

L

F

1

S

D  
K

U

U

U

S

E

R

M

A

N

U

A

L

XCube





30 Technology Way, Nashua, NH 03060

(978) 799-2024

www.x3-c.com

Logger stream reader

(LF1)

software developers kit

Release: Gazelle 121.0

#### 2012-07-12



Logger Software

Stream Reader

Software Developers Kit

Copyright © XCube, 2012. All rights reserved. No part of this manual may be reproduced, copied, translated, transmitted or published in any form or by any means without prior written permission from XCube.

All registered and unregistered trademarks and company names contained in this manual are property of their respective owners.

# DISCLAIMERS

The information in this user’s manual has been carefully reviewed and is believed to be accurate. XCube assumes no responsibility for any inaccuracies that may be contained in this document, (including the software and products described in it) and makes no commitment to update or to keep current the information in this manual, or to notify any person or organization of the updates.

XCube retains the right to make changes to product descriptions and/or specifications at any time, without prior notice. Specifications and information contained in this manual are furnished for informational use only and should not be construed as a commitment by XCube.

XCube provides this manual “as is” and assumes no responsibility whatsoever, and disclaims any warranty of any kind, either express or implied, relating to the sale and/or use of XCube products including but not limited to the implied warranties or conditions of merchantability or fitness for a particular purpose or merchantability. In no event shall XCube, its directors, officers, employees or agents be liable for any indirect, special, incidental, or consequential damages (including damages for lost profits, loss of business, loss of use or data, interruption of business and the like), even if XCube has been advised of the possibility of such damages arising from any defect or error in this manual or product.

Any disputes arising between XCube and the customer shall be governed by the laws of Hillsborough County in the State of New Hampshire, USA. The State of New Hampshire, County of Hillsborough shall be the exclusive venue for the resolution of any such disputes. XCube’s total liability for all claims will not exceed the price paid for the hardware product.

FCC Statement: This equipment has been tested and found to comply with the limits for a Class B digital device pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy, and if not installed and used in accordance with the manufacturer’s instruction manual, may cause interference with radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does harmful interference to radio or television reception, which can be determined by turning the equipment off and on, you are encouraged to try to correct the interference by one or more of the following measures: Reorient or relocate the receiving antenna. Increase the separation between the equipment and the receiver. Connect the equipment into an outlet on a circuit different from that to which the receiver is connected. Consult the dealer or an experienced radio/television technician for help.

# DOCUMENT HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Note | Create/Changed by |
|  |  |  |  |
|  |  |  |  |

TABLE OF CONTENTS

[DISCLAIMERS 3](#_Toc329865832)

[DOCUMENT HISTORY 4](#_Toc329865833)

[OVERVIEW 6](#_Toc329865834)

[Abbreviations and Definitions 6](#_Toc329865835)

[Symbols 6](#_Toc329865836)

[Related Documents 6](#_Toc329865837)

[Build information 7](#_Toc329865838)

[Stream reading overview 7](#_Toc329865839)

[Generate File Names 8](#_Toc329865840)

[Time Filters 9](#_Toc329865841)

[Packet Reader 10](#_Toc329865842)

[Important / Useful Classes 12](#_Toc329865843)

# OVERVIEW

The purpose of this document is to describe how to use the Logger File Reader Software Developers Kit. This SDK provides access to XCube stream data stored in the LF1 format. With this SDK, the developer is shielded from low level file internals. Using the SDK is the preferred access method for the stream data.

The SDK provides the following functions:

* Project access classes
* Index access through time filters
* Sequential reading of the stream
* Access to the stream’s data

## Abbreviations and Definitions

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

## Symbols

|  |  |
| --- | --- |
|  | NOTES  Tips and additional information to help you complete a task. |

## Related Documents

* DOxygen files are generated from the SDK headers. These should be used for detailed descriptions of the classes and their methods.

# Build information

The SDK contains all XCube libraries and headers needed to read an LF1 stream. A sample application has been include to show how to read a stream. Also included are DOxygen files generated from header files.

* The sample application (lf1ReaderSample) was compiled under the Eclipse environment. As part of the SDK, the entire workspace has been provided. In a production development environment, the x3c directory may be moved to a more appropriate place.
* All headers for classes are provided under x3c/include. Additional include subdirectories exist under the x3c/include directory.
* The following internal libraries are provided under x3c/lib:

Indexer

packetReader

projectManagement

x3cCommon

* The following external libraries are required for linking:

rt

pthread

* SDK libraries and sample code were compiled on Ubuntu 12.04 with GNU 4.6.3.

# Stream reading overview

The following steps are used when reading a LF1 stream:

1. Generate a list of file names for the stream.
2. Create a time filter to set both the start and end times on the stream
3. Add the file names and time filter to a packet reader.
4. Start the packet reader.
5. Read a packet
6. Process the packet
7. Free the packet
8. Repeat reads until done

In the following sections, each step will be explained in detail. Included in each section will be the classes to use, data formats, and any additional details that might be useful.

# Generate File Names

To use the packet reader, we must pass in a list of the file names for a given stream. To understand what that means, we need to look at how the XCube Recorder stores the stream data.

When the Recorder is capturing data, it writes the stream data to a series of disk drives. Typical configurations use 16 or 32 disks. As the Recorder receives stream data, it writes out that data to all 16 disks. Disk 10 might have packet 1, disk2 packet 2, disk4 packet 3, and so on. How and where the packets are stored on the disks is unimportant; it is just important to know that we cannot just grab 1 file and assume that we have the entire stream.

Given that our stream data is on 16 (or 32) disks, the file hierarchy to get to a stream will be something like:

<mount point>/<base name>/<project name>/data/<stream name>

Where:

|  |  |
| --- | --- |
| Mount point | The file path up to the directory name which gets a disk/folder number |
| Base Name | Folder name under the mount point. The base name is appended to the mount point + disk number |
| Project Name | Project name |
| Data | Hardcoded value |
| Stream Name | Stream name |

Examples:

char sz0[] = "/mnt/data\_sets/vol2/disk0/xcube/lgr00000\_2011\_05\_04\_153253/data/T51\_E1\_R0\_2011\_05\_04\_153320.lf1";

char sz1[] = "/mnt/data\_sets/vol2/disk1/xcube/lgr00000\_2011\_05\_04\_153253/data/T51\_E1\_R0\_2011\_05\_04\_153320.lf1";

char sz2[] = "/mnt/data\_sets/vol2/disk2/xcube/lgr00000\_2011\_05\_04\_153253/data/T51\_E1\_R0\_2011\_05\_04\_153320.lf1";

…

char ex2-0[] = "/mnt/disk0/xcube/lgr00000\_2012\_06\_04\_153253/data/T51\_E1\_R0\_2012\_06\_04\_153320.lf1";

char ex2-1[] = "/mnt/disk1/xcube/lgr00000\_2012\_06\_04\_153253/data/T51\_E1\_R0\_2012\_06\_04\_153320.lf1";

char ex2-2[] = "/mnt/disk2/xcube/lgr00000\_2012\_06\_04\_153253/data/T51\_E1\_R0\_2012\_06\_04\_153320.lf1";

…

In the first set (sz0 – sz2), the mount point is “/mnt/data\_sets/vol2/disk” and the base name is “/xcube”. In the second set, the mount point is “/mnt/disk”. It is important to see that on both sets that a number is appended to the mount point.

When the packet reader is initialized, it requires a character array of these file names with the full path.

The Projects class provides some of this functionality. When instantiated, the Projects class takes the mount point, base name, and number of disks. From this class, we can get a list of created projects and set a project to be active. On the active project, we can get a list of captured streams which are return as a STREAM structure. Given a selected STREAM, we can return all of the full file paths for that stream.

The following snippet of code shows the steps for retrieving the stream’s file paths.

string mnt ("/mnt/data\_sets/vol2/disk");

string base ("/xcube");

Projects projects(mnt, base, 16);

vector <PROJECT> prjs = projects.pmGetProjects();

projects.pmActivateProject("lgr00000\_2011\_05\_04\_153253");

vector<STREAM> strms;

if (projects.pmGetStreams(strms) != 0)

{

fprintf(stdout, "Error retrieving streams for current project\n");

return -1;

}

vector<STREAM>::iterator it1;

for (it1 = strms.begin(); it1 < strms.end(); it1++)

{

STREAM st = \*it1;

if(st.strmName.compare("T51\_E1\_R0\_2011\_05\_04\_153320.lf1") != 0)

continue;

vector<string> streamFiles;

projects.pmGetStreamFiles(&st, streamFiles);

vector<string>::iterator it2;

for (it2 = streamFiles.begin(); it2 < streamFiles.end(); it2++)

{

string dirName = \*it2;

// at this point, dirName has the full path/name of the stream

fprintf(stdout, "Stream Names with path: %s\n", dirName.c\_str());

}

}

# Time Filters

A time filter (CTimeFilter) can be used to set the starting and ending points to the stream. This filter must be applied before the packet reader is initialized.

The start and end times for the filter are set by passing two unsigned longs for both the start and end time. The first value is for seconds and the second if for nanoseconds.

CTimeFilter startFlt;

startFlt.setStartTime(0x4dc1aa08, 0x14700);

startFlt.setEndTime(0x4dc1aa08, 0x1200000);

…

pktReader->setTimeFilter(startFlt);

If no time filter is created, then the entire stream will be used.

If the stream has been indexed, the packet reader will seek to the nearest start time. If no index exists, the packet reader will do a linear search to the start time.

# Packet Reader

The CPacketReader is the main reading class. To use this class, we need to:

1. add the time filter,

CTimeFilter startFlt;

startFlt.setEndTime(0x4dc1aa08, 0x1200000);

startFlt.setStartTime(0x4dc1aa08, 0x14700);

pktReader->setTimeFilter(startFlt);

1. initialize the CPacketReader,

pktReader->initialize(4, fileNames, (size\_t) READ\_BUFFER\_SIZE, (size\_t) READ\_BUFFER\_CHUNK);

1. start,

nRetVal = pktReader->start();

1. get the next packet,

X3cPacket \*pkt = pktReader->getPacket();

If pkt == NULL, then the stream is finished.

1. process the packet,
2. free the packet

pktReader->freePacket(pkt);

1. finally, repeat packet reads and processing until done.

From the stream packet, we can get the packet header and a pointer to the payload. The header has the packet timestamp and the payload length. The payload is the raw data that was captured. It is important to note

|  |  |
| --- | --- |
|  | NOTES  It is important to remember that the packet returned is returning a pointer to some data. When a freePacket(pkt) is called, the data is no longer valid. If you want to use the data after a freePacket() call, **you MUST make a copy of the data**. |

When initializing CPacketReader, the input buffer and read size are arguments passed in. Both values are in Mbytes. These values will be applied for each individual stream file that exists. So if a stream is spread across 16 disk files, 16 input buffers will be allocated. It is important to remember that you will be allocating (number of files) \* (input buffer size).

The full code looks like:

// Create the packet reader

CPacketReader \*pktReader = new CPacketReader();

/\*\*\*

\* Create a time stamp filter. This will be used for the start/end times of the

\* stream.

\*

\* If the stream has been indexed, it will also seek to that position. Also included

\* are a couple of time sample tests.

\*/

CTimeFilter startFlt;

startFlt.setEndTime(0x4dc1aa08, 0x1200000);

startFlt.setStartTime(0x4dc1aa08, 0x14700);

pktReader->setTimeFilter(startFlt);

/\*\*

\* initialize the readers.

\*/

pktReader->initialize(4, fileNames, (size\_t) READ\_BUFFER\_SIZE, (size\_t) READ\_BUFFER\_CHUNK);

/\*\*

\* start reading in the packets

\*/

try

{

nRetVal = pktReader->start();

if(0 == nRetVal)

{

bool isDone = false;

while (false == isDone)

{

X3cPacket \*pkt = pktReader->getPacket();

if (NULL == pkt)

{

fprintf(stdout, "Finished reading. packet <%d>\n", currentPacket);

currentPacket = 10;

isDone = true;

}

else

{

/\*\*

\* at this point, we have a packet. now we need to

\* do something with it. In the sample code, we are just

\* writing out the timestamp for the LF1 header.

\*/

UINT32 sec, nsec;

sec = pkt->getHeader()->ts.tv\_sec;

nsec = pkt->getHeader()->ts.tv\_nsec;

fprintf(stdout, "Packet: %d : 0x%x:0x%x\n", currentPacket, sec, nsec);

/\*\*

\* MAKE SURE TO DELETE the packet when finished!!!

\*/

pktReader->freePacket(pkt);

currentPacket++;

}

}

pktReader->stop();

}

# Important / Useful Classes

The following are the primary classes and structures to use. For detailed information on the classes, see the Doxygen generated files.

|  |  |
| --- | --- |
| **Class** | **Description** |
| CPacketReader | Main class used to read a LF1 stream. Internally multithread. |
| Projects | Class to retrieve project information |
| PROJECT | The project structure. Contains some project information and the stream list. |
| STREAMS | Project stream structure. Contains information about the stream. It does not wrap the stream data. |
| CTimeFilter | A start / end time filter that can be applied to the packet reader |
| X3cPacket | Wrapper class over the actual packet read from disk. Contains the packet time stamp, packet size, and a pointer to the packet data. |
|  |  |

The following classes are lower level and should not be used directly:

CFileReader

CFrameFilter

All indexing classes (CIndexer, CIndexReader, CIndexWriter)

CPacketBuffer

CStreamFilter

CX3cFile

X3cFileBuf