctly for the chosen display instrument.

A metric is identified by its name and the group it belongs to. The metric for inclusion in the panel must be defined, see section 6 on page 31.

Options are optional and follow the same syntax rules as described previously in section 5.8.6 on page 26 except that here they apply only to the metric they are defined for. For example in a panel displaying more than one line on a graph a specific colour, which will override the default, can be specified.

Examples of metric definitions are:

```
metric seq_100 (group(serfboard_demo02) options(max=20))
metric seq_200 (group(serfboard_demo02))
metric reversals (group(serfboard_demo02))
```

5.8.10 Dashboard

The dashboard option allows an embedded dashboard to be defined. This is explained in detail in section 5.9 on page 29

5.8.11 Example Panel Configuration

An example of a panel configuration that uses the display instrument `BarChart` to render the metrics is as follows:

```
panel Pinst (width(640) height(400) origin(530,580) control(BarChart)
    delay(10000)
    title("Instance Counts")
    description("Instances count in state machine states.")
    options(show_yaxis=YES,legend_columns=2)
    metric State_instances (group(serfboard_demo02))
    )
```

The metric `State_instances` is defined in the group `serfboard_demo02`.

5.9 Embedded Dashboards

A panel in a dashboard could be a summary to a more detailed embedded dashboard. This is achieved by defining a complete dashboard with in a panel. When the user clicks on the title of the panel the embedded dashboard is launched in a new window.

An example of this would be the ability to monitor multiple machines. The metrics from multiple machines are fed to *Serfboard*, but under normal load or operation there is no need for an operator to keep flipping from one dashboard to another. Rather an overall dashboard is defined made up of panels, typically using a digital display control, each of which displays the key indicators of a single machine. This allows the operator to view all the machines key metrics on one page. Each panel has a complete dashboard defined with in it that will display more detail (and history) of the state of the machine. If any of the key indicators exceeds normal values and are highlighted the operator clicks on the title of that panel and the complete dashboard for that machine is launched in a new window.

Embedding a complete dashboard in a panel could make reading and understanding the configuration difficult, so for readability it is advisable to define the embedded dashboard in a separate configuration file and use the `include` directive to include it in the correct place (as explained in appendix 5.3 on page 21).

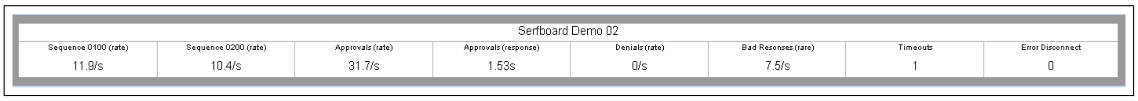
Example:

Panel `Demo02` is set up to monitor the key components of the Demo02 dashboard and when clicking on the title launches the Demo02 dashboard in a separate browser window. The Demo02 dashboard is defined in the configuration file `serfboard_demo02.dboard`, which is shown later in this document, see appendix A.3 on page 37.

The panel configuration is as follows

```
panel Demo02 (width(1280) height(80) origin(0,260) control(display)
   include ${CONFIGS}/serfboard_demo02.dboard
   delay(10000)
   title("Serfboard Demo 02")
   description("Serfboard Demo 02 - click for full dashboard")
   metric seq_100 (group(serfboard_demo02)
      options(title="Sequence 0100 (rate)",type=RATE))
   metric seq_200 (group(serfboard_demo02)
      options(title="Sequence 0200 (rate)",type=RATE))
   metric good_responses (group(serfboard_demo02)
      options(title="Approvals (rate)",type=RATE))
   metric good_responses (group(serfboard_demo02)
      options(title="Approvals (response)",type=RESPONSE))
   metric deny_responses (group(serfboard_demo02)
      options(title="Denials (rate)",type=RATE))
   metric bad_responses (group(serfboard_demo02)
      options(title="Bad Responses (rare)",type=RATE))
   metric Timeouts (group(serfboard_demo02)
      options(title="Timeouts",type=COUNT,alert=yes))
   metric Disconnected (group(serfboard_demo02)
      options(title="Error Disconnect",type=COUNT,alert=yes))
   )
```

An example of this definition on a dashboard is as follows:



Serfboard Demo 02

Sequence 0100 (rate)	Sequence 0200 (rate)	Approvals (rate)	Approvals (response)	Denials (rate)	Bad Resonses (rare)	Timeouts	Error Disconnect
11.9/s	10.4/s	31.7/s	1.53s	0/s	7.5/s	1	0

6 Metric Configuration

6.1 Overview

The metric configuration defines the incoming metrics to `serfboard`. If `serfboard` receives a metric from a metric feed probe that is not defined, the metric will be ignored.

`serfboard` can load more than one metric configuration file as per the `load` command explained in section 4.4.6 on page 15.

6.2 Reserved Words

Reserved words have a special meaning in terms of directing the parsing of the configuration file. The reserved words are:

```
average_window  description  metric_group  title
metric          include
```

6.3 Include Directive

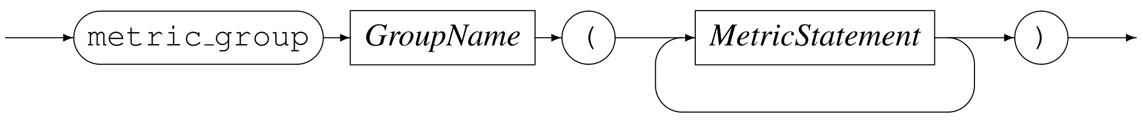
The include directive may be used in a metric configuration and follows the same rules as the include directive for the dashboard configuration as shown in section 5.3 on page 21.

6.4 *Metrics definition*

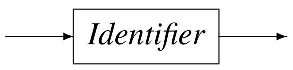
The configuration file starts by defining the name of the metric group followed by the metrics in this group. *Serfboard* can load more than one metric configuration file. A metric must have a unique name within a group and all metrics can be uniquely identified by its name qualified by the group name.

See appendix A.4 on page 39 for an example metric configuration file.

Body



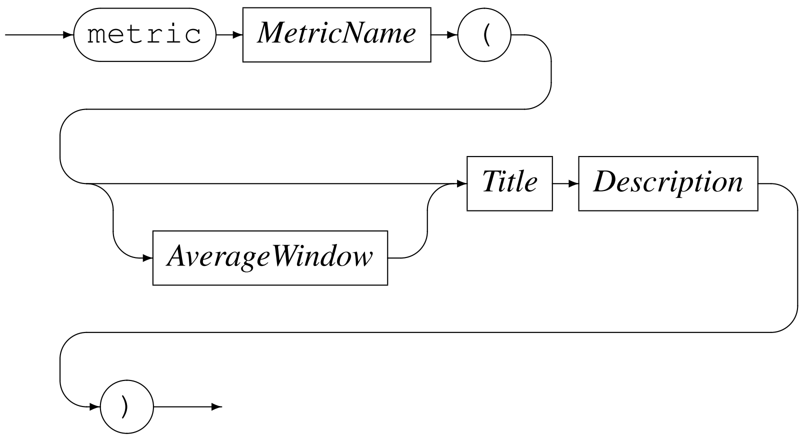
GroupName



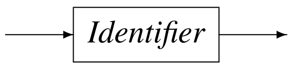
6.4.1 *MetricStatement*

This statement defines a metric within the metric group.

MetricStatement

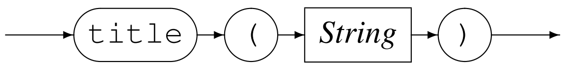


MetricName



The metric name identifies the metrics that `serfboard` will accept from a metric feeder. The metric feeder must supply both the group name and the metric name.

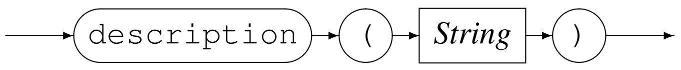
Title



This string is used as a label by some instruments. An example is:

```
title("Ptester Console")
```

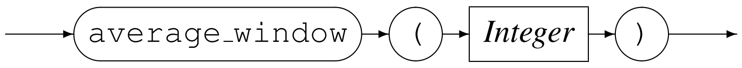
Description



The description is used in the help facility of the web page. An example is:

```
description("Ptester console for entering commands")
```

AverageWindow



This parameter defines the window duration (in seconds) over which the average value of the metric is smoothed. The default, if not specified is 60 seconds.

6.4.2 Example Metric Definition

An example of a metric configuration that corresponds to an incoming metric called `Sessions` is as follows:

```
metric Sessions (
   average_window(120)
   title("Sessions Offered")
   description("Sessions Offered"))
```

7 Accessing Dashboards

Once the `serfboard` server is running and accepting metric data from the feed agents it is ready to serve the metrics to the web application. The *Serfboard* web application is made up of CGI programs that display the dashboards. To invoke a dashboard from a web browser the user needs to supply the group name and CGI port number to connect to. This information can be generated in either of the following ways:

- Dynamically at run time. The web user can fill in a form that allows them to choose the group name. In this instance the port may also be entered by the user or derived within the form. This approach makes it easy to link to new dashboards as the user just needs to be given the new information.
- Statically from with an HTML page. The link can be embedded in an invoking HTML page. This requires changes to the web page before users can use a new dashboard.

In both cases the link is similar to:

```
serfboard/cgi-bin/serfdashcgi?group=serfboard_demo01&amp;port=9000
```

An example can be seen in the demonstration index page which is located in

```
${CODEMAGUS_HOME}/serfboard/html/serfdemo.html
```

and shown in appendix B on page 43.

A *Serfboard* Complete example

A.1 Introduction

The first step is to list the metrics that will be received and the instruments to be used to render them. A strip of about 70 to 110 pixels in height should be left open for the page header at the top. The size and position for each instrument is then added in.

This dashboard is supplied with *Serfboard* as `serfboard_demo02` and information on running it and the replay script is in the installation manual Serfboard Installation Guide and Reference Version 1 [3]. The replay function simulates a real system sending metrics to the `serfboard` instance.

A.2 Dashboard Specification

Name	Size and position	Instrument	Metrics URL
Logo	281x47*(0,1)	image	images/codemagus_logo.jpg
Home	100x50@(0,150)	URL	/
Dashboard Demo Index	200x50@(150,150)	URL	serfdemo.html
Pevent	100x50@(1290,110)	eventLog	
Pstatus	100x50@(1400,110)	SystemStatus	
Pconsole	70x50@(1510,110)	ConsoleLink	
Psess	380x400@(0,170)	Graph	Sessions
Psess2	150x400@(370,170)	Dials	Sessions
Pmess	380x400@(530,170)	Graph	seq_100 seq_200 reversals
Pmess2	150x400@(900,170)	Dials	seq_100 seq_200 reversals
Pgood	380x400@(1060,170)	Graph	good_responses
Pgood2	150x400@(1430,170)	RateRespDials	good_responses
Perrors	520x80@(0,580)	warning	Timeouts Disconnected
Pdeny	520x460@(0,670)	Graph	deny_responses bad_responses
Pinst	640x400@(530,580)	BarChart	State_instances
Pdials	400x400@(1180,580)	Dials	seq_100 seq_200 reversals Sessions
Phist	1050x140@(530,990)	Graph	Timeouts Disconnected

Table 1: Panel Configuration

Note that in this example there are no metrics specified for the console panel, as this does not take any input data from the server.

The configuration files needed to build this dashboard are shown below.

A.3 Dashboard Configuration

For the example above, the dashboard configuration file is as follows:

```
dashboard serfboard_demo02 (
description("Serfboard Demo 02")
image (origin(0,1) width(281) height(47)
    url("/serfboard/html/images/codemagus_logo.jpg"))
link (origin(0,150) width(100) height(50) text("Home")
    url("/"))
link (origin(150,150) width(200) height(50) text("Dashboard Demo Index")
    url("/serfboard/html/serfdemo.html"))
panel Pevent (width(100) height(50) origin(1290,110) control(eventLog)
   delay(15000)
   title("System Events")
   description("System events log")
   options(max_events=20,expire=0,truncate=YES)
   )
panel Pstatus (width(100) height(50) origin(1400,110) control(systemStatus)
   delay(15000)
   title("System Status")
   description("System Status warning light")
   options(timeout=50)
   )
panel Pconsole (width(70) height(50) origin(1510,110) control(ConsoleLink)
   delay(0)
   title("Ptester console")
   description("Ptester console for entering commands")
   )
panel Psess (width(380) height(400) origin(0,170) control(Graph)
   delay(10000)
   title("Sessions Offered")
   description("This graph plots the rate of sessions offered (per second) against time.")
   options(duration=300,type=RATE,scaling=NONE,average_window=NO,y1max=40)
   metric Sessions (group(serfboard_demo02))
   )
panel Psess2 (width(150) height(400) origin(370,170) control(Dials)
   delay(10000)
   title("Sessions")
   description("Sessions")
   options(duration=300,scaling=AUTO,average_window=YES)
   metric Sessions (group(serfboard_demo02) options(type=RATE,max=40))
   )
panel Pmess (width(380) height(400) origin(530,170) control(Graph)
   delay(10000)
   title("Message Sequences")
   description("This graph plots the rate of the message sequences.")
   options(duration=300,type=RATE,scaling=NONE,average_window=YES,
         y1max=40,legend_columns=2)
   metric seq_100 (group(serfboard_demo02))
   metric seq_200 (group(serfboard_demo02))
   metric reversals (group(serfboard_demo02))
   )
panel Pmess2 (width(150) height(400) origin(900,170) control(Dials)
   delay(10000)
   title("Messages")
   description("Messages")
   options(duration=300,scaling=NONE,average_window=YES)
   metric seq_100 (group(serfboard_demo02) options(type=RATE,max=20))
   metric seq_200 (group(serfboard_demo02) options(type=RATE))
   metric reversals (group(serfboard_demo02) options(type=RATE))
   )
panel Pgood (width(380) height(400) origin(1060,170) control(Graph)
   delay(10000)
   title("Good Responses")
   description("Good responses from type 0100, 0200, 0220 and 0420 messages")
   options(duration=300,type=RATERESP,scaling=NONE,average_window=YES,
         y1max=40,y2max=1)
   metric good_responses (group(serfboard_demo02))
   )
panel Pgood2 (width(150) height(400) origin(1430,170) control(RateRespDials)
   delay(10000)
   title("Good responses")
   description("Good responses")
   options(duration=300,average_window=YES)
   metric good_responses (group(serfboard_demo02))
   )
panel Perrors (width(520) height(80) origin(0,580) control(display)
   delay(10000)
   title("Error Alerts")
   description("Error Alerts")
   options(duration=300)
   metric Timeouts (group(serfboard_demo02)
      options(display_value="current_count",alert=YES))
   metric Disconnected (group(serfboard_demo02)
      options(display_value="current_count",alert=YES))
   )
panel Pdeny (width(520) height(460) origin(0,670) control(Graph)
   delay(10000)
   title("Requests Denied and Bad Responses")
   description("Requests denied and Bad Responses")
   options(duration=300,type=RESP,scaling=AUTO,average_window=YES,y1max=40)
   metric deny_responses (group(serfboard_demo02))
   metric bad_responses (group(serfboard_demo02))
   )
panel Pinst (width(640) height(400) origin(530,580) control(BarChart)
   delay(10000)
   title("Instance Counts")
   description("Instances count in state machine states.")
   options(show_yaxis=YES,legend_columns=2)
   metric State_instances (group(serfboard_demo02))
   )
panel Pdials (width(400) height(400) origin(1180,580) control(Dials)
   delay(10000)
   title("Messages and Sessions")
   description("Messages and Sessions")
   options(duration=300,scaling=NONE,average_window=YES)
   metric seq_100 (group(serfboard_demo02) options(type=RATE,max=40))
   metric seq_200 (group(serfboard_demo02) options(type=RATE,max=40))
   metric reversals (group(serfboard_demo02) options(type=RATE,max=40))
   metric Sessions (group(serfboard_demo02) options(type=RATE,max=40))
   )
panel Phist (width(1050) height(140) origin(530,990) control(Graph)
   delay(15000)
   title("Error History")
   description("Error History")
   options(type=TICK,legend_columns=2,xticks=9)
   metric Timeouts (group(serfboard_demo02))
   metric Disconnected (group(serfboard_demo02))
   )
)
```

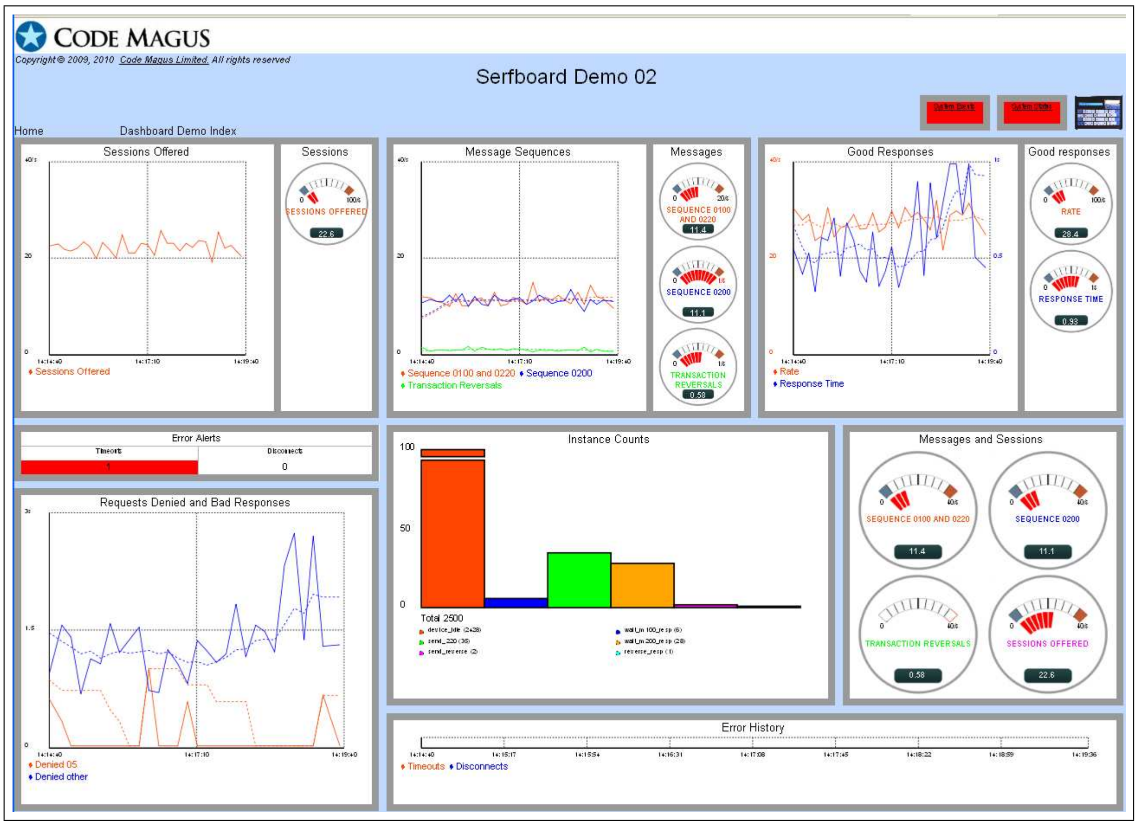
A.4 Metrics Configuration

For the example above, the metric configuration file is as follows:

```
-- File: serfboard_demo.metric
-- Commands for Serfboard Dashboard serfboard_demo01
--
-- Copyright (c) 2010 Code Magus Limited. All rights reserved.
--
-- $Author: janvlok $
-- $Date: 2011/03/18 10:09:40 $
-- $Id: serfboard_demo.metric,v 1.2 2011/03/18 10:09:40 janvlok Exp $
-- $Revision: 1.2 $
-- $State: Exp $
-- $Log: serfboard_demo.metric,v $
-- Revision 1.2  2011/03/18 10:09:40  janvlok
-- Option name changes
--
-- Revision 1.1  2011/01/26 14:37:10  janvlok
-- Embedded dashboards
--
metric_group serfboard_demo01 (
metric Sessions (
   title("Sessions Offered")
   description("Sessions offered"))
metric CC_auth_OKAY (
   title("Credit Card Approvals")
   description("Credit card completions"))
metric CC_auth_DENY (
   title("Credit Card Denials")
   description("Credit card completions"))
metric CC_completion (
   title("Credit Card completion")
   description("Credit card completions"))
metric logon_OKAY (
   title("Logon Responses")
   description("POS device logons successful"))
metric Timeouts (
   title("Timeouts")
   description("Timeout waiting for a response to an request sent."))
metric Error_disconnect (
   title("Disconnects")
   description("Unexpected disconnects"))
metric hsm_errors (
   title("HSM Errors")
   description("HSM Errors - serious problem"))
metric DB_auth_DENY_INV_PIN (
   title("Pin Errors")
   description("Debit card authorisations denied - Invalid PIN"))
metric logon_BAD (
   title("Bad Logon")
   description("Short logon response received. Very serious error, that needs to be fixed, before continui
metric logon_DENY (
   title("Logon Deny")
   description("Logon denied. Very serious error, that needs to be fixed, before continuing."))
metric Connection_Retry_1 (
   title("Connections First Retry")
   description("Connection retry - first attempt"))
metric Connection_Retry_2 (
   title("Connections Second Retry ")
   description("Connection retry - second attempt"))
metric Connection_Retry_3 (
   title("Connections Third Retry ")
   description("Connection retry - third attempt"))
metric Connection_Refused (
   title("Connections Refused")
   description("Unable to establish an Network connection"))
metric State_instances (
   title("Instances")
   description("Snap shot of all the states in the state machine with instances waiting to transition out
)
metric_group serfboard_demo02 (
metric Sessions (
   average_window(120)
   title("Sessions Offered")
   description("Sessions Offered"))
metric seq_100 (
   average_window(120)
   title("Sequence 0100 and 0220")
   description("Message types 0100 and 0220"))
metric seq_200 (
   average_window(120)
   title("Sequence 0200")
   description("Message type 0200"))
metric reversals (
   average_window(120)
   title("Transaction Reversals")
   description("Transaction reversals"))
metric good_responses (
   title("Requests approved")
   description("Requests approved"))
metric Timeouts (
   title("Timeouts")
   description("Timeout waiting for a response to a request sent"))
metric Disconnected (
   title("Disconnects")
   description("Serious problem. The circuit was disconnected and the Integrator must do a logon after the
metric State_instances (
   title("Instances of states")
   description("Snap shot of all the states in the state machine with instances waiting to transition out
metric deny_responses (
   title("Denied 05")
   description("Request denied - response 05"))
metric bad_responses (
   title("Denied other")
   description("Request denied - other response codes"))
)
metric_group serfboard_demo03 (
metric cpu_tot_idle (
   average_window(180)
   title("Idle")
   description("cpu idle"))
metric cpu_tot_user (
   average_window(180)
   title("User")
   description("cpu user"))
metric cpu_tot_kernel (
   average_window(180)
   title("Kernel")
   description("cpu kernel"))
metric cpu_tot (
   average_window(180)
   title("Total")
   description("Total cpu used: user + kernel"))
metric disk_tot_nread (
   average_window(180)
   title("Read")
   description("Bytes Read"))
metric disk_tot_nwritten (
   average_window(180)
   title("Written")
   description("Bytes Written"))
metric disk_tot_reads (
   average_window(180)
   title("Reads")
   description("Reads"))
metric disk_tot_writes (
   average_window(180)
   title("Writes")
   description("Writes"))
metric net_tot_ipackets (
   average_window(180)
   title("ipackets")
   description("ipackets"))
metric net_tot_opackets (
   average_window(180)
   title("opackets")
   description("opackets"))
metric net_tot_rbytes (
   average_window(180)
   title("rbytes")
   description("rbytes"))
metric net_tot_obytes (
   average_window(180)
   title("obytes")
   description("obytes"))
metric memory_freemem (
   title("freemem")
   description("Memory: Free."))
metric memory_availrmem (
   title("availrmem")
   description("Memory: availeble."))
)
```

A.5 Dashboard View

Once metrics have been received by *Serfboard* the dashboard should look similar to this:



B Invoking Web Page Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!--
File: serfboard_index.html
This html page displays the list of available Demo Dashboards.
$Author: hayward $
$Date: 2012/05/16 13:10:42 $
$Id: serfdemo.html,v 1.9 2012/05/16 13:10:42 hayward Exp $
$Name:  $
$Revision: 1.9 $
$State: Exp $
$Log: serfdemo.html,v $
Revision 1.9  2012/05/16 13:10:42  hayward
Add favicon.ico to the web pages.

Revision 1.8  2011/01/26 14:37:35  janvlok
Embedded dashboards

Revision 1.7  2010/08/23 12:44:46  janvlok
Combined the demo’s into one serfboard

Revision 1.6  2010/05/10 12:07:57  hayward
Remove old serfboard index html pages
update the serfdemo page and tidy it.

Revision 1.5  2010/05/05 22:51:39  hayward
Update dashboard descriptions to better fit their purpose.

Revision 1.4  2010/05/05 22:42:19  hayward
Improve layout of demo index.

-->
<html>
  <head>
    <meta name="generator" content=
    "HTML Tidy for Linux/x86 (vers 31 October 2006), see www.w3.org">
    <link rel="shortcut icon" href="/serfboard/html/images/favicon.ico"/>
    <title>
      Code Magus Demo Dashboards.
    </title>
    <meta http-equiv="content-type" content="text/html; charset=us-ascii">
    <style type="text/css">
BODY {  background-color: #BFD9FF; );
            font-family: "times new roman", Symbol, serif; font-size:large}
      A:link { color: DarkBlue }
      A:visited { color: maroon }
      A:active { color: fuchsia }
      span.TT {font-family: monospace; font-weight: lighter; font-size: large;}
      ul.grp { border-color: black; border-width: medium;border-style:solid;background-color: #87CEFA ; }
      li.oddrow { background-color: LightBlue; width: 100%;
         border-bottom-color: LightBlue; border-bottom-width: thick;
         border-bottom-style: solid; }
      li.evenrow { background-color: LightCyan; width: 100%;
         border-bottom-color: LightCyan; border-bottom-width: thick;
         border-bottom-style: solid; }
      p.row {margin: 0px 0px 0px 20px; background-color: transparent; }
      table.products {text-align: left; border: 0; width: 60%; }
      form.grp { border-color: black; border-width: medium;border-style:solid;
         background-color: PowderBlue; }
      table.tabinp { width: 60%; background-color: PowderBlue; }
      td.xcolhead   {text-align: right; }
      input.inpbox {width: 40%;}
    </style>
  </head>
```

```
<body>
  <div>
    <h2>
      Code Magus Demo Dashboards
    </h2>
    <blockquote>
      <table class="products" frame="none" cellpadding="2" cellspacing="2"
      summary="Code Magus Dashboards">
        <colgroup span="1"></colgroup>
        <colgroup span="1"></colgroup>
        <tr>
          <td style="vertical-align: top">
            <h3>
              Pre-defined Dashboards
            </h3>
            <ul class="grp">
              <li class="oddrow">
                <a href=
                "/serfboard/cgi-bin/serfdashcgi?dname=serfboard_demo01&amp;port=9000">
                Serfboard Demonstration 01.</a>
                <p class="row">
                  This dashboard is an example of Credit and Debit card
                  transactions from an ATM device showing approvals, denials
                  and rate and response times.
                </p>
              </li>
              <li class="evenrow">
                <a href=
                "/serfboard/cgi-bin/serfdashcgi?dname=serfboard_demo02&amp;port=9000">
                Serfboard Demonstration 02.</a>
                <p class="row">
                  This dashboard is an example showing different ways that
                  the number of sessions and rate and response time in a
                  system under test may be displayed.
                </p>
              </li>
              <li class="oddrow">
                <a href=
                "/serfboard/cgi-bin/serfdashcgi?dname=serfboard_demo03&amp;port=9000">
                Serfboard Demonstration 03.</a>
                <p class="row">
                  This dashboard is an example of monitoring the key
                  components of a physical machine. These components include
                  CPU time, memory used and network activity.
                </p>
              </li>
              <li class="evenrow">
                <a href=
                "/serfboard/cgi-bin/serfdashcgi?dname=serfboard_demo&amp;port=9000">
                Serfboard Demonstration.</a>
                <p class="row">
                  This dashboard is an example of monitoring the key
                  components of the three demonstration dashboards.
                </p>
              </li>
            </ul><br>
            <h3>
              Launch an ad hoc Dashboard
            </h3>
            <p>
              An ad hoc dashboard may be launched if the dashboard name and
              connection port number of a running instance of serfboard are
              known.
            </p><br>
            <form class="grp" id="launch" action=
            "/serfboard/cgi-bin/serfdashcgi" name="launch">
```

```
                <input type="hidden" name="host" id="host" value="localhost">
                <table class="tabinp" summary="Adhoc Dashboard">
                  <tr>
                    <td class="xcolhead">
                      <label for="dname">Dashboard Name:</label>
                    </td>
                    <td>
                      <input class="xinpbox" type="text" name="dname" id=
                      "dname" size="15">
                    </td>
                  </tr>
                  <tr>
                    <td class="xcolhead">
                      <label for="port">Port:</label>
                    </td>
                    <td>
                      <input class="inpbox" type="text" name="port" id="port">
                    </td>
                    <td align="right">
                      <input type="submit" name="submit" id="submit" value=
                      "Submit"> <input type="reset" name="reset" id="reset"
                      value="Reset">
                    </td>
                  </tr>
                </table>
              </form>
            </td>
          </tr>
        </table>
      </blockquote>
    </div>
  </body>
</html>
```

References

[1] recio: Record Stream I/O Library Version 1. CML Document CML00001-01, Code Magus Limited, July 2008. PDF.

[2] Serfboard Instruments Guide and Reference Version 1. CML Document CML00024-01, Code Magus Limited, July 2008. PDF.

[3] Serfboard Installation Guide and Reference Version 1. CML Document CML00025-01, Code Magus Limited, July 2008. PDF.

[4] Serfboard User Guide Version 1. CML Document CML00027-01, Code Magus Limited, July 2008. PDF.

[5] cmlxsnmp: SNMP Metric Probe. CML Document CML00044-01, Code Magus Limited, June 2009. PDF.

[6] cmlxaixp: AIX Performance Metric Probe. CML Document CML00045-01, Code Magus Limited, June 2009. PDF.

[7] cmlxwinp: Windows Performance Metric Probe. CML Document CML00048-01, Code Magus Limited, June 2009. PDF.

[8] cmlxwasp: Websphere Application Server Performance Metric Probe. CML Document CML00049-01, Code Magus Limited, June 2009. PDF.

[9] cmlxsolp: Solaris Performance Metric Probe. CML Document CML00065-01, Code Magus Limited, June 2009. PDF.

[10] cmdname: Command Name Resolver Library Version 1. CML Document CML00076-01, Code Magus Limited, December 2010. PDF.