# refclock: Reference Clock API and Server Reference Version 1

## CML00058-01

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



August 16, 2016

# Contents

# 1 Introduction

It is important to have metrics collected and observations made on the multiple platforms of distributed and client-server applications be relative to a uniform clock, or synchronised clocks. It is not always convenient to use protocols such as Network Time Protocol (NTP) to keep the clocks of the various machines of such systems in sync. Clocks on components of a system under test (SUT) which are more than a few minutes out from one another cause problems in observing and statistically determining correlations in the metrics of the SUT, which often have to be mitigated against by running the tests longer and pruning to take out the noise, or to re-align the data back to some uniform clock for post-processing. However, non-functional testing often requires on-line analysis and to support this the test systems' observations; any collected metrics from the distributed components of the SUT; and those of the machines and network elements hosting the SUT need to be made in available in real time reporting and rendering.

The approach then is to time-stamp collected metrics from components of the SUT and the test system using a nominated clock on a component of the test system as a reference clock. Keeping this clock synchronised with a specific time zone should not be a problem, should that be required, and a protocol such as NTP should suffice. The frequency with which time-stamps are required makes it inhibitive to continuously refer to the reference clock. The Code Magus Limited Reference Clock API comprises a set of functions which determine the drift between the reference clock and supplies subsequent time-stamps from the local clock with an adjustment applied.

Figure 1 on page 3 shows the relationship between the reference clock server, each component of a system under test and where they use the adjusted time when producing metrics, reports and logs that report on the performance of the system under test.

The Code Magus Limited Reference Clock package provides following services:

- Code Magus Limited Reference Clock Server. The server responds to requests for its time when asked. See section 2 on page 4 for more information.
- Code Magus Limited Reference Clock API. The API supplies alternative functions used to obtain time values to those supplied by the standard C library. It also allows the caller to discover the difference between the local clock time and that of the reference clock server. See section 3 on page 5 for more information.

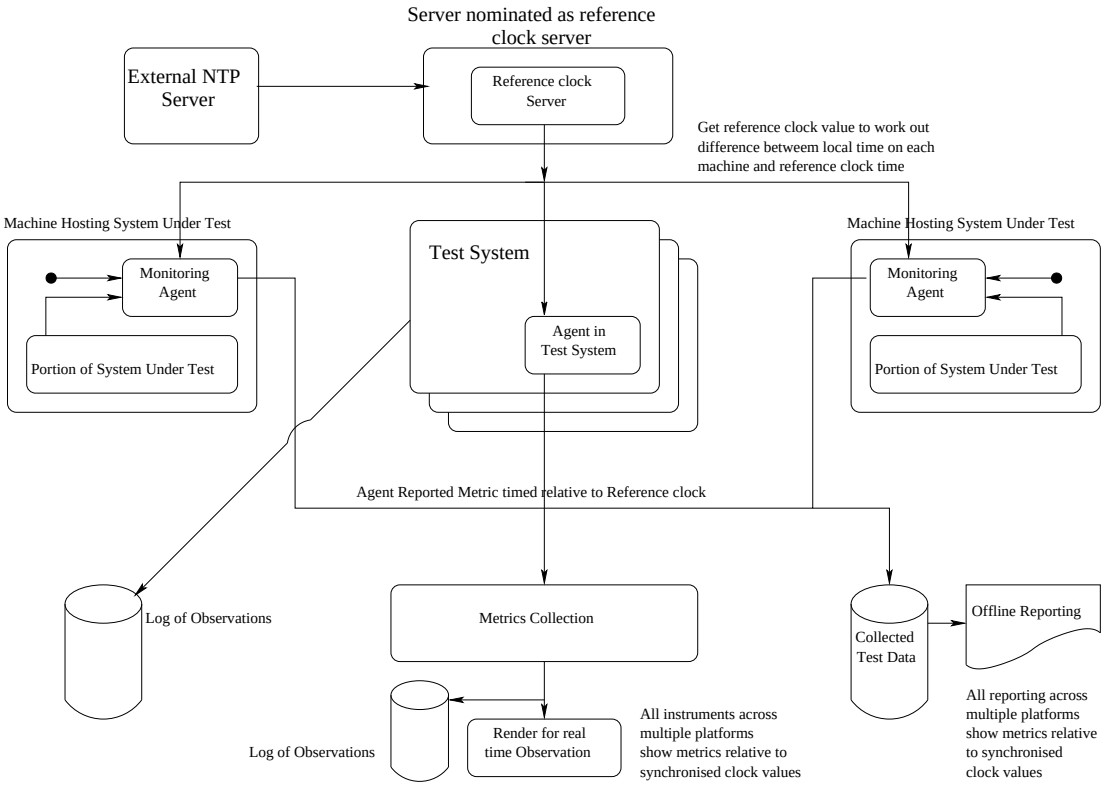



Figure 1: Overview of a System under Test using a Code Magus Limited Reference Clock implementation

# 2 Code Magus Limited Reference Clock Server

## 2.1 Introduction

This section describes `cmltimed`, which is the Code Magus Limited Reference Clock server. `cmltimed` listens for inbound connections from clients and responds to their clock request messages. The response message contains the current time of the machine on which the server is running.

## 2.2 Starting and Stopping the Server

The `cmltimed` server is started from the command line. Using `--help` will show the options that may be passed to the server at start-up.

```
Code Magus Limited Reference Clock Server V1.0: build 2011-10-31-10.13.45
[./cmltimed] $Id: cmltimed.c,v 1.7 2010/10/25 10:51:40 hayward Exp $
Copyright (c) 2010 by Code Magus Limited. All rights reserved.
 [Contact: stephen@codemagus.com].
Usage: cmltimed [OPTION...]
  -p, --port=<port>                          Listen port
  -m, --master-server=<IPaddress>:<port>     Synchronise with CML master clock
                                             reference server
  -v, --verbose                              Verbose processing
  -t, --trace                                Message trace

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

Where the options are:

- '`-p|--port`' Specifies the TCP/IP port that `cmltimed` will listen on for inbound client connections. This is a required option.
- '`-v|--verbose`' When specified, `cmltimed` will output diagnostic information about its processing.
- '`-t|--trace`' When specified, `cmltimed` prints a trace of all communication messages.

The server can be shut down by sending it a `SIGTERM` signal.

# 3 Code Magus Limited Reference Clock API

## 3.1 Introduction

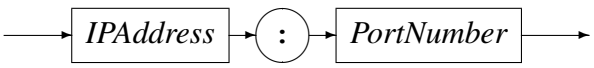
This section describes the Code Magus Limited Reference Clock API. The API provides the following function calls:

- `refclock_error()` - see section 3.3 on page 6
- `refclock_delta_time()` - see section 3.4 on page 6
- `refclock_gettimeofday()` - see section 3.5 on page 6
- `refclock_time()` - see section 3.6 on page 6

The function prototypes are made available to the application program by including the header file `refclock.h`, see appendix B on page 7. A sample program that uses the API is shown in appendix C on page 10.

## 3.2 Contacting the Server

The API obtains the TCP/IP address of the Code Magus Limited Reference Clock server from the environment variable `CODEMAGUS_REFCLOCK_SERVER`. If this environment variable is not set, the API will continue with unsynchronised clocks, using the local clock. The value of this variable must have the following format:




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

*PortNumber* is an integer and specifies the port number the server is listening on.

### 3.2.1 Examples

An example of how to set this on Unix to either a network address (192.168.20.20) or the host name (www.codemagus.com) using port 60060 is as follows:

```
export CODEMAGUS_REFCLOCK_SERVER=192.168.20.20:60060
```

...or

```
export CODEMAGUS_REFCLOCK_SERVER=www.codemagus.com:60060
```

On Windows this can be set from the control panel, system, environment variables or from the DOS command prompt as follows:

```
set CODEMAGUS_REFCLOCK_SERVER=192.168.20.20:60060
```

...or

```
set CODEMAGUS_REFCLOCK_SERVER=www.codemagus.com:60060
```

## 3.3 The `refclock_error()` Function

```
char *refclock_error(void);
```

Function `refclock_error()` returns a pointer to a message relating to the last error returned by one of the other API functions.

## 3.4 The `refclock_delta_time()` Function

```
int refclock_delta_time(long long int *delta_time);
```

Function `refclock_delta_time()` sets the clock difference in microseconds between the local machine and the Code Magus Limited Reference Clock server in the supplied `delta_time` argument. On success zero is returned, otherwise on error -1 is returned and a formatted error message can be obtained by calling `refclock_error()`.

## 3.5 The `refclock_gettimeofday()` Function

```
int refclock_gettimeofday(struct timeval *tv,void *tz);
```

Function `refclock_gettimeofday()` sets `tv` to the time since the Epoch (00:00:00 UTC, January 1, 1970) returned from the result from calling `gettimeofday` on the local machine and adjusting it with the difference as calculated from the Code Magus Limited Reference Clock server. If successful zero is returned, otherwise on error -1 is returned and a formatted error message can be obtained by calling `refclock_error()`.

## 3.6 The `refclock_time()` Function

```
time_t refclock_time(time_t *t);
```

Function `refclock_time()` returns the time in seconds since the Epoch (00:00:00 UTC, January 1, 1970) from the seconds portion of the result from calling `gettimeofday()` on the local machine and adjusting it with the difference as calculated from the Code Magus Limited Reference Clock server. On success zero is returned, otherwise on error -1 is returned and a formatted error message can be obtained by calling `refclock_error()`. If t is not NULL then the time is also stored at the location pointed to by t.

# A How the Time Difference is Calculated

In order to perform clock synchronisation, the time difference between the reference clock server and the local machine must be calculated. This difference is then used to adjust the local time when a time value is required.

On the first call to the API a connection is made to the Code Magus Limited Reference Clock server. Multiple requests are made to the server and the response time of each request is measured. The client chooses the request with the lowest response time and takes the difference between the local time for the request and the time returned by the clock server and adjusts this difference by half the response time for that request. This value is stored locally in microseconds as the time drift between the two machines and is used to adjust all subsequent requests for a time value; either positively or negatively depending on which clock is ahead of the other.

# B API Header file `refclock.h`

```
#ifndef REFCLOCK_H
#define REFCLOCK_H
  /* File refclock.h
   *
   * This header file describes the Code Magus Limited reference clock API. This
   * API has five exposed functions:
   *    1) refclock_delta() - returns the time difference in microseconds
   *       between the local machine and the Code Magus Limited Reference Clock
   *       Server.
   *    2) refclock_delta_message() - returns a text string describing the
   *       time difference between the local machine and the Code Magus Limited
   *       Reference Clock
   *       Server.
   *    3) refclock_gettimeofday() is a replacement for the gettimeofday()
   *       function. The time returned to the caller is the time as per the
   *       Code Magus Limited Reference Clock Server.
   *    4) refclock_time() is a replacement for the time() function. The
   *       time returned to the caller is the time as per the Code Magus Limited
   *       Reference Clock Server.
   *    5) refclock_resync() resynchronises the local machine with the Code
   *       Magus Limited Reference Clock Server.
   *
   * The TCP/IP address of the Code Magus Limited Reference Clock Server is
   * obtained from the environment variable $CODEMAGUS_TIMED_HOST. The value of
   * this variable is of the following format:
   *    <ipaddress|hostname>|<port number>
   *    for example: www.codemagus.com:60060
   *                 192.168.20.20:60060
   *
   * Author: Jan Vlok.
```

```
 *
 * Copyright (c) 2010 Code Magus Limited. All rights reserved.
 *
 */

/* $Author: janvlok $
 * $Date: 2012/03/16 13:29:15 $
 * $Id: refclock.h,v 1.12 2012/03/16 13:29:15 janvlok Exp $
 * $Name:  $
 * $Revision: 1.12 $
 * $State: Exp $
 *
 * $Log: refclock.h,v $
 * Revision 1.12  2012/03/16 13:29:15  janvlok
 * Auto resyn implementation
 *
 * Revision 1.11  2011/07/19 12:37:07  janvlok
 * Implent a function to rsync the clock
 *
 * Revision 1.10  2010/08/25 09:29:53  janvlok
 * Refclock delta message implementation
 *
 * Revision 1.9  2010/04/15 14:59:07  janvlok
 * Documentation and review changes
 *
 * Revision 1.8  2010/02/09 16:35:36  janvlok
 * Made delta stor a global define
 *
 * Revision 1.7  2010/02/03 09:08:12  hayward
 * Changes to environment variable name
 * from doc review.
 *
 * Revision 1.6  2010/02/03 06:55:27  hayward
 * Remove incorrect statement that refclock_delta
 * must be called before the time returning
 * functions.
 *
 * Revision 1.5  2010/02/02 22:05:54  hayward
 * Changes required for documentation.
 *
 * Revision 1.4  2010/02/02 18:05:17  hayward
 * Change documentation.
 *
 * Revision 1.3  2010/01/27 16:01:55  janvlok
 * Extra documentation
 *
 * Revision 1.2  2010/01/27 13:35:10  janvlok
 * CMLTIMD to CMLTIMED
 *
 * Revision 1.1  2010/01/27 10:59:53  janvlok
 * Take on
 *
```

```
   */

static char *cvs_refclock_h =
   "$Id: refclock.h,v 1.12 2012/03/16 13:29:15 janvlok Exp $";

  /*
   * Defines and constants:
   */

#define CODEMAGUS_REFCLOCK_SERVER "CODEMAGUS_REFCLOCK_SERVER"
#define REFCLOCK_DELTA_STORE "REFCLOCK_DELTA_STORE"
#define REFCLOCK_DELTA_MESSAGE "REFCLOCK_DELTA_MESSAGE"
#define REFCLOCK_RESYNC_INTERVAL 36000

  /*
   * Exported functions:
   */

  /* Function refclock_error() returns a message relating to the last error
   * returned by one of the other functions of the refclock API.
   */

char *refclock_error(void);

  /* Function refclock_delta_time() sets the clock difference in microseconds
   * between the remote and local machines in the supplied argument delta_time.
   * The remote time can be calculated by adding this delta returned to the
   * local time.
   * On success, zero is returned, otherwise -1 and the function
   * refclock_error() returns a formatted string detailing the error.
   */

int refclock_delta_time(long long int *delta_time);

  /* Function refclock_delta_message() returns a text string describing the
   * time difference between the remote and local machines.
   */

char *refclock_delta_message(void);

  /* Function refclock_gettimeofday() sets tv to the time since the Epoch
   * (00:00:00 UTC, January 1, 1970) returned from the result from calling
   * gettimeofday on the local machine and adjusting it with the difference as
   * calculated from the Code Magus Limited Reference Clock server.
   * On success, zero is returned, otherwise -1 and the function
   * refclock_error() returns a formatted string detailing the error.
   */

int refclock_gettimeofday(struct timeval *tv,void *tz);

  /* Function refclock_time() returns the time in seconds since the Epoch
   * (00:00:00 UTC, January 1, 1970) from the seconds portion of the result from
```

```
   * calling gettimeofday() on the local machine and adjusting it with the
   * difference as calculated from the Code Magus Limited Reference Clock
   * server. If t is not NULL then the time is also stored at the location
   * pointed to by t.
   * On success, zero is returned, otherwise -1 and the function
   * refclock_error() returns a formatted string detailing the error.
   */

time_t refclock_time(time_t *t);

  /* Function refclock_resync() resynchronises the local machine with
   * and the Code Magus Limited Reference Clock Server.
   */

void refclock_resync(void);

#endif /* REFCLOCK_H */
```

# C Sample Program

```
  /* File: sample.c
   *
   * Sample program to demonstrate the usage of Code Magus
   * reference clock API.
   *
   * Author: Jan Vlok
   *
   * Copyright (c) 2010 Code Magus Limited. All rights reserved.
   *
   */

  /*
   * $Author: hayward $
   * $Date: 2012/07/24 09:49:23 $
   * $Id: sample.c,v 1.4 2012/07/24 09:49:23 hayward Exp $
   * $Name:  $
   * $Revision: 1.4 $
   * $State: Exp $
   *
   * $Log: sample.c,v $
   * Revision 1.4  2012/07/24 09:49:23  hayward
   * Add Large File Support (LFS)
   *
   * Revision 1.3  2010/04/15 17:46:30  janvlok
   * Document changes
   *
   */

static char *cvs =
   "$Id: sample.c,v 1.4 2012/07/24 09:49:23 hayward Exp $";
```

```
  /*
   * Large File Support Required from various environments:
   */

#define _LARGEFILE_SOURCE
#define _LARGEFILE64_SOURCE
#define _LARGE_FILES
#define _FILE_OFFSET_BITS 64

#include <stdlib.h>
#include <stdarg.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <popt.h>
#include <time.h>

#include <rprintf.h>
#include <osmods.h>
#include <memdebug.h>
#include <refclock.h>



  /* Function main()
   */

int main(int argc,char *argv[],char *envp[])
   {
   int rc;
   unsigned long long int delta;
   char *cp, wstring[80];
   time_t t1, t2;
   struct timeval tv;


  /* Get the time difference:
   */
   rc = refclock_delta_time(&delta);
   if (rc < 0)
      {
      fprintf(stderr,"%s\n",refclock_error());
      return -1;
      }
   printf("Delta = %lld microseconds\n",delta);

  /* Get the time in seconds:
   */
   t1 = refclock_time(&t2);
   if (t1 < 0)
      {
```

```
    fprintf(stderr,"%s\n",refclock_error());
    return -1;
    }
 printf("time() = %d(%d) seconds\n",t1,t2);

/* Get gettimeofday:
 */
 rc = refclock_gettimeofday(&tv,NULL);
 if (rc < 0)
    {
    fprintf(stderr,"%s\n",refclock_error());
    return -1;
    }
 printf("gettimeofday() = %d.%6.6d seconds\n",tv.tv_sec,tv.tv_usec);

/* Print time, using a NULL argument for refclock_time():
 */

 printf("time() = %d seconds\n",refclock_time(NULL));

 return 0;
 } /* main */
```