DataLogger and Collector/Router

Code Review

June 1, 2000

Dorota Genser

- **Repository**
  - cvsuser@d02ka.fnal.gov:/cvsroot/d0cvs

- **Packages/Versions**

  Those are the newest versions which are not in production but the resent changes are rather minor.

  - **datalogger_util v00-02-06**

    Utilities and common/base classes used by DataLogger and Collector/Router

  - **datalogger v00-01-10**

    DataLogger plus dlsam scripts (Maarten's and my test scripts)

  - **collector v00-00-05**

    Collector/Router

• Build/Platforms supported

- released as a part of t00.92.00 D0 test release using KAI compiler
- mostly used and tested on OSF1
- build and run on Linux and IRIX
- no Windows NT support
- complied with gcc 2.95 but not tested

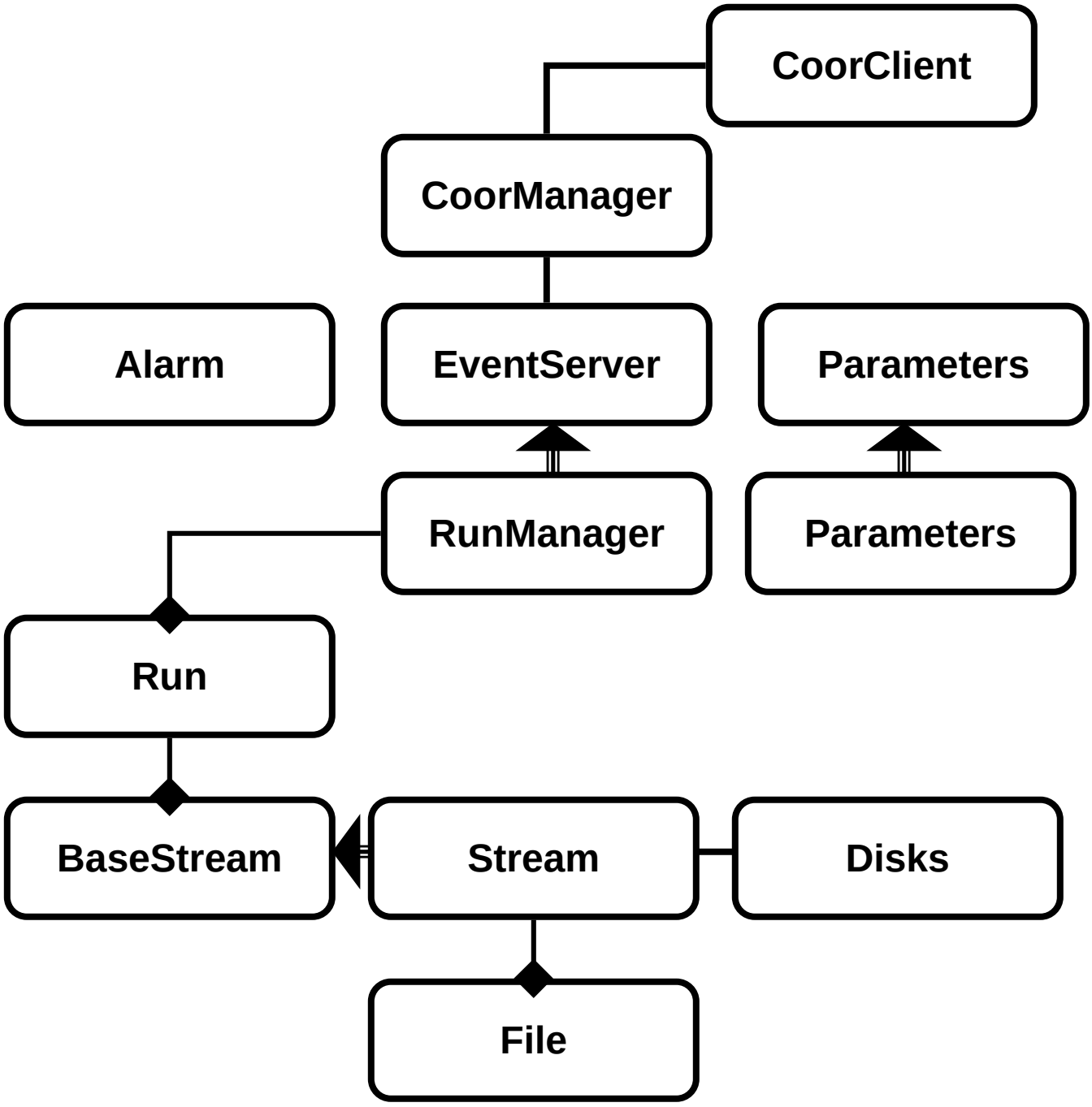
• Dependencies

- KAI version 3.3f
- ACE version 4.6
- thread_util v00-03-02
- itc v00-00-13
- itc_event v00-00-02
- evpack (datalogger)

• Documentation

There is HTML user guide document for each of the package.

DataLogger


CoorClient
CoorManager
Alarm
EventServer
Parameters
RunManager
Parameters
Run
BaseStream
Stream
Disks
File

• DataLogger Util Classes

– Alarm Class

- simple interface that allow user send alarm messages to Significant Event Server and also log them using ITC INFO macro.
- all methods are static and it has no members.
- Internally it is using State class that has static members to keep current state of the application and pointer to the ITC client connection object. Access to this class is protected using locks to be sure that Alarm calls are thread safe.
- has optional method to activate signal handling. If signal is received the state is changed and the application stops.
- has methods to start communication to Significant Event Server (SES)
- has set of methods (Info, Warn, Error, Fatal) that user can use to send alarm messages if communication with SES is started. The method "Fatal" also changes the state and the application stops.
- has method to check the state used by the application in most of the infinite loops.

– EventServer Class

- abstract class that wraps around ITC server object. This class is used in CoorManager class.
- performs all tasks like starting ITC server, setting network parameters that are common for Collector/Router and DataLogger.
- has a few pure virtual methods that are specific to the application.

– CoorManager Class

- handles all control comunication from COOR, Monitor Server or other external application as a ITC server.
- does all of the string decoding and handles most of the commands internally.
- Starts/stops EventServer based on the external commands (COOR) or internal state.
- is using CoorClient class to stores configuration information about each COOR client, which is passed to the user code as a parameter to run stop/start calls.

– Parameters Class

- reads arguments (same for DataLogger and Collector/Router)
- handles all string and file manipulations necessary to read setup information.
- writes output setup file and prints help

• DataLogger Classes

– Parameters Class

- derives from dlu::Paramters
- sets additional configuration parameters specific for DataLogger by overwriting method that sets single parameter.

– RunManager Class

- derives from dlu::EventServer
- has map of Run objects created and deleted based on the COOR commands passed as run stop/start calls from CoorManager. (There is also some extra code that creates/deletes Run objects on demand used for testing and before COOR communication. This part can be cleaned now.)
- checks all existing Run objects for each arriving event, assuming that event can belong to more than one run. It sends warning if the event can not be written by at least one Run.
- uses locks to protect its internal structure since this objects is accesses from three different threads (EventServer processor, CoorManager processor and main thread).

– Run Class

- created for each run and deleted when run is over.
- Has map of Stream objects. They can be BaseStream objects (recording OFF) or Stream objects for runs that have events written to files.
- Each event is checked if it belongs to any of the streams from the list.

– BaseStream Class

- simple class that has statistics only used in runs which has recording OFF switch turn on.

– Stream Class

- derives from BaseStream.
- has internal thread to write events to files
- has list of File objects, one for each file
- has one Disks object to mange list of disks
- can have one "lost events" file for events that arrived after file is closed and there is no other place to put them.

– Disks Class

- has list of disks and their current state
- is used to select next available disk.

– File Class

- decides if the event can be written and writes it to disk file.
- keeps all information about state of output file and statistics.
- creates metadata file and event catalog when data file is closed

– LFile Class

- same as File class but with no restrictions on data file size and event range.

• Future Changes

– DSM Communication

- list of valid streams needed to run multiple DataLoggers. This can be done using current interface but some additional interface can be better.
- dynamic list of disks for each stream.
- both changes are not difficult to implement but DSM is not ready…
- having smarter DSM could also simplify CoorManager class...

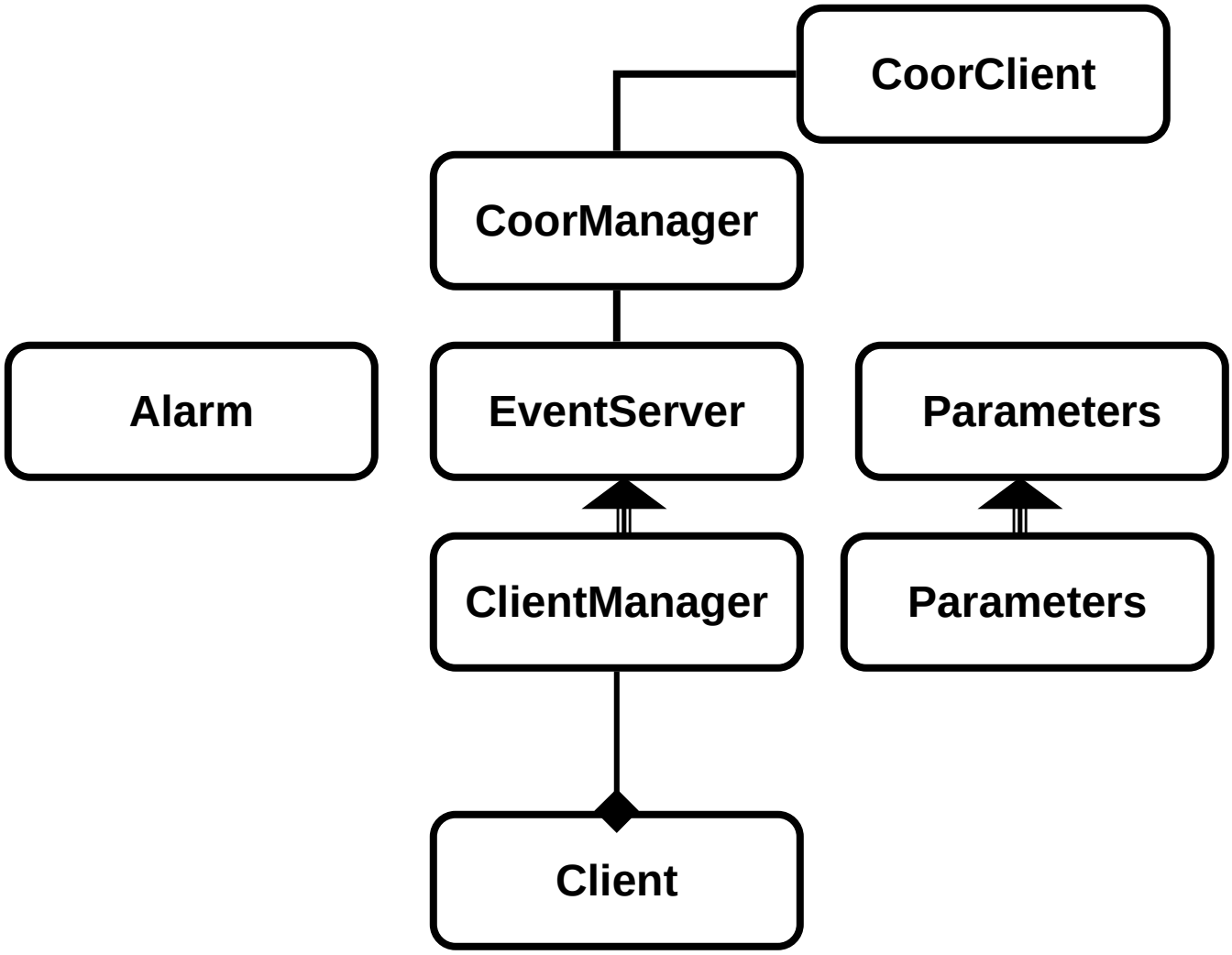
– Luminosity Block

- currently event number is used to assure that event ranges are not overlap between files. This should be changed to use luminosity block as soon as this number is available. This should be very easy…

– Other

- there is no limit of the number of Files created per stream. Depending of the filename string it is possible that DataLogger can create duplicated data files - no checking is done. Currently the limit is 999 partitions (three digits) but be ware..
- disk management was only used in my tests (not in production)…

Collector/Router



• Collector/Router Classes

– Client Class

- has ITC client connection to send events to CR receiver
- it has functionality to match allowed stream numbers with arriving events but this is not used.

– ClientManager Class

- equivalent to DataLogger RunManager class but holds Client objects, which are rather stable.

– Parameters Class

- equivalent to DataLogger Parameters class

• Future Changes

– DSM communication

- information about CR client are read from setup file. This should be changed to some dynamic setting from DSM.
- Stream number assignment for each client. At this moment all events are sent to all clients…

• DataLogger_Util Files

– /datalogger_util (C++ include files)

- DataLogger_util.hpp
  - main header that user can use. It defines all general variables and includes all useful headers.
- Alarm.hpp
  - Alarm class to send messages to Significant Event Server. Also handles signals and keeps state of the application in the internal static structure.
- CoorClient.hpp
  - Container class used by CoorManager for configuration information send by COOR.
- CoorManager.hpp
  - Class that manages all COOR messages. It decodes string commands and manages most of them internally.
- EventServer.hpp
  - Based class used to receive events. It is controlled by CoorManager based on the COOR commands
- Parameters.hpp
  - Base class that has most of functionality needed to read setup file, setup string and also keeps common setup parameters like event server post number etc.
- general.hpp
  - namespace string and macros.

Header that are old and obsolete or used only for test programs

- CL_Conn.hpp
  - (obsolete) Loose end connection, need for old d0me package and old Collector/Router toy program.
- Mixer.hpp
  - used in mL3_evpack test application to simulate L3 with lost/mixed events.
- Status.hpp
  - (obsolete) Replaced by new Alarm class.

–/src (C++ source files)

- DataLogger_util.cpp
  - some general stuff like string manipulation.
- Alarm.hpp
  - Alarm class, interface to Significant Event Server, signal handing and state of the application.
- CoorManager.cpp
  - Receiving, decoding and handing most of COOR commands.
  - Handing EventServer object.
- EventServer.cpp
  - Receiving events interface.
- Parameters.cpp
  - Reading arguments.
  - Reading and saving setup files.

Old files which are not used in the production code.

- CL_Conn.cpp
  - Connection class with statistics used for tests.
- Status.cpp
  - Old “Alarm” class.

– /bin (C++ programs)

- coor_server_util.cpp
  - This program is used to send commands to DataLogger or Collector/Router which are understood by CoorManager. This way application can be asked for internal information, status of the connections or stopped.

– /scripts

- DAQ.pl
  - This is simple perl script to start DataLogger and Collector/Router programs on given host. It is using coor_server_util program to check, query or stop applications.
- DAQ.setup
  - example of the setup file used by the DAQ.pl script.

– /doc

- dlu_quick_guide.html
  - user guide.

– /toy

- CRtoy.cpp
  - Old Collector/Router toy program that can be still useful for tests.
- mL3_evpack.cpp
  - Test program that simulates L3.

– /ups

- datalogger_util.table
  - ups table needed to install it in the D0 online ups database

• DataLogger Files

– /datalogger (C++ include files)

- DataLogger.hpp
  - It defines all general variables and includes all needed headers.
- Disks.hpp
  - Disks class used by Stream class
- File.cpp
  - File and LFile classes used by Stream class
- Parameters.hpp
  - Parameters class, (derives from dlu::Parameters) holds all static configuration needed for DataLogger
- Run.hpp
  - Run class, which can have number of Stream objects.
- RunManager.hpp
  - RunManager class which derives from dlu::EventServer. It holds and maintains Run objects.
- Stream.hpp
  - BaseStream and Stream class.

–/src (C++ source files)

- Disks.cpp
  - Disks class, which manipulates list of disks.
- File.cpp
  - File and LFile classes, open disk file, writes events and holds information about it.
- Parameters.cpp
  - Parameter class (derives from dlu::Parameters) that holds static configuration of DataLogger
- Run.cpp
  - Run class, has list of Stream or BaseStream objects.
- RunManager.cpp
  - derives from dlu::EventServer and holds or active Run objects.
- Stream.cpp
  - Stream and BaseStream classes

–/bin (C++ programs)

- datalogger.cpp
  - DataLogger main program.

– /scripts

- dl_sam_fix.pl
  - perl script used to test DataLogger. It has disk managing added.
- dlsam
- dlsam.config
- query_dlsam
- start_dlsam
- stop_dlsam
  - all scripts above are Maarten's dlsam scripts, which are not part of DataLogger.

– /doc

- dl_quick_user_guide.html
  - instructions how to use datalogger, release notes.
- help.txt
  - help printed by the datalogger program using "-h" argument
- setup_file.txt
  - example of the setup files with default configuration for DataLogger.

– /ups

- datalogger.table

• Collector/Router Files

– /collector (C++ include files)

- Collector.hpp
  - It defines all general variables and includes all needed headers.
- Client.hpp
  - Client class to hold information about the event receiver.
- ClientManager.hpp
  - ClientManager class which derives from dlu::EventServer. It holds and maintains Client objects.
- Parameters.hpp
  - Parameters class, (derives from dlu::Parameters) holds all static configuration needed for Collector

– /src (C++ source files)

- Client.cpp
- ClientManager.cpp.
- Parameters.cpp

- **/bin (C++ programs)**
  - collector.cpp
    - Collector/Router main program.
- **/doc**
  - cr_quick_user_guide.html
    - instructions how to use collector, release notes.
  - help.txt
    - help printed by the collector program using "-h" argument
  - setup_file.txt
    - example of the setup files with default configuration for Collector.
- **/ups**
  - collector.table