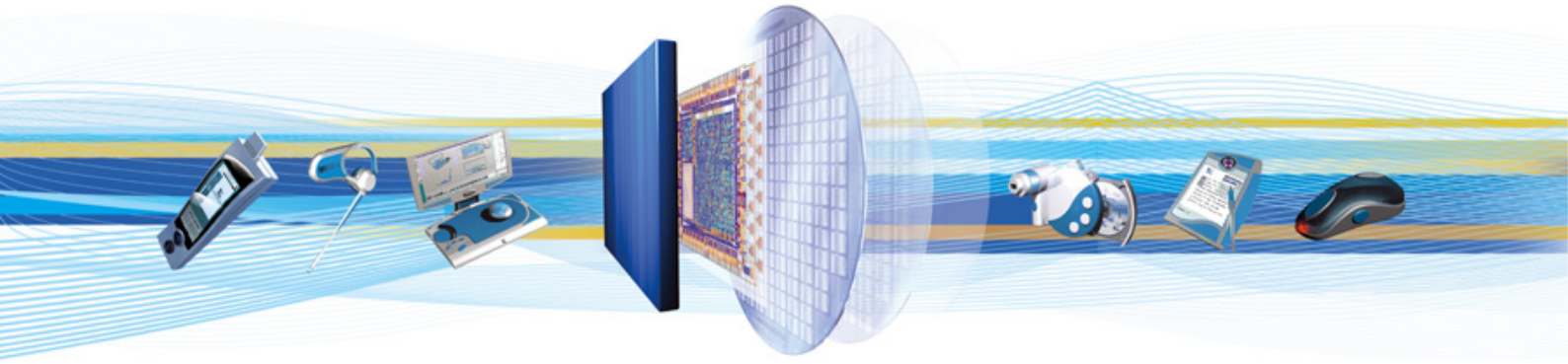
CSR Synergy Framework 3.1.0

Log Transport API

API Description

August 2011



Cambridge Silicon Radio Limited

Churchill House
Cambridge Business Park
Cowley Road
Cambridge CB4 0WZ
United Kingdom

Registered in England and Wales 3665875

Tel: +44 (0)1223 692000
Fax: +44 (0)1223 692001
www.csr.com

Contents

List of Figures

1 Introduction

The intention of this document is to describe how to implement the Log Transport API that is required by the logging functionality within the CSR Synergy Framework.

The CSR Synergy Framework Log Transport API makes it possible on a given platform to determine how logging is to be transported from the platform to some place where it can be stored. The Log Transport API does not make any requirements on the transport mechanism, which means that transport schemes like shown below may be used.

- To a file
- Via Ethernet
- Via UART
- Memory
- …

The Log Transport API also does not make any assumptions about the semantics of how the log data are handled. It is entirely up to the implementation of a given log transport to implement the mechanism that is needed on a particular platform. Examples of such mechanisms may be:

- Delaying actual log data writes until a certain amount has been accumulated
- Limiting the amount of retained logging data e.g. by using a ring buffer
- …

The CSR Synergy Framework logging system is used by all technologies running in the framework and the framework itself and is capable of logging events, raw HCI data, BCSP state machine events, etc.

When a log trace is available it is much easier to track down issues and determine where and why issues occurred. This will help reduce the turn-around time for solving issues. Because of this, CSR strongly recommends that logging is implemented.

2 CSR Synergy Framework Log Functionality

2.1 Basics

The CSR Synergy Framework Logging system consists of two parts, namely:

1) the `Log API` that is used by the technologies
2) the `Log Transport API` which is used for transporting the log messages to some "storage" media

In **Error! Reference source not found.Error! Reference source not found.** these two parts are depicted.

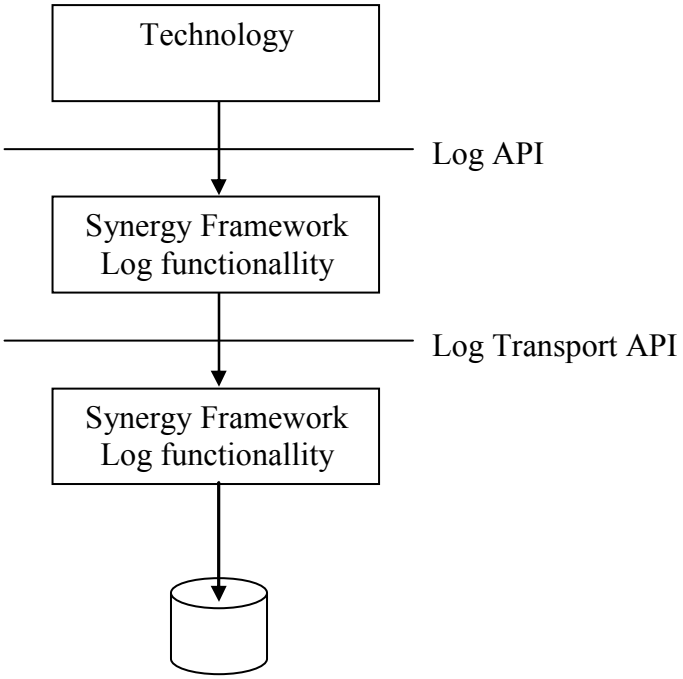


Figure 1: Synergy Log System

2.2 Log Transport API

The Log Transport functionality should be implemented with the following in mind:

- As the log write function is running in context of the task/function calling it the log write function should be implemented with speed in mind. If the media on which the log data are written is a "slow" media I may be necessary to implement a buffering system between the log write and the actually write to the storage media.
- If the operating system on which CSR Synergy Framework is running utilises multi-threading, the Log Transport function **must** implement proper mutual exclusion locking (mutex). Refer to the CSR Synergy Framework Log Transport reference implementation (`./bsp/ports/pcwin/src/logtrans`) for hints on where to put these locks.
- If the target system does not feature an actual filesystem capable of holding the log-files, the logs can be transferred to a host system via a network connection, UART or similar.

The logging transport API has two fundamental operations:

- Allocation of a memory buffer in which a logging provider (Pcap, Btsnoop, …) can store logging data
- Returning a memory buffer with data to the logging transport (log data write)

The API is very simple but it has been designed in a way that makes it flexible enough to cover all cases ranging from a single log transport for a single log format to multiple concurrent transports and formats. The design uses a *logging transport handle* which is basically an opaque pointer.

This handle is created by the application and registered with a given log provider (e.g. Pcap) which then uses this handle to inform the application which logging transport it wants to perform an operation (allocation, log data write) on. An example of log and log transport initialisation and deinitialisation is shown below:

```
void *ltPcap, *ltSnoop;
CsrLog *logPcap, *logSnoop;

ltPcap = logtransportCreateFile("pcap.log");
ltSnoop = logtransportCreateUart(3);

logPcap = CsrLogPcapCreate(ltPcap);
logSnoop = CsrLogBtsnoopCreate(ltSnoop);

CsrLogRegister(logPcap);
CsrLogRegister(logSnoop);
[..]
CsrLogUnregister(logPcap);
CsrLogUnregister(logSnoop);

CsrLogPcapDestroy(logPcap);
CsrLogBtsnoopDestroy(logSnoop);

logtransportCloseFile(ltPcap);
logtransportCloseUart(ltSnoop);
```

In this example two logging transports are set up, one for a file named `pcap.log` and one for UART3. The file handle is assigned to a Pcap logging instance, and the UART handle is assigned to a Btsnoop logging instance. Following this, the two logging handles are registered with the logging framework. Finally, the deinitialisation procedure, symmetric to the initialisation, is shown.

2.2.1 CsrLogTransportWrite

Prototype

```
#include "csr_logtransport.h"

CsrSize CsrLogTransportWrite(void *ltHdl, CsrLogContext context,
                             void* data, CsrSize size);
```

Description

Writes logging data to the logging transport specified by the logging transport handle `ltHdl`. The logging transport handle is an opaque pointer created by the application that the logging framework only uses to identify which log transport it is making a given request on.

Parameters

Type	Argument	Description
`void *`	`ltHdl`	The log transport handle that is being logged on
`CsrLogContext`	`context`	Which context that is being logged. For at definition of contexts see **Error! Reference source not found.Error! Reference source not found.**
`void *`	`data`	Pointer to data that are to be written. This pointer has previously been allocated with CsrLogTransportAlloc(), and it is the responsibility of CsrLogTransportWrite() to free it if necessary.
`CsrSize`	`size`	Amount of the data to write in bytes

Table 1: Arguments to `CsrLogTransportWrite`

Returns

The actual number of bytes written. If a number less than size is returned, it indicates that the write failed for some reason.

2.2.2 CsrLogTransportAlloc

Prototype

```
#include "csr_logtransport.h"

void *CsrLogTransportAlloc(void *ltHdl, CsrSize size);
```

Description

Called by the log system when it needs a data buffer to store logging data in. The buffer returned will not be deallocated by the logging code – it is passed back to the log transport implementation when the buffer has been filled.

Parameters

Type	Argument	Description
`void *`	`ltHdl`	The log transport handle that the requested data buffer will be logged on
`CsrSize`	`size`	The minimum required size of the buffer in bytes

Table 2: Arguments to `CsrLogTransportAlloc`

Returns

A pointer to a data buffer large enough to hold at least `size` bytes.

3 Document References

Ref	Title

Terms and Definitions

BlueCore®	Group term for CSR's range of Bluetooth wireless technology chips
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
CSR	Cambridge Silicon Radio
HCI	Host Control Interface
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards

Document History

Revision	Date	History
1	10 JUL 09	Initial version
2	30 Nov 09	Ready for release 2.0.0
3	20 APR 10	Ready for release 2.1.0
4	OCT 10	Ready for release 2.2.0
5	DEC 10	Ready for release 3.0.0
6	Aug 11	Ready for release 3.1.0

TradeMarks, Patents and Licences

Unless otherwise stated, words and logos marked with ™ or ® are trademarks registered or owned by CSR plc or its affiliates. Bluetooth® and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc. and licensed to CSR. Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by CSR plc.

CSR reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, CSR cannot accept responsibility for any errors.

No statements or representations in this document are to be construed as advertising, marketing, or offering for sale in the United States imported covered products subject to the Cease and Desist Order issued by the U.S. International Trade Commission in its Investigation No. 337-TA-602. Such products include SiRFstarIII™ chips that operate with SiRF software that supports SiRFInstantFix™, and/or SiRFLoc® servers, or contains SyncFreeNav functionality.

Life Support Policy and Use in Safety-critical Compliance

CSR's products are not authorised for use in life-support or safety-critical applications. Use in such applications is done at the sole discretion of the customer. CSR will not warrant the use of its devices in such applications.

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information.