|  |  |
| --- | --- |
|  |  |

EPTF CLL Ring Buffer Support, User Guide

Contents

[1 Introduction 2](#_Toc211150552)

[1.1 Revision history 2](#_Toc211150553)

[1.2 About this Document 2](#_Toc211150554)

[1.2.1 How to Read this Document 2](#_Toc211150555)

[1.2.2 References 2](#_Toc211150556)

[1.2.3 Abbreviations 2](#_Toc211150557)

[1.2.4 Terminology 3](#_Toc211150558)

[1.3 System Requirements 3](#_Toc211150559)

[2 Ring Buffer Support 3](#_Toc211150560)

[2.1 Overview 3](#_Toc211150561)

[2.2 Description of files in this feature 3](#_Toc211150562)

[2.3 Description of required files from other features 4](#_Toc211150563)

[2.4 Installation 4](#_Toc211150564)

[2.5 Configuration 4](#_Toc211150565)

[3 Examples 5](#_Toc211150566)

[3.1 Demo Module 5](#_Toc211150567)

# Introduction

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2007-11-12 | PA1 | First draft version | eistfal |
| 2007-11-21 | PA2 | Updated after review | eistfal |
|  |  |  |  |
|  |  |  |  |

## About this Document

### How to Read this Document

This is the User Guide for the Ring Buffer Support of the Ericsson Performance Test Framework (TitanSim), Core Load Library (CLL). TitanSim CLL is developed for the TTCN-3 ‎[1] Toolset with TITAN ‎[2]. This document should be read together with the Function Description of the Ring Buffer Support feature ‎[6]. For more information on the TitanSim CLL please consult the Product Revision Information ‎[3], the Users Guide ‎[4] and the Function Specification ‎[5] of the TitanSim.

### References

1. ETSI ES 201 873-1 v3.2.1 (2007-02)  
   The Testing and Test Control Notation version 3. Part 1: Core Language
2. 1/198 17-CRL 113 200 Uen  
   User Guide for the TITAN TTCN-3 Test Executor
3. 109 21-CNL 113 512-2 Uen   
   TitanSim CLL for TTCN-3 toolset with TITAN, Product Revision Information
4. 155 17-CNL 113 512 Uen   
   TitanSim CLL for TTCN-3 toolset with TITAN, Function Specification
5. 198 17-CNL 113 512 Uen  
   TitanSim CLL for TTCN-3 toolset with TITAN, User Guide
6. 13/155 16-CNL 113 512  
   EPTF CLL Ring Buffer Support, Function Description
7. TitanSim CLL for TTCN-3 toolset with TITAN, Reference Guide  
   <http://ttcn.ericsson.se/products/libraries.shtml>

### Abbreviations

CLL Core Load Library

EPTF Ericsson Load Test Framework, formerly TITAN Load Test Framework

TITANSim Ericsson Load Test Framework, formerly TITAN Load Test Framework

TTCN-3 Testing and Test Control Notation version 3 **Error! Reference source not found.**

### Terminology

*TITANSim Core (Load) Library(CLL)* is that part of the TITANSim software that is totally project independent. (I.e., which is not protocol-, or application-dependent). The TITANSim CLL is to be supplied and supported by the TCC organization. Any TITANSim CLL development is to be funded centrally by Ericsson

## System Requirements

In order to use the Ring Buffer Support feature the system requirements listed in TitanSim CLL User Guide ‎[5] should be fulfilled.

# Ring Buffer Support

## Overview

The EPTF CLL Ring Buffer component is a fundamental component providing an implementation for ring buffers in a load test environment.

Ring buffer (or circular buffer) is a [data structure](http://en.wikipedia.org/wiki/Data_structure) that uses a single, fixed-size [buffer](http://en.wikipedia.org/wiki/Buffer_%28computer_science%29) as if it were connected end-to-end. This structure can be easily used for buffering [data streams](http://en.wikipedia.org/wiki/Data_stream).

TitanSim Ring Buffer is a sequence of elements with maximized size, called capacity. It allows random access to elements and provides constant time insertion and removal at both front-end and back-end of the sequence. If you insert a new element to a full buffer (i.e. number of elements is equal with buffer capacity), the element at the other end is being overwritten.

TitanSim Ring Buffer component provides ring buffer implementation both for integer and generic types. Generic (type independent) ring buffer support is implemented through macro-preprocessor of C++ compiler.

## Description of files in this feature

The EPTF CLL Ring Buffer Support API includes the following files:

* EPTF\_CLL\_IntegerRingBuffer\_Functions.ttcn: This TTCN-3 module contains the public interface of ring buffer for integer values.
* EPTF\_CLL\_GenericRingBuffer\_Functions.ttcnin: This TTCN-3 include file contains the public interface of generic ring buffer implementation can be used for any user defined TTCN3 data type.
* EPTF\_CLL\_RingBuffer\_Definitions.ttcn: This module contains common type definitions for ring buffer support.
* EPTF\_CLL\_GenericRingBuffer\_Definitions.ttcnin: This include file contains type definitions for generic ring buffer support.
* EPTF\_CLL\_RingBuffer\_PrivateFunctions.ttcn: This module contains common private functions are used by integer ring buffer and generic ring buffer.

NOTE: TitanSim Ring Buffer component consists of two .ttcnin files. These support the type independent, generic ring buffer management with the usage of C++ macros. The Makefile Generator treats the .ttcnin files as include files. They will be added to the Makefile as special include file which will not be translated by the compiler, but will be checked for modification when building the test suite.

## Description of required files from other features

The TitanSim Ring Buffer does not have any dependencies to other features of TitanSim CLL.

## Installation

Since EPTF\_CLL\_RingBuffer is used as a part of the TTCN-3 test environment this requires TTCN-3 Test Executor to be installed before any operation of these functions. For more details on the installation of TTCN-3 Test Executor see the relevant section of ‎[2].

If not otherwise noted in the respective sections, the following are needed to use EPTF\_CLL\_RingBuffer:

* Copy the files listed in section [‎2.2] to the directory of the test suite or create symbolic links to them.
* Import the Ring Buffer demo or write your own application using Ring Buffer Support.
* Create Makefile or modify the existing one. For more details see the relevant section of ‎[2].
* Edit the config file according to your needs, see following section [**Error! Reference source not found.**].

## Configuration

The TitanSim Ring Buffer Support does not require any configuration setting.

# Examples

The “demo” directory of the deliverable contains the following files:

* RingBuffer\_demo.ttcn: This module contains the demo use cases.
* MyMessageTypeRingBuffer\_Functions.ttcnpp: This module contains the type definitions needed by the demo use cases. It illustrates how to instantiate the code for supporting a ring buffer built from a user defined type.

## Demo Module

The demo module (RingBuffer\_demo.ttcn) illustrates a typical usage of the ring buffer. It defines a special message type and generates instances of it. The generated messages are stored in a ring buffer until a certain message arrives and then all the messages are dumped from the ring buffer. It is possible to traverse the ring buffer from both begin-to-end and end-to-begin.

Step 1 Preparation: define an arbitrary TTCN3 type and instantiate the code for managing ring buffer of this new defined type

module MyMessageTypeRingBuffer\_Functions

{

type record MyMessageType {

enumerated {OK, ERROR} severity,

charstring data

}

#define EPTF\_BASE\_TYPE MyMessageType

#include "EPTF\_CLL\_GenericRingBuffer\_Functions.ttcnin"

#undef EPTF\_BASE\_TYPE

}

Step 2 Initialization: define and initialise a ring buffer

import from MyMessageTypeRingBuffer\_Functions all;

var MyMessageTypeRingBuffer v\_buffer;

f\_EPTF\_MyMessageType\_RB\_init(v\_buffer, 50);

Step 3 Collect Data: store messages in ring buffer (assume, that f\_GenerateMessage()generates a new message)

var MyMessageType v\_myMessage;

do

{

f\_GenerateMessage(v\_myMessage);

f\_EPTF\_MyMessageType\_RB\_push\_back(v\_buffer, v\_myMessage);

}

while ( v\_myMessage.severity != ERROR);

Step 4 Dump: empty ring buffer from begin-to-end,

while (not f\_EPTF\_MyMessageType\_RB\_empty(v\_buffer))

{

log( f\_EPTF\_MyMessageType\_RB\_front(v\_buffer) );

f\_EPTF\_MyMessageType\_RB\_pop\_front(v\_buffer);

}

Dump from reverse direction:

while (not f\_EPTF\_MyMessageType\_RB\_empty(v\_buffer))

{

log( f\_EPTF\_MyMessageType\_RB\_back(v\_buffer) );

f\_EPTF\_MyMessageType\_RB\_pop\_back(v\_buffer);

}