The Lire Roadmap

February 2003

Joost van Baal

Francis J. Lacoste

Introduction

This document gives a roadmap for the development of the Lire software. It serves as a reference point when working on the software and states the current ideas and plans of the LogReport developers.

The roadmap contains two parts: the first one contains scheduled work, the second one contains not-yet scheduled work. The scheduled work is ordered by planned Lire release. The unscheduled work is ordered by expected amount of work; generally, long-term projects are the ones that imply deep infrastructural changes. The schedules items are deemed needed to make the Lire code viable as a volunteer-driven Open Source project.

The LogReport development team publishes a new Lire release about every two months. We plan to continue this scheme: March 2003 we'll ship Lire 2.0, late April 2003 we'll ship Lire 2.1.

For each item, we try to list the person name and email responsible for the feature. This will usually be a person who can be reached via the <`development@logreport.org`> mailing list. You should also find the current status of the item and the items that need to be implemented before. Furthermore, and estimation of the needed man-hours is listed. This are the man-hours left to spent, as of the date this document is published.

If you want to work on an item described in this roadmap, send an email to the <`development@logreport.org`> mailing list.

> **Important:** One should also consult the `BUGS` file in CVS which contains many small bugs and wishlist items.

Furthermore, please read Francis's message Toward Lire 1.3 and 2.0 : issues to solve in the current log analysis framework (mid:%3c1039118700.2981.156.camel@Arendt.Contre.COM%3d) dated December 5 2002, to the <`development@logreport.org`> mailing list for some second, more detailed, thoughts on this roadmap.

Lire 2.0 features

Lire 2.0 will get released early March 2003.

Configuration API

Status: Usable, not yet fully integrated
Responsible: Wessel Dankers
Prerequisites: None
Est. Man-hours: 60

Lire's framework should contain a configuration API that should be used by all of its components. See the messages PROPOSAL: Lire Configuration Framework (mid:%3c20020430172451.GJ18114@Contre.COM%3d) and PROPOSAL: Configuration Framework and Extensibility (mid:%3c1039128325.21156.212.camel@Arendt.Contre.COM%3d) to the <`development@logreport.org`> mailing list for more details.

The library code for the new Configuration API is in CVS, but it's not yet fully integrated with the rest: everywhere Lire uses configuration variables, it should use the new API. It can do all the parsing and group stuff now.

Integration is being done in a way that provides enough in "backward compatibility" to offer a migration path for our users' configuration while also minizing the changes needed to our source tree. Like for the DLF API scenario, we will be able to release even if the whole source tree isn't converted to the new API yet.

The configuration API integration roadmap looks like the following:

1. Write a tool **lr_env2conf** which exports the environment-based configuration to the XML file format. (Migration tool for our users' configuration).
2. Write a perl module Lire::Config::Legacy which exports the XML-based configuration to the old environment variables. (Makes it possible to use the new API from the perl code which uses the old API.)
3. Write a tool **lr_environment** which exports the XML-based configuration to the old environment variables. (Makes it possible to use the new API from the shell scripts while minizing the required changes). This tool should replace the old **lr_env** tool.
4. Make sure that the API makes it easy for Lire's extensions (DLF converters, analysers, etc.) to add configuration parameters.
5. Rewrite **lr_config** to take advantage of the new API features.
6. Document the public part of the API.
7. Modify exising code to use the new API.

Items 1, 2 and 3 must be completed for a release to happen. Items 4 and 5 add new features which are really desirable, but we can cut there in the worst scenario.

Storage API

Status: Implemented in simple form
Responsible: Joost van Baal e.a.
Prerequisites: the Section called *Configuration API*
Est. Man-hours: 60

Lire should offer an API to a persistent store which could be used by all components to store and retrieve parts of data.

A start of a storage API (Lire::DlfStore) is implemented, which is adequate for now; we can improve on it as needs arise.

The old archive implementation is used for two purposes. It is used for both inter-components communication during one jobs life-time and for long-term storage. The first part should be moved to a control file mechanism like the one described in the message PROPOSAL: New Online Responder Architecture (mid:%3c20020421210816.GY12459@Contre.COM%3d) to the <`development@logreport.org`> mailing list. The second functionality is the proper domain of the persistent store API, Lire::DlfStore.

The persistent store should also be arranged around the user/server hierarchy described in the configuration framework proposal.

See also Francis's message PROPOSAL: Datastore oriented mode of operation (mid:%3c1039122289.21157.160.camel@Arendt.Contre.COM%3d) to the <`development@logreport.org`> mailing list.

Improved Merging Interface

Status: Started
Responsible: None
Prerequisites: the Section called *Configuration API*, the Section called *Storage API*
Est. Man-hours: 20

Although the current distribution supports merging of reports, it should be better integrated in the **lr_cron** and **lr_config** interfaces.

Complete DLF Converter API

Status: Implemented, not yet fully integrated
Responsible: Francis J. Lacoste
Prerequisites: the Section called *Storage API*
Est. Man-hours: 80

The DLF Converter API is complete and tested. What is left is integration in the **lr_dlf2xml** pipeline, as well as writing documentation and porting the convertors.

Not all convertors are planned to get converted to use this new API. Since backward compatibility with the old interface is part of the new API, this won't cause any problems. Backwards compatibility is offered via the Lire::OldDlfAdapter module, which translates the old API into the new one. This choice is made because migrating all convertors before the 2.0 release would take time, be risky and wouldn't give much benefit. One convertor will get ported, and examples will be written on how to port and use the new API, in order to gain mindshare.

When bugs are found in old convertors, or extra features are needed, they'll get ported to the new API eventually.

See also Francis's message (http://logreport.org/contact/lists/development/msg00810.php) Running unit tests in current CVS (mid:%3c1044051213.19533.94.camel@Arendt.Contre.COM%3d) to the <`development@logreport.org`> mailing list about Lire unit tests and Test::Unit. The current CVS contains several unit tests which were added as part of the DLF converter API upgrading. Unit tests make it easier to modify the internals while maintaining confidence that everything still holds together. See also Chapter 15. Making Lire "Test-infected" in the Lire Developers Manual.

1. The DLF converters are no longer aware of the underlying DLF implementation. (i.e. print to STDOUT or whatever). A `output_dlf` is used for that purpose (or it could be part of the dlf_maker functionality).
2. Space and 8bit printable characters are no longer removed.
3. The DLF API supports log continuation. See the thread About log continuation feature (http://logreport.org/contact/lists/development/msg00596.php) on the <`development@logreport.org`> mailing list.

Lire 2.1 features

Lire 2.1 will get released late April 2003

Allow Multiple Schemas in the DLF Conversion Process

Status: Implemented, to be integrated
Responsible: Francis Lacoste
Prerequisites: the Section called *Complete DLF Converter API*
Est. Man-hours: 120

Currently, one log file can only generate reports about one superservice. Removing that limitations (i.e. one DLF converter can generate DLF for multiple schemas from the same log file) enables us to solve the "cross-superservice" problem. A report can contain subreport from all the supported schemas (login, daemon, proxy, etc.)

See the thread Tackling the Cross-Superservice Problem (http://logreport.org/contact/lists/development/msg00586.php) for a more in-depth discussion. This also includes a discussion on why this approach is better than the previous multiple-event types DLF approach (as proposed in Proposals to LogReport (http://logreport.org/contact/lists/development/pdf00004.pdf)).

This feature is implemented as part of the new DLF converter API, by changing the internal of the DLF conversion process from a pipe paradigm to a store paradigm The changes needed to support this in the report generation process are minimal once the DLF API is fully integrated.

Separation of the Analysis Process

Status: Started
Responsible: None
Prerequisites: the Section called *Complete DLF Converter API*
Est. Man-hours: 20

Currently, the analysers that produce the derived schema and extended schema DLF data work in the report generation process. Those should be moved into a separate process between the log normalisation process and the report generation process. This would permit some optimisations in the data format and is necessary to support other report generation backends.

As part of the integration of Lire::DlfStore into lr_dlf2xml work is being done on some items which are related to separation of the analysis process. This is needed to integrate the new DLF API into **lr_dlf2xml**.

SQL Based Backend

Status: Experimental Code in CVS
Responsible: None
Prerequisites: the Section called *Separation of the Analysis Process*
Est. Man-hours: 120

An important target for Lire is to develop an SQL based reporting engine which should offer more scalable reporting and probably make it worthwhile to use the framework in an interactive development.

Arnaud Gaillard has written some proof-of-concept code to make Lire work with an SQL backend.

Small unscheduled items

Directory Superservice

Status: Not started
Responsible: None
Prerequisites: None

Arnaud Taddei has expressed interest in developping a directory superservice for LDAP servers.

More firewall services

Status: Not started
Responsible: None
Prerequisites: None

Users have requested ‘Firewall-1’, ‘snort’ and ‘Watchguard Soho’ firewall DLF converters.

Dynamic Registration of Output formats

Status: Not started.
Responsible: None
Prerequisites: None

Although the Lire framework is meant to be extended and there are already several APIs provided to do so, some components are still statically registered. For example, each output format must be registered in several static lists. (Services and superservices are registered dynamically.) It would be better if those lists could be built dynamically. We also could provide simple tools to register new components (something like a **lire-install** command).

Big unscheduled items

Featureful Online Responder

Status: Conceptual
Responsible: Joost van Baal e.a.
Prerequisites: the Section called *Configuration API*, the Section called *Storage API*

The online responder we offer on our website should be able to support all the features the command line Lire supports. Installation of a responder should be better documented and easier. There was an initial proposal sent to the <`development@logreport.org`> mailing list in the message PROPOSAL: New Online Responder Architecture (mid:%3c20020421210816.GY12459@Contre.COM%3d).

Overhaul of the Email Superservice

Status: Conceptual
Responsible: None
Prerequisites: None

The current DLF schema of the email superservice is starting to show its limits. A lot of information is lost in the email log files and several important reports (like refused connections, spam control, etc.) cannot be generated. The schema should be rewritten to closer resemble the log that one actually sees with different fields for anti-spam activity, message collection, routing, etc. This will make writing email DLF converter a lot easier. The current DLF schema could become a derived schema and the stateful logic that is replicated across all email service could be moved to a generic analyser.

Internationalisation Framework

Status: Not started
Responsible: None
Prerequisites: None

Standard internationalisation components like xml-i18n-tools or gettext should be integrated into the framework. The XML components should also be modified to support other charsets than basic ASCII.

Support reports in multiple languages.

Status: Not started
Responsible: None
Prerequisites: the Section called *Internationalisation Framework*

Lire should get i18n-ed, and support other languages in its error messages and other output, as well as in the report specifications. People have requested l10n to French. This is a long-term task.

C Based Implementation of the APIs

Status: Not started
Responsible: None
Prerequisites: None

To make the Lire framework usable in more contexts, it might be worthwile to reimplement the APIs in C so that mapping for other languages than perl could be made available. This would also make it possible to support lex/yacc based DLF converters. It could also help performance.

Configuration GUI

Status: Not started
Responsible: None
Prerequisites: the Section called *Configuration API*

There should be a better configuration interface than the lr_config script we offer now. The CGI interface should get completed. A GUI interface should get added.

Some research on GUI libraries has been done by Plamen Bozukov in September 2001 (Message-ID: <Pine.LNX.4.10.10109271704180.25208-200000@pozvanete.bg>). He came to the following conclusion:

Table 1. Comparison of GUI libraries

score 1-5	GPL	portability	requirements	easy	features	binding
Qt	4	5	5	5	5	3

score 1-5	GPL	portability	requirements	easy	features	binding
V	5	5	5	4	4	2
FLTK	5	5	5	4	3	5
GTK	5	4	5	5	4	4
WxWindows	5	5	4	5	5	5
Tk	5	5	5	3	4	5

QT

License: QPL/GPL. For windows version, license is possibly problematic. Portability: Microsoft Windows 95/98/2000, Microsoft Windows NT, MacOS X, Linux, Solaris, HP-UX, Tru64 (Digital UNIX), Irix, FreeBSD, BSD/OS, SCO and AIX. Requirements: C++ Compiler X libraries for Unix. Binding: Perl-binding is very old - Last updated November 17th, 1997. Python and Ruby.

V

License: GNU LGPL. Portability: Windows,OS/2,Unix.

FLTK

License: LGPL. Portability: Unix,Windows,OS/2. Requirements: C++ Compiler X libraries for Unix. Bindings: Perl: 2 different solutions; Python

GTK

License: GNU LGPL. Portability: Unix, Windows. Requirements: C Compiler X libraries for Unix. Bindings: all possible languages. There is the glade interface builder which makes it easy to design GTK interface.

WxWindows

License: GNU Library General Public. Portability: Unix,Windows,Mac. Requirements: C++ Compiler GTK libraries for Unix. Bindings: Perl, good binding for Python.

Tk

Portability: Windows, Unix, Mac. Bindings: all possible scripting languages.

More information on various GUI toolkits is on The GUI Toolkit, Framework Page (http://www.free-soft.org/guitool/).