Celoxica Manager

User Guide

About This Document

This document describes the Celoxica Manager user guide.

Copyright Information

Celoxica and the Celoxica logo are trademarks of Celoxica Limited.

All other products or services mentioned herein may be trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous development and improvement. All particulars of the product and its use contained in this document are given by Celoxica Limited in good faith. However, all warranties implied or express, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. Celoxica Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any incorrect use of the product. The information contained herein is subject to change without notice and is for general guidance only.

Sales	sales@celoxica.com
Customer Support	support@celoxica.com
Website	http://www.celoxica.com

UK Head Office
Celoxica Limited
34 Porchester Road
London
W2 6ES, UK
Phone: +44 (0) 20 7313 3180

US Head Office
Celoxica Inc.
275 Madison Avenue, Suite 404
New York, NY
10016, USA
Phone: +1 (0) 212 880 2075

US Chicago Office
Celoxica Inc.
141 W Jackson Blvd, Suite 2350
Chicago, IL
60604, USA
Phone: +1 (0) 312 893 1204

Content

Revisions

Revision	Date	Description of Changes
R2012-7.0	05 OCT 2012	Release R2012-7.0 - Miscellaneous enhancements
R2012-ub-beta	03 SEP 2012	Initial version

1. Functional Overview

1.1 Pre-requisites

The following software packages need to be installed in order to use the Celoxica Manager:

- Python 2.6 or greater
- Package python-zmq
- Package celoxica-base (where the Celoxica Manager is packaged)

1.2 Celoxica Manager

The Celoxica Manager, namely `clxmgr.py`, is a Python script which allows the user to monitor and parameterize the following processes:

- **Master Daemon**

  The Master Daemon or the Manager Daemon, namely `clxmgrd`, is responsible for monitoring all the other Celoxica processes and switching to the backup node in case of failure on the master node. See section 1.3 for more details.

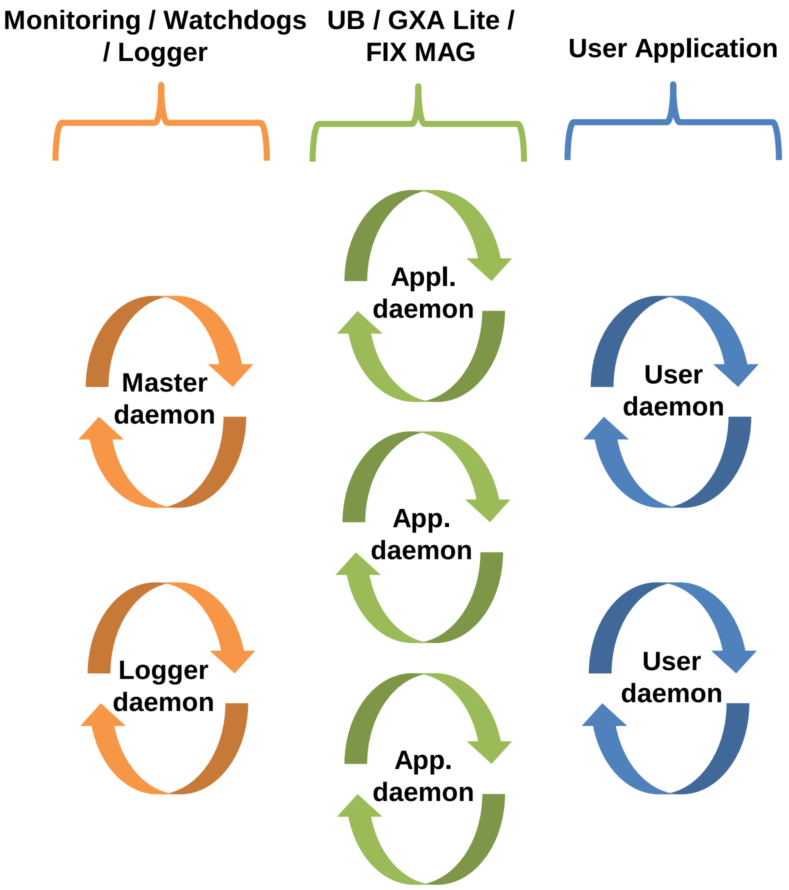
- **Logger Daemon**

  This process, namely `clxlogd`, is responsible for recording application logs asynchronously. See section 1.4 for more details.

- **Application processes**

  The following Celoxica applications are monitored by the Manager:

  1. UltraBook (UB)
  2. FIX Market Access Gateway (MAG)
  3. Generic eXecution Accelerator (GXA Lite)



The Manager is configured using an XML-style '.cfg' file. When the Manager initializes, it loads this configuration file; see section 2 for more details.

The Manager provides commands that allow the user to monitor the processes; see section 4 for more details.

1.3 Master Daemon

The Master Daemon can start and stop processes, acts as a watchdog and sends alerts in case of issues (e.g. process down or restarted).

It is responsible for ensuring and maintaining system availability, and is also able to manage failover to a backup node – note that this feature is not available yet.

The list of processes to watch and actions are fully configurable, for instance:

- Automatic process restart

  When a process is stopped, the Master Daemon tries to restart the process. The number of restart attempts and the timespan during which the Master Daemon tries to restart the process are configurable.

- Stop watching a process after n restart attempts

  If the Master Daemon fails to restart a process, the process will cease to be watched.

The Master Daemon configuration settings are part of the *Manager Configuration* file.

1.4 Logger Daemon

The processes which use the Logger Daemon write their logging information in shared memory. The Logger Daemon reads the shared memory, collects and formats the logs, and writes the formatted log messages to files. Log messages are coloured according to the log level. The log size is set at startup to avoid disk space issues during runtime. The number of log files is configurable and the logs can be written to specific files according to filtering expressions.

The Logger Daemon therefore provides a simple and homogeneous logging approach across applications.

If the Logger Daemon is down for any reason, the processes will write their logs into default logging files.

The Logger Daemon is automatically started when the Master Daemon starts.

The Logger Daemon configuration settings are part of the *Manager Configuration* file.

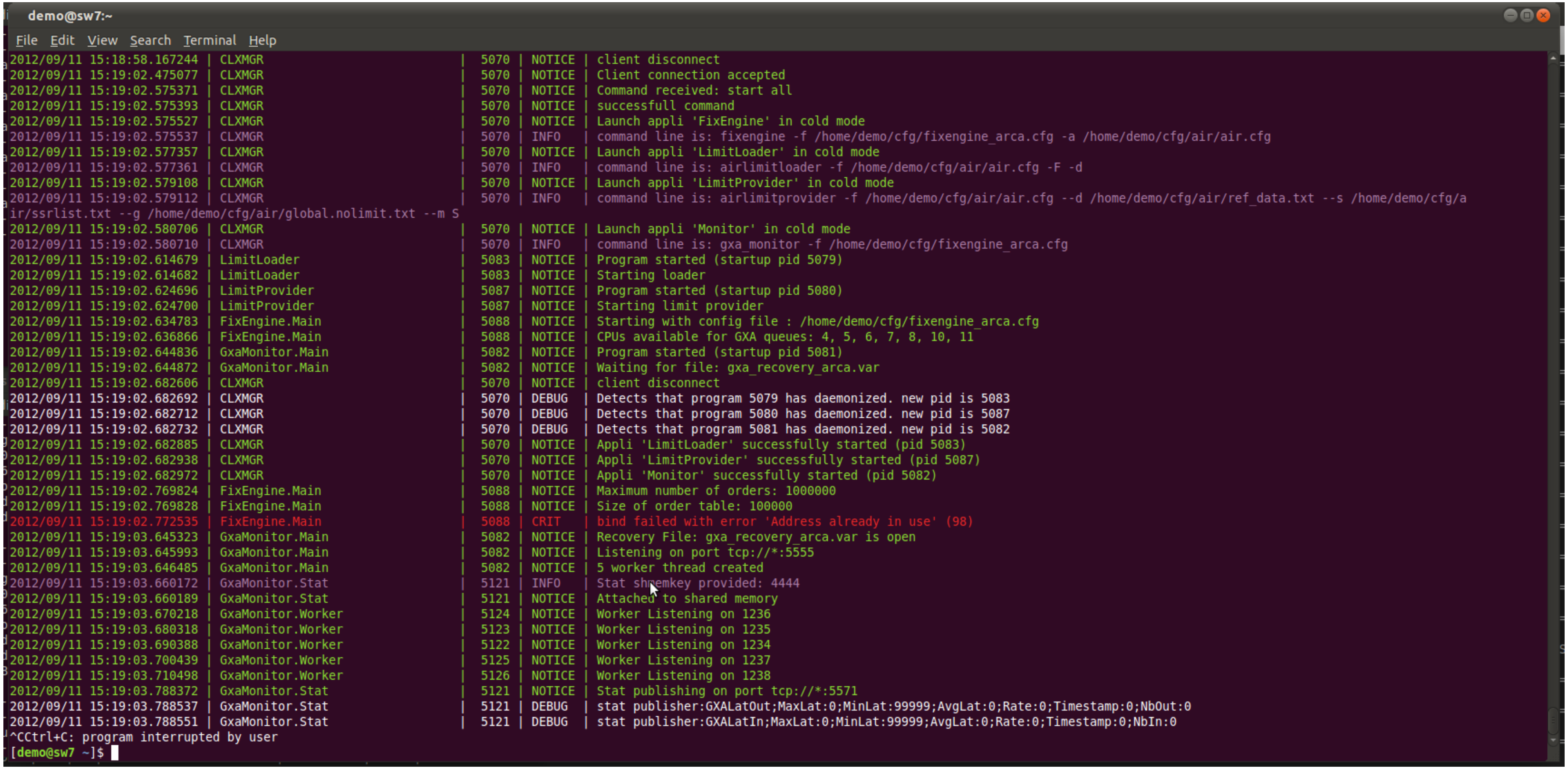
The generated log files can be viewed via a Python script that is provided in the celoxica-base package, and which is used as follows:

```
viewlog.py [options] <file>
```

To view the help screen for this script:

```
viewlog.py -h
```

An example of a log file display using the viewlog command is shown below.



```
2012/09/11 15:18:58.167244 | CLXMGR                           |  5070 | NOTICE | client disconnect
2012/09/11 15:19:02.475077 | CLXMGR                           |  5070 | NOTICE | Client connection accepted
2012/09/11 15:19:02.575371 | CLXMGR                           |  5070 | NOTICE | Command received: start all
2012/09/11 15:19:02.575393 | CLXMGR                           |  5070 | NOTICE | successfull command
2012/09/11 15:19:02.575527 | CLXMGR                           |  5070 | NOTICE | Launch appli 'FixEngine' in cold mode
2012/09/11 15:19:02.575537 | CLXMGR                           |  5070 | INFO   | command line is: fixengine -f /home/demo/cfg/fixengine_arca.cfg -a /home/demo/cfg/air/air.cfg
2012/09/11 15:19:02.577357 | CLXMGR                           |  5070 | NOTICE | Launch appli 'LimitLoader' in cold mode
2012/09/11 15:19:02.577361 | CLXMGR                           |  5070 | INFO   | command line is: airlimitloader -f /home/demo/cfg/air/air.cfg -F -d
2012/09/11 15:19:02.579108 | CLXMGR                           |  5070 | NOTICE | Launch appli 'LimitProvider' in cold mode
2012/09/11 15:19:02.579112 | CLXMGR                           |  5070 | INFO   | command line is: airlimitprovider -f /home/demo/cfg/air/air.cfg --d /home/demo/cfg/air/ref_data.txt --s /home/demo/cfg/a
ir/ssrlist.txt --g /home/demo/cfg/air/global.nolimit.txt --m S
2012/09/11 15:19:02.580706 | CLXMGR                           |  5070 | NOTICE | Launch appli 'Monitor' in cold mode
2012/09/11 15:19:02.580710 | CLXMGR                           |  5070 | INFO   | command line is: gxa_monitor -f /home/demo/cfg/fixengine_arca.cfg
2012/09/11 15:19:02.614679 | LimitLoader                      |  5083 | NOTICE | Program started (startup pid 5079)
2012/09/11 15:19:02.614682 | LimitLoader                      |  5083 | NOTICE | Starting loader
2012/09/11 15:19:02.624696 | LimitProvider                    |  5087 | NOTICE | Program started (startup pid 5080)
2012/09/11 15:19:02.624700 | LimitProvider                    |  5087 | NOTICE | Starting limit provider
2012/09/11 15:19:02.634783 | FixEngine.Main                   |  5088 | NOTICE | Starting with config file : /home/demo/cfg/fixengine_arca.cfg
2012/09/11 15:19:02.636866 | FixEngine.Main                   |  5088 | NOTICE | CPUs available for GXA queues: 4, 5, 6, 7, 8, 10, 11
2012/09/11 15:19:02.644836 | GxaMonitor.Main                  |  5082 | NOTICE | Program started (startup pid 5081)
2012/09/11 15:19:02.644872 | GxaMonitor.Main                  |  5082 | NOTICE | Waiting for file: gxa_recovery_arca.var
2012/09/11 15:19:02.682606 | CLXMGR                           |  5070 | NOTICE | client disconnect
2012/09/11 15:19:02.682692 | CLXMGR                           |  5070 | DEBUG  | Detects that program 5079 has daemonized. new pid is 5083
2012/09/11 15:19:02.682712 | CLXMGR                           |  5070 | DEBUG  | Detects that program 5080 has daemonized. new pid is 5087
2012/09/11 15:19:02.682732 | CLXMGR                           |  5070 | DEBUG  | Detects that program 5081 has daemonized. new pid is 5082
2012/09/11 15:19:02.682885 | CLXMGR                           |  5070 | NOTICE | Appli 'LimitLoader' successfully started (pid 5083)
2012/09/11 15:19:02.682938 | CLXMGR                           |  5070 | NOTICE | Appli 'LimitProvider' successfully started (pid 5087)
2012/09/11 15:19:02.682972 | CLXMGR                           |  5070 | NOTICE | Appli 'Monitor' successfully started (pid 5082)
2012/09/11 15:19:02.769824 | FixEngine.Main                   |  5088 | NOTICE | Maximum number of orders: 1000000
2012/09/11 15:19:02.769828 | FixEngine.Main                   |  5088 | NOTICE | Size of order table: 100000
2012/09/11 15:19:02.772535 | FixEngine.Main                   |  5088 | CRIT   | bind failed with error 'Address already in use' (98)
2012/09/11 15:19:03.645323 | GxaMonitor.Main                  |  5082 | NOTICE | Recovery File: gxa_recovery_arca.var is open
2012/09/11 15:19:03.645993 | GxaMonitor.Main                  |  5082 | NOTICE | Listening on port tcp://*:5555
2012/09/11 15:19:03.646485 | GxaMonitor.Main                  |  5082 | NOTICE | 5 worker thread created
2012/09/11 15:19:03.660172 | GxaMonitor.Stat                  |  5121 | INFO   | Stat shmemkey provided: 4444
2012/09/11 15:19:03.660189 | GxaMonitor.Stat                  |  5121 | NOTICE | Attached to shared memory
2012/09/11 15:19:03.670218 | GxaMonitor.Worker                |  5124 | NOTICE | Worker Listening on 1236
2012/09/11 15:19:03.680318 | GxaMonitor.Worker                |  5123 | NOTICE | Worker Listening on 1235
2012/09/11 15:19:03.690388 | GxaMonitor.Worker                |  5122 | NOTICE | Worker Listening on 1234
2012/09/11 15:19:03.700439 | GxaMonitor.Worker                |  5125 | NOTICE | Worker Listening on 1237
2012/09/11 15:19:03.710498 | GxaMonitor.Worker                |  5126 | NOTICE | Worker Listening on 1238
2012/09/11 15:19:03.788372 | GxaMonitor.Stat                  |  5121 | NOTICE | Stat publishing on port tcp://*:5571
2012/09/11 15:19:03.788537 | GxaMonitor.Stat                  |  5121 | DEBUG  | stat publisher:GXALatOut;MaxLat:0;MinLat:99999;AvgLat:0;Rate:0;Timestamp:0;NbOut:0
2012/09/11 15:19:03.788551 | GxaMonitor.Stat                  |  5121 | DEBUG  | stat publisher:GXALatIn;MaxLat:0;MinLat:99999;AvgLat:0;Rate:0;Timestamp:0;NbIn:0
^CCtrl+C: program interrupted by user
[demo@sw7 ~]$
```

2. Manager Configuration

2.1 Configuration File Schema

The basic structure of the Celoxica Manager configuration file is shown in the following schema. This schema shows tags but not values.

```
<config>
      <clxmgr>
            <max-stops-in-delay>
            <stops-delay>
            <kill-clxlogd-at-exit>
            <zmq-cnx-pubstring>
            <zmq-cnx-reqstring>
            <log-level>
      </clxmgr>

      <clxlogd>
            <logkey>
            <logpath>
            <routes>
                  <default-route>
                  <route>
            </routes>
      </clxlogd>

      <init-appli>
            <executable>
            <parameters>
      </init-appli>

      <appli>
            <executable>
            <cold-parameters>
            <hot-parameters>
      </appli>
</config>
```

2.2 Configuration Settings

2.2.1 General Configuration Settings

This section describes the general configuration settings.

2.2.1.1 config

Description This tag initiates a body for the Celoxica Manager configurations to be made in a file. This tag is at the top of the file.

Usage `<config>`

2.2.2 Master Daemon Settings

This section describes the Master Daemon configuration settings.

2.2.2.1 config.clxmgr

Description This tag initiates a body for the Master Daemon configurations to be made.

Usage `<clxmgr>`

2.2.2.2 config.clxmgr.max-stops-in-delay

Description Sets the upper limit on the number of times that a monitored application is restarted automatically by the Master Daemon, during the timespan defined by `<stops-delay>` (see below).

Usage `<max-stops-in-delay> max <max-stops-in-delay>`

Parameters `max` Integer

Example `<max-stops-in-delay> 10 </max-stops-in-delay>`

2.2.2.3 config.clxmgr.stops-delay

Description Sets the timespan during which a monitored application is restarted automatically by the Master Daemon, up to a maximum of `<max-stops-in-delay>` times.

Note that this parameter takes precedence over `<max-stops-in-delay>` i.e. if a time of `<stops-delay>` elapses before the number of restarts reaches `<max-stops-in-delay>`, no more restart attempts will be made.

Usage `<stops-delay> timespan <stops-delay>`

Parameters `timespan` Integer.

Expressed in seconds.

Example `<stops-delay> 60 </stops-delay>`

2.2.2.4 config.clxmgr.kill-clxlogd-at-exit

Description When enabled, the automatic Logger Daemon is stopped when the Master Daemon is stopped.

Usage `<kill-clxlogd-at-exit> kill </kill-clxlogd-at-exit>`

Parameters `kill` Boolean

Example `<kill-clxlogd-at-exit> 0 </kill-clxlogd-at-exit>`

2.2.2.5 config.clxmgr.zmq-cnx-pubstring

Description Sets the port used by the Master Daemon to publish a process state change to the Manager GUI.

Usage `<zmq-cnx-pubstring> port </zmq-cnx-pubstring>`

Parameters `port` Publication port

Example `<zmq-cnx-pubstring> tcp://*:50001 </zmq-cnx-pubstring>`

2.2.2.6 config.clxmgr.zmq-cnx-reqstring

Description Sets the port used by the Manager GUI to request the processes states from the Master Daemon.

Usage `<zmq-cnx-reqstring> port </zmq-cnx-reqstring>`

Parameters `port` Request port

Example `<zmq-cnx-reqstring> tcp://*:50002 </zmq-cnx-reqstring>`

2.2.2.7 config.clxmgr.log-level

Description Sets the Master Daemon logging level.

Usage `<log-level> level </log-level>`

Parameters `level` Integer.

Possible values are:

- 5 Log notice – normal, but significant, condition
- 6 Log info – informational message
- 7 Debug – debug-level message

Example `<log-level> 5 </log-level>`

2.2.3 Logger Daemon Settings

This section describes the Logger Daemon configuration settings.

The logger configuration settings for applications using the Logger daemon are described in section *Client Application Configuration*.

2.2.3.1 config.clxlogd

Description This tag initiates a body for the Logger Daemon configurations to be made.

Usage `<clxlogd cpu>`

Attributes `cpu` Optional CPU.

Example `<clxlogd cpu=”0”>`

2.2.3.2 config.clxlogd.logkey

Description Sets the Logger Daemon shared memory key.

Usage `<logkey> key </logkey>`

Parameters `key` Integer

Example `<logkey> 11 </logkey>`

2.2.3.3 config.clxlogd.logpath

Description Sets the path to which the Logger Daemon will write the log files.

Usage `<logpath> repository </logpath>`

Parameters `repository` Repository path

Example `<logpath> /home/log </logpath>`

2.2.3.4 config.clxlogd.routes

Description This tag initiates a body for the Logger Daemon routes configurations to be made.

Usage `<routes>`

2.2.3.5 config.clxlogd.routes.default-route

Description Sets the parameters for the default log files that the Logger Daemon will write to.

Usage `<default-route destination nb-files max-file-size>`

Attributes

`destination` File base name.

If set to e.g. “defaultlog”, files are created with names defaultlog.1, defaultlog.2, etc. according to the other settings (see below).

`nb-files` Number of log files to keep.

When the number specified is reached, the oldest file is deleted and a new one is opened for writing (also see below).

Default is 10, minimum is 3 and maximum is 100.

`max-file-size` Maximum log file size expressed in bytes.

The logger always writes to <destination>.1. When this file reaches the size specified, the suffix for all existing log files is incremented by 1, and a new <destination>.1 is opened for writing.

Default is 10485760 bytes.

Minimum is 10000 bytes.

Maximum is 1073741824 bytes.

Example `<default-route destination=”log” nb-files=”10” max-file-size=”10000000”>`

In this example, the logger thread will always write to `log.1`. When `log.1` reaches 10000000 bytes in size, `log.1` is renamed to `log.2`, the old `log.2` becomes `log.3` etc. and if `log.10` exists, it will be deleted.

2.2.3.6 config.clxlogd.routes.route

Description Sets the parameters for the specific/filtered log that the Logger Daemon will write to.

This node is optional: if not defined, the settings in `<default-route>` will apply. Several `<route>` nodes can be defined.

Usage `<route regexp destination nb-files max-file-size>`

Attributes

`regexp` Filtering expression with the format “name.level”:

- “name” is used to identify the process
- “level” is used to identify the logging level (*, NOTICE, INFO or DEBUG)

If a log matches the filtering expression, it will be written to the log files as per the parameters in this node, otherwise `<default-route>` applies.

`destination` File base name.

If set to e.g. “filteredlog”, files are created with names filteredlog.1, filteredlog.2, etc. according to the other settings (see below).

`nb-files` Number of log files to keep.

When the number specified is reached, the oldest file is deleted and a new one is opened for writing (also see below).

Default is 10, minimum is 3 and maximum is 100.

`max-file-size` Maximum log file size expressed in bytes.

The logger always writes to <destination>.1. When this file reaches the size specified, the suffix for all existing log files is incremented by 1, and a new <destination>.1 is opened for writing.

Default is 10485760 bytes.

Minimum is 10000 bytes.

Maximum is 1073741824 bytes.

Example `<route regexp=”FIX*.NOTICE” destination = “FIX” nb-files=”10” max-file-size=”10000000”>`

In this example, a `NOTICE` log from a process whose name starts with `“FIX”` is appended to the current `FIX.n` log file.

`<route regexp=”UB*.*” destination = “UB” nb-files=”10” max-file-size=”10000000”>`

In this example, all logs from a process whose name starts with `“UB”`” are appended to the current `UB.n` log file.

2.2.4 Initialization Application Settings

This section describes the initialization application configuration settings.

2.2.4.1 config.init-appli

Description This tag initiates a body for the initialization configurations to be made.

Usage `<init-appli name comment>`

Attributes	`name`	Initialization application name
	`comment`	User-defined comment

Example `<market name=”Init_App” comment=”my_comment”>`

2.2.4.2 config.init-appli.executable

Description Sets the path to the initialization application executable to be launched.

Usage `<executable> init-exec-path </executable>`

Parameters `init-exec-path` Path to the executable

Example `<executable> /home/apps/Init_App </executable>`

2.2.4.3 config.init-appli.parameters

Description Sets the configuration parameters to initialize the applications.

Usage `<parameters> app-params </parameters>`

Parameters `app-params` Application parameters e.g switches, paths to config files, etc.

Example `<parameters> -f /home/cfg/fixengine.cfg </parameters>`

2.2.5 Applications Settings

This section describes the applications configuration settings.

2.2.5.1 config.appli

Description

This tag initiates a body for the application configurations to be made.

This tag must be defined as many times as there are applications monitored by the Master Daemon.

Usage

```
<appli name startup-index comment>
```

Attributes

`name`	Application name
`startup-index`	Application start id. Must be unique across all defined applications. Applications are launched in the `startup-index` order.
`comment`	User-defined comment

Example

```
<appli name=”FixEngine” startup-index=”1”
comment=”my_comment”>
```

2.2.5.2 config.appli.executable

Description

Sets the path to the application executable to be launched.

Usage

```
<executable> exec-path </executable>
```

Parameters

`exec-path`	Path to the executable

Example

```
<executable> /home/apps/FixEngine </executable>
```

2.2.5.3 config.appli.cold-parameters

Description

Sets the configuration parameters to execute the application in a cold start scenario. For instance, a memory clean-up might be performed for the first start of a day.

Usage

```
<cold-parameters> cold-params </cold-parameters>
```

Parameters

`cold-params`	Cold-start application parameters e.g. switches, paths to config files, etc.

Example

```
<cold-parameters> -f
/home/cfg/FixEngine_arca_cold.cfg </cold-
parameters>
```

2.2.5.4 config.appli.hot-parameters

Description

Sets the configuration parameters to execute the application in a hot start scenario. For instance, no memory clean-up might be needed for intra-day restarts.

Usage

```
<hot-parameters> hot-params </hot-parameters>
```

Parameters

`hot-params`	Hot-start application parameters e.g. switches, paths to config files, etc.

Example

```
<hot-parameters> -f
/home/cfg/FixEngine_arca_hot.cfg </hot-parameters>
```

3. Client Application Configuration

3.1 Configuration File Schema

The basic structure of the logger configuration file of a client application using the Logger Daemon is shown in the following schema. Tags are shown in the schema, values are not.

```
…
        <logger>
                <key>
                <file>
                <size>
        </logger>
…
```

3.2 Configuration Settings

3.2.1 Logger

The logger configuration allows the setting of parameters for an application to use the Logger Daemon. These configuration parameters are part of the application configuration file.

3.2.1.1 logger

Description This tag initiates a body for the logger configurations to be made.

Usage `<logger>`

3.2.1.2 logger.key

Description Sets the Logger Daemon shared memory key.

Usage `<key> key </key>`

Parameters `key` integer

Example `<key> 11 </key>`

3.2.1.3 logger.file

Description Sets the default log file if the Logger Daemon is not present.

Usage `<file> file </file>`

Parameters `file` File path

Example `<file> /tmp/log.log </file>`

3.2.1.4 logger.size

Description Sets the size of the shared memory if not created.

Usage `<size> size </size>`

Parameters `size` Log size expressed in bytes

Example `<size> 5242880 </size>`

4. Manager Commands

4.1 Usage

The command-line usage for the Celoxica Manager is as follows:

```
clxmgr.py [-f <config>] <command>
```

4.2 Commands

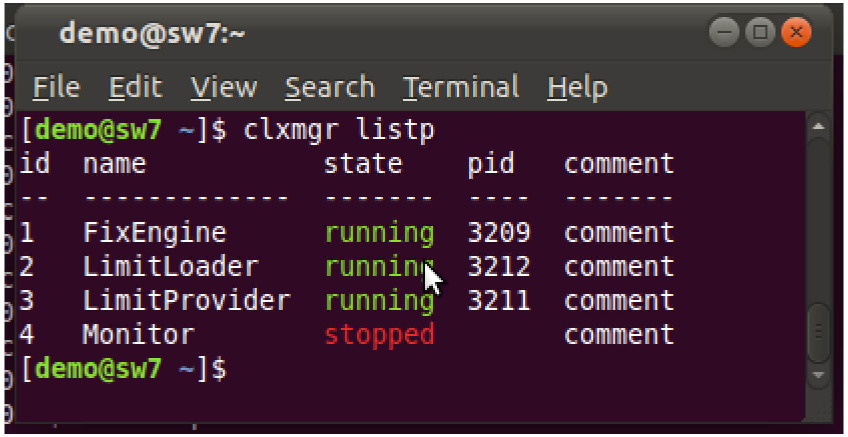
The commands available are:

1. listp

This command returns the process list monitored by the master daemon e.g.:

```
$HOME/bin/clxmgr.py -f $HOME/cfg/clxmgr.cfg listp
```

Output example:



```
[demo@sw7 ~]$ clxmgr listp
id  name           state    pid   comment
--  -------------  -------  ----  -------
1   FixEngine      running  3209  comment
2   LimitLoader    running  3212  comment
3   LimitProvider  running  3211  comment
4   Monitor        stopped        comment
[demo@sw7 ~]$
```

The Id, the name and the comment provided in the command output respectively correspond to the attributes `startup-index`, `name` and `comment` as defined by the configuration node `config.appli`.

2. status

This command returns the Logger Daemon status e.g.:

```
$HOME/bin/clxmgr.py -f $HOME/cfg/clxmgr.cfg status
```

Output example:

```
clxlogd running; pid 16449
```

3. start

This command starts an application process. Note that the initialization application (configuration node *config.init-appli*) must be started before other applications.

To launch the initialization application:

```
$HOME/bin/clxmgr.py -f $HOME/cfg/clxmgr.cfg start
```

To launch all applications apart the initialization application:

```
$HOME/bin/clxmgr.py -f $HOME/cfg/clxmgr.cfg start all
```

To launch one single application defined by its name `<process_name>`:

```
$HOME/bin/clxmgr.py -f $HOME/cfg/clxmgr.cfg start
<process_name>
```

4. stop

This command stops an application process.

To stop all running applications excluding the initialization application:

```
$HOME/bin/clxmgr.py -f $HOME/cfg/clxmgr.cfg stop all
```

To stop one single application defined by its name `<process_name>`:

```
$HOME/bin/clxmgr.py -f $HOME/cfg/clxmgr.cfg stop
<process_name>
```

5. attach

This command attaches a process defined by its `<pid>` and its `<process_name>` to the Master Daemon so that it can be watched e.g.:

```
$HOME/bin/clxmgr.py -f $HOME/cfg/clxmgr.cfg attach
<process_name> <pid>
```

6. exit

This command kills only the Master Daemon:

```
$HOME/bin/clxmgr.py -f $HOME/cfg/clxmgr.cfg exit
```

7. help

This command lists all available commands:

```
$HOME/bin/clxmgr.py -f $HOME/cfg/clxmgr.cfg help
```