messagefacility - Feature #3982

Global message logger extensions

06/04/2013 10:54 AM - Kurt Biery

Status:	Closed	**Start date:**	09/15/2014
Priority:	High	**Due date:**	10/31/2014
Assignee:	Kyle Knoepfel	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.13.00	**Spent time:**	128.00 hours

Description

Jim has created a proposal for adding syslog and sqlite destinations to the Message Facility. It is included below:

Proposal for a global message logger extension to MF:

Introduction -
We have a reasonable common message logging facility library built into most, if not all of, the processes and libraries within artdaq and art. The code reports errors and information using the API from this facility. The facility has a very flexible back-end API permitting messages to be sent to files, the console, or statistics calculators. These configurable back-end components are called destinations. Destinations can be used to connect the logger to centralized facilities within a system. These types of destinations are always experiment specific going all the way back to CDF and D0 on their trigger processing farms. NOvA has one that uses DDS to deliver the log message to central server.
The DS50 DAQ project document proposed adding a message logger destination utilizing XMPP (the instant messaging protocol). This is still a good direction, but cannot be achieved in a short period of time because the protocol and associated server tools are fairly complex. Given the importance of a global logger, XMPP cannot be rushed into.

Proposal -
I propose that we add two new destinations and an archiving tool directly to the message facility package. These new components will be directly used within DS50 DAQ for the global logger. At the end of this section I will highlight possible integration issues. The first destination translates message facility messages to syslog messages. The second destination writes message facility messages to an sqlite3 database. Both can be built directly into the message facility library because of the minimal dependencies that they introduce.

Section 4.3.3 of the ds50 project document lists the message facility message format. Here are the data passed to a destination associated with each logged message. It is as common-separate-value names.

timestamp, host name, host address, severity, category, optional_application_name, optional_process_ID, optional_current_run_event_number, software_module_name, user_supplied_text_message

1) syslog destination:
There is one connection permitted in an application. We recommend opening it with the following call. The first argument will need to be discussed further. It is likely to be not very relevant.
openlog(“MF”, 0, LOG_LOCAL0)
This configuration will direct all output from MF to a syslog facility of LOG_LOCAL0. The syslogd configuration will them be written to forward these message to a global syslogd destination for final logging.
The configuration of the final destination should be written to attach to an MF message receiver script (described next). The syslogd daemon will feed all remote data in stdin of this script.
The MF syslog destination should write messages to syslog with the following call:
syslog(<map severity from message to this>, <serialized header plus payload as string>)
The serialized header should be a CSV version of the header information as shown above with the user payload message appended to it. This format will permit very easy parsing on the receiving script end.

The MF message receiver script - this script should likely be written in Ruby, so that text processing and database interactions are made easy. It needs to unpack the header and payload. We recommend defining an sqlite3 database that can be used to store all incoming messages.

sqlite3 database tables - we recommend a minimum of two tables. One for headers, and one for general payload. Many interesting queries can be performed on header information and this information is more predictable in size than the variable-length payload. Optional additional tables should be used for specific categories of message. One of the metadata fields is the category - a user assigned string value. There are fixed names used for certain types of messages used by art and hopefully used by artdaq in the future. These specific typed message have a well-known and established payload format. The purpose of the extra optional tables is to store the payload of known category messages since the payload can be readily decoded into known fields and values. An example is the timing records out of art. The timing information is fixed and can be decoded and directed into a “category_timing” payload table for easier analysis. All unrecognized categories will be sent to the free-form text payload table.

2) sqlite3 destination:
This is a simple destination that does something similar to the receiving script in the previous section. Its purpose is to write all message logger data into an sqlite3 database directly out of the process.

Integration issues -
1) current message logger initialization is likely not correct. It is likely that the art initialization function is initializing the logger, along with the main programs that call it (aggregator and event builder applications).
2) Linux comes with an alternative syslogd message collector daemon called syslog-ng. It is far better than the default one for a distribute logger configuration and should likely be used.
3) The diagnostic services of art (timing, memory, and trace) are likely writing to stdout or stderr. These will need to change in the mainstream art release to message logger messages, or at least have a configuration switch to permit both modes.
4) Having the two destinations directly built into the message facility allows for easy configuration using the standard MF fhicl fragments.

Concluding comments -
The Ruby message receiving script can be copied out of the message facility library and modified to write to any database or to any other global recording system.
Syslogd is a known solution for this sort of global message logging problem. It is widely used and tested. The development effort is low for this solution.

Subtasks:

Feature # 7008: Implement most urgently-required features of Issue #3982	**Closed**
Feature # 7140: Implement plugin sqlite3 destination of Issue #3982	**Closed**

History

#1 - 06/28/2013 10:37 AM - Kurt Biery

- Due date set to 08/30/2013

Discussion notes from Alessandro/Jim/Kurt:

Jim described his proposals.

Alessandro asked about a direct write to a database. Jim mentioned that this is the second option.

Candidate databases: sqlite3 and postgres. Since postgres is used elsewhere on DarkSide, it seems like a reasonable choice for this also.

Alessandro says that a database writer might be enough.

Where do we encode the database passwords? Maybe we just restrict the source of inserts to certain machines.

We described the message viewer that comes with the MessageFacility.

We should change the reported name of the message source to give us more information about whether the boardreader is running a 1495. Jim reminds me that there are metadata fields that are useful for this.

Alessandro just needs the database. If we also provide the syslogng transport, that is fine with him. The crucial point is having the messages in the DB.

#2 - 08/19/2013 11:58 AM - Christopher Green

- Status changed from New to Accepted

- Target version set to 1.09.00

- Start date deleted (06/04/2013)

- Estimated time set to 24.00 h

The time estimate assumes the simplest solution, namely writing to syslog and assuming an external postprocessor to handle the translation into the customer's database of choice. The more general solution would include an overhaul of the currently inadequate messagefacility plugin mechanism, and would therefore need a design discussion in order to come up with a believable time estimate.

#3 - 10/30/2013 11:20 AM - Christopher Green

- Status changed from Accepted to Assigned

- Assignee set to Paul Russo

#4 - 02/17/2014 12:26 PM - Christopher Green

- Target version changed from 1.09.00 to 521

#5 - 08/21/2014 11:05 AM - Kurt Biery

- Due date changed from 08/30/2013 to 10/31/2014

- Priority changed from Normal to High

Yesterday, I talked with Alessandro Razeto and others from DarkSide, and the subject of centralized message logging and display came up again. The context of our discussion was making the final changes to the ds50daq and global DS-50 DAQ systems so that RunControl has complete control of the system. The idea is to start pmt.rb (the artdaq Process Management Tool) in the background and send commands to it from RunControl when a system reconfiguration is requested by the user (for example, including or excluding one or more BoardReaders to include or exclude some number of hardware modules).

When we switch to such a model, it would be nice to have centralized message logging and viewing since there will no longer be an operator console that is showing the artdaq and PMT messages.

DarkSide is planning to start an extended data-taking period in December, and it would be nice to have this functionality in place for that data-taking.

Please consider my increase in priority and change in due date as a request to have the syslog-ng option for the message facility available for DarkSide this fall.
Thanks,
Kurt

#6 - 09/15/2014 11:17 AM - Christopher Green

- Status changed from Assigned to Accepted

- Assignee deleted (Paul Russo)

#7 - 11/24/2014 03:03 PM - Kyle Knoepfel

- Status changed from Accepted to Resolved

- Assignee set to Kyle Knoepfel

This feature has been implemented according to subissues #7008 and #7140.

#8 - 11/24/2014 03:06 PM - Kyle Knoepfel

- Target version changed from 521 to 1.13.00

#9 - 02/16/2015 10:32 AM - Christopher Green

- Status changed from Resolved to Closed