MPS History Server

Important MPS messages can arrive in bursts of about 200 per second!

Setup

Environment

- Effectively, you can only develop MPS History Server on lcls-builder.
- Check out the CVS module physics/mps/mpshist into your workspace in the LCLS VLAN
    - From now on, the root of project is referred to as $MPSHIST_ROOT
- Use ant to build the jars.
- Stop the production version of mpshist and run the version in your workspace.
- After you are done, remember to stop you version and run the production version

Dependencies

For library paths, see the build.xml file. Note that the dependent jar files are copied into the lib folder and are part of CVS. To update the jar files, run ant update_dependencies. This should copy the files from $PHYSICS_TOP on dev onto the lib folder. You can then commit these updated dependent jar files into CVS.

- ezJCA (version R0-0-10)
- javainterfaces (current version)
- MPS Config
- xal4lcls
    - ext.jar (current version)
    - xal.jar (current version)

Test

- See the development section on how to run your own version of mpshist.
- "Poke" MPS - note that the PV's mentioned here may not exist. Ask Sonya for some test PV's that can be used.

```
caput POSI:EP02:400:MTLM_LGC_BYPV 1
```

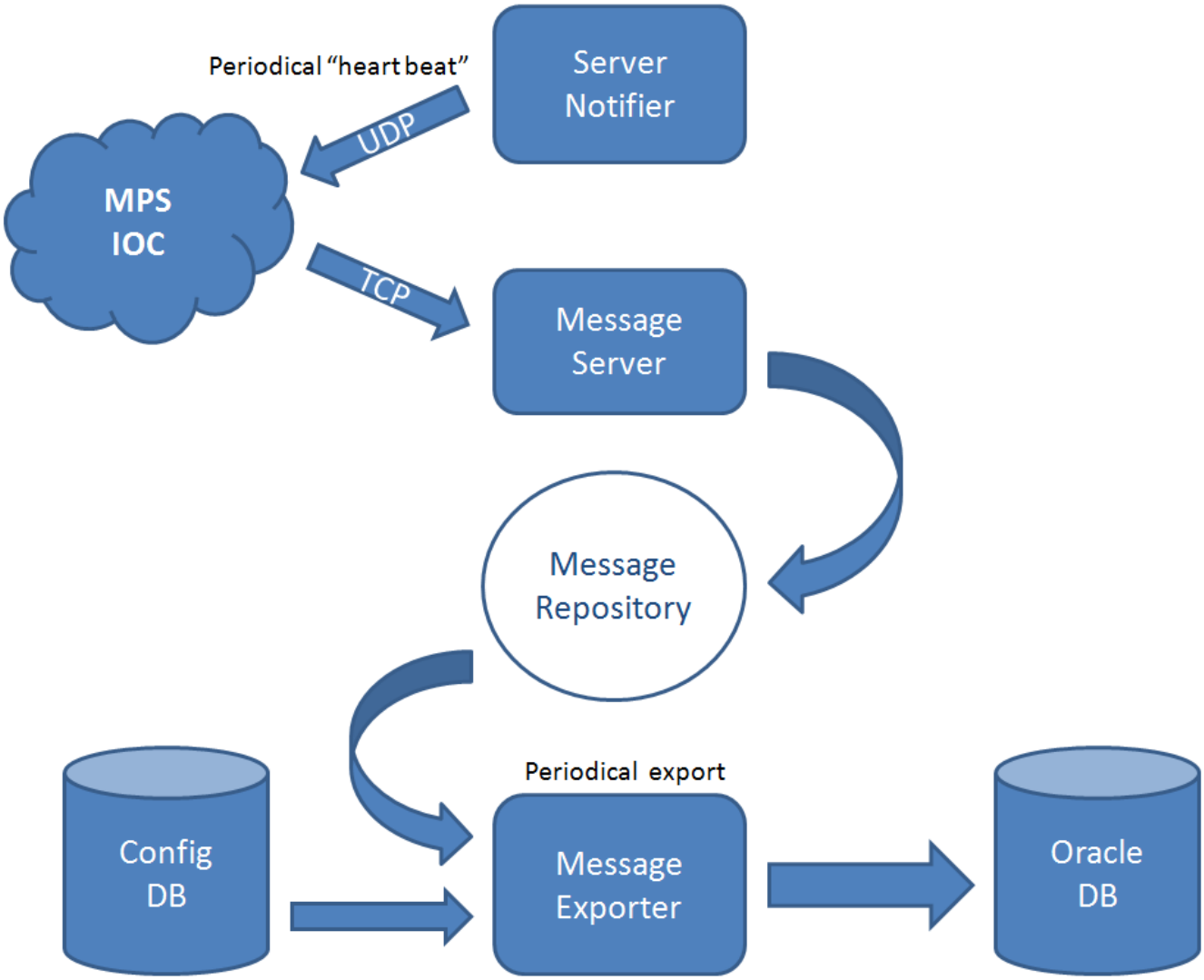
wait 10 seconds!

```
caput POSI:EP02:400:MTLM_LGC_BYPV 0
```

- Check the HistoryViewer tab on the MPS GUI
- Stop MpsHistManager
- **Important:** start the production server (see #Start / Stop)

Development

Overview



Launch Script

- mpshist under $MPSHIST_ROOT
- Gets the current version of the Config DB
    - Assembles the path to it
- Sets up Oracle Wallet
- Runs MpsHistManager and tells the process to write all message into a log file

Classes and Functions

Message

edu.stanford.slac.mpshist.message.Message

- Java representation of an MPS message

```
///************************************************************************
// * Data Pointers
// ************************************************************************/
///* Note: The history message header begins at the first byte of TCP Data
// */
//
//#define HISTORY_MESSAGE_HEADER_SECTION_POINTER(_pointerToFullMessage)      ((epicsInt8  *)
(_pointerToFullMessage)   + 0)
//#define HISTORY_MESSAGE_PROTOCOL_VERSION_POINTER(_pointerToHistoryHeader)  ((epicsUInt8 *)
(_pointerToHistoryHeader) + 0)
//#define HISTORY_MESSAGE_TYPE_POINTER(_pointerToHistoryHeader)              ((epicsUInt8 *)
(_pointerToHistoryHeader) + 1)
//#define HISTORY_MESSAGE_DATA_SECTION_POINTER(_pointerToHistoryHeader)      ((epicsInt8  *)
(_pointerToHistoryHeader) + 2)
//
//#define HISTORY_MESSAGE_PROTOCOL_VERSION(_pointerToHistoryHeader)
(*HISTORY_MESSAGE_PROTOCOL_VERSION_POINTER(_pointerToHistoryHeader))
//#define HISTORY_MESSAGE_TYPE(_pointerToHistoryHeader)                      (*HISTORY_MESSAGE_TYPE_POINTER
(_pointerToHistoryHeader))
//
///************************************************************************
// * Super Message
// ************************************************************************/
///* All history message struct data.
// */
//
//
//
//#define HISTORY_SUPER_MESSAGE_SIZE                                          (16 +
MPSHistoryProtocolHeaderSize)
//
//#define HISTORY_SUPER_TIMESTAMP_SECPASTEPOCH_POINTER(_pointerToHistoryData)  ((epicsUInt32 *)
((_pointerToHistoryData) +  0))
//#define HISTORY_SUPER_TIMESTAMP_NSEC_PULSEID_POINTER(_pointerToHistoryData)  ((epicsUInt32 *)
((_pointerToHistoryData) +  4))
//#define HISTORY_SUPER_TIMESLOT_POINTER(_pointerToHistoryData)               ((epicsUInt8  *)
((_pointerToHistoryData) +  8))
//#define HISTORY_SUPER_TYPE_POINTER(_pointerToHistoryData)                   ((epicsUInt8  *)
((_pointerToHistoryData) +  9))
//#define HISTORY_SUPER_ID_POINTER(_pointerToHistoryData)                     ((epicsUInt32 *)
((_pointerToHistoryData) + 10))
//#define HISTORY_SUPER_OLD_VALUE_POINTER(_pointerToHistoryData)              ((epicsUInt32 *)
((_pointerToHistoryData) + 14))
//#define HISTORY_SUPER_NEW_VALUE_POINTER(_pointerToHistoryData)              ((epicsUInt32 *)
((_pointerToHistoryData) + 18))
//
//#define HISTORY_SUPER_TIMESTAMP_SECPASTEPOCH(_pointerToHistoryData)
(*HISTORY_SUPER_TIMESTAMP_SECPASTEPOCH_POINTER(_pointerToHistoryData))
//#define HISTORY_SUPER_TIMESTAMP_NSEC_PULSEID(_pointerToHistoryData)
(*HISTORY_SUPER_TIMESTAMP_NSEC_PULSEID_POINTER(_pointerToHistoryData))
//#define HISTORY_SUPER_TIMESLOT(_pointerToHistoryData)                       (*HISTORY_SUPER_TIMESLOT_POINTER
(_pointerToHistoryData))
//#define HISTORY_SUPER_TYPE(_pointerToHistoryData)                           (*HISTORY_SUPER_TYPE_POINTER
(_pointerToHistoryData))
//#define HISTORY_SUPER_ID(_pointerToHistoryData)                             (*HISTORY_SUPER_ID_POINTER
(_pointerToHistoryData))
//#define HISTORY_SUPER_OLD_VALUE(_pointerToHistoryData)                      (*HISTORY_SUPER_OLD_VALUE_POINTER
(_pointerToHistoryData))
//#define HISTORY_SUPER_NEW_VALUE(_pointerToHistoryData)                      (*HISTORY_SUPER_NEW_VALUE_POINTER
(_pointerToHistoryData))
```

- There are various message types that we display in human readable form
    - Beam destination: Beam destination changed from PREVIOUS DESTINATION_NAME to CURRENT DESTINATION_NAME
    - Beam rate: Beam rate rate after DEVICE_NAME changed from PREVIOUS RATE_NAME to CURRENT RATE_NAME
    - Fault: DEVICE_NAME FAULT_NAME changed from PREV_STATE to CURR_STATE
    - Bypass time: DEVICE_NAME FAULT_NAME is bypassed for BYPASS_TIME_IN_SECONDS sec **OR** DEVICE_NAME FAULT_NAME bypass had cleared
    - Bypass value: DEVICE_NAME FAULT_NAME changed from PREV_VALUE to CURR_VALUE

edu.stanford.slac.mpshist.message.MessageRepository

- A singleton that caches MPS messages before they are exported to Oracle

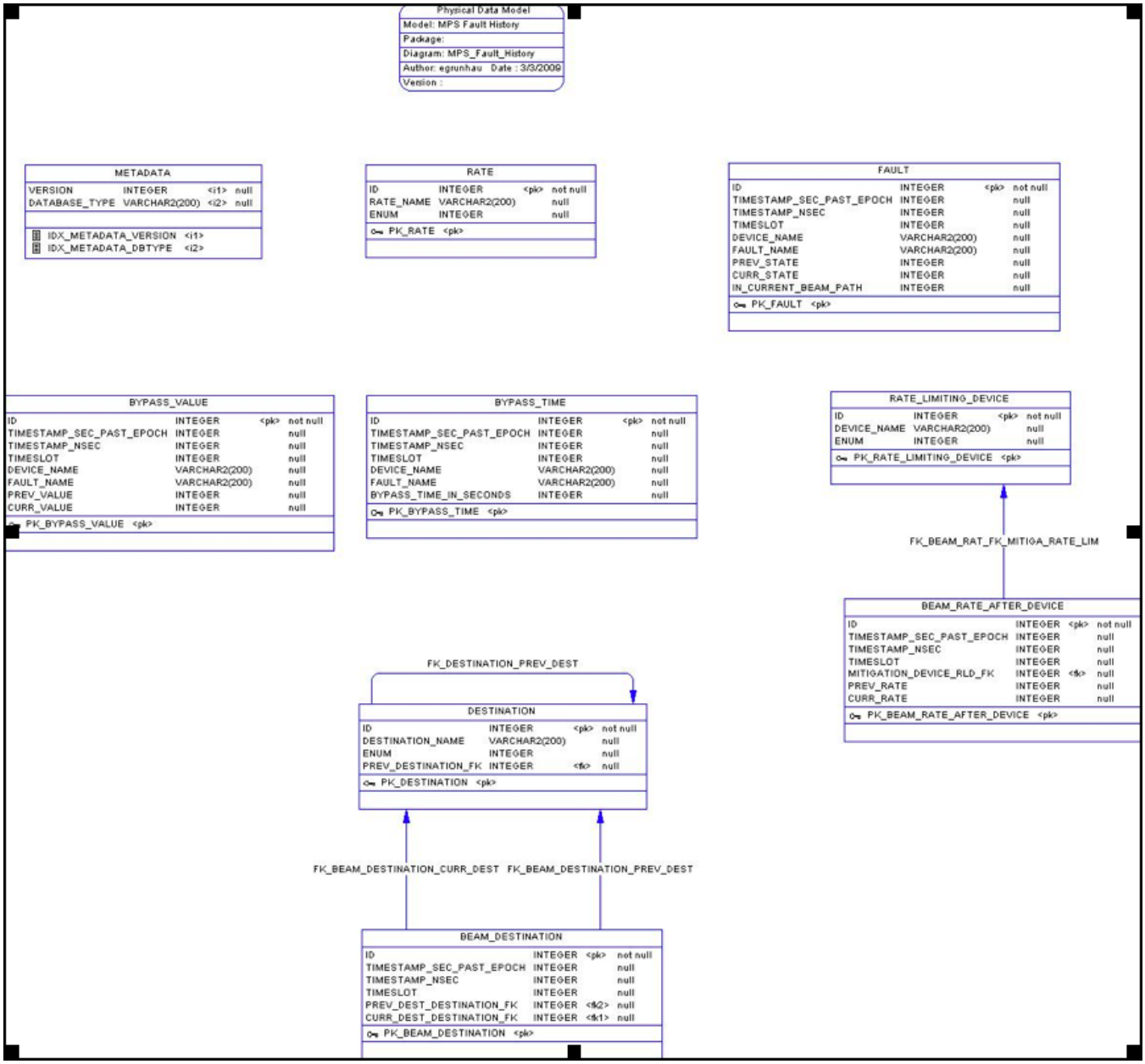
Other

edu.stanford.slac.mpshist.MessageCounter

- A thread-safe way to count messages, both received and exported, and periodically put the numbers into cmlog
- Primarily used for debugging

edu.stanford.slac.mpshist.MessageExporter

- An instance of Runnable that periodically saves messages to the Oracle DB



- Eli wrote a PL/SQL procedure that hides (almost) all the details

```
BEGIN MPS_HISTORY.MPS_HISTORY_PKG.DML_%s1 (%s2); END;
```

  - %s1 = table name
  - %s2 = CSVs in the same order as in the schema with following caveats:

        - top-to-bottom = left-to-right
        - ignore the ID column
        - replace the FKs with enums from the referenced table

edu.stanford.slac.mpshist.MessageServer

- An instance of Runnable that caches messages from MPS in memory
- Tries to create a ServerSocket bound to a port specified as an argument to the constructor

edu.stanford.slac.mpshist.MpsHistManager

- The main class
- Loads the Config DB via MPS Config
- Creates an instance of #edu.stanford.slac.mpshist.MessageExporter
- Tries to create an instance of #edu.stanford.slac.mpshist.MessageServer on ports 1 through 10
- Notifies MPS that an instance of MPS History Server server is up

edu.stanford.slac.mpshist.MpsHistProperties

- Contains (almost) all parameters and settings for the module
    - Including the address of the MPS IOC that sends messages

```
public static final InetSocketAddress SOCKET_ADDRESS = new InetSocketAddress("ioc-bsy0-mp01.slac.
stanford.edu", 30000);
```

edu.stanford.slac.mpshist.ServerNotifier

- A runnable that periodically notifies MPS about a running server
    - Sends out Datagram packets

Release

- Add a note in $MPSHIST_ROOT/RELEASE_NOTES
    - Increment the tag version accordingly
- Commit to CVS
- Tag with mpshist-R#-#-#
- Check out the tagged version on development into your workspace using eco and build using ant

```
$ eco
Enter name of module/package to checkout: mpshist
Enter name of tag or [RETURN] to use HEAD>mpshist-R0-0-26
Using mpshist-R0-0-26. The name of the directory will be mpshist-R0-0-26.
cvs checkout -P -r mpshist-R0-0-26 -d mpshist-R0-0-26 mpshist
cvs checkout: Updating mpshist-R0-0-26
...
$ cd mpshist/mpshist-R0-0-26/
$ ant
...
```

- Push to production and upgrade using cram

Start / Stop

- Only one MPS History Server is active at any time
- To start a new server, you must login as laci on lcls-daemon2

```
/etc/rc3.d/S99st.mpshist start
```

- To stop a production server, type

```
/etc/rc3.d/S99st.mpshist stop
```

    - **Wait 10 seconds!**

Troubleshoot

- The clearest sign of problems is an empty table in the HistoryViewer tab on the MPS GUI
- Every few months, there seem to be weird socket exceptions that don't halt the server (to be investigated)

Log files

- Check /u1/lcls/tools/mpsHistoryServer/mpshist.log

Cmlog

- To check for cmlog messages, specify "MPS History" as facility.

Is the server running?

- To test whether the Server is running, go as laci to lcls-daemon2

```
ps -ef | grep mpshist
```

- Also, check nohup.out in laci's home directory

Is the server working?

- To test whether the server is working, use POSI:EP02:400:MTLM_LGC_BYPV, e.g.

```
caput POSI:EP02:400:MTLM_LGC_BYPV 1
caput POSI:EP02:400:MTLM_LGC_BYPV 0
```

  Verify that there is a message in the MPS GUI (not in cmlog).

Reboot

- See #Starting and Stopping

References

http://java.sun.com/docs/books/tutorial/networking/ (esp. sockets and datagrams)
http://darksleep.com/player/JavaAndUnsignedTypes.html (java types from/to bytes)
MPS Fault Logging Statistic Reports