

Stefan Ritt

MIDAS NEWS

Agenda

- Some news since last workshop at TRIUMF 2017
- Transition to CMake
- Review of new custom pages
- New interactive history plot
- Transition to C++ 11 (part one)

CMake

Building midas

- Traditionally, midas is built with
 - Makefile (UNIX)
 - Xcode project (Mac)
 - VS project (Windows)
- Proposal by Mu3e group in Mainz to move midas to CMake in May 2019
- By now we converted most things to CMake, but keep old Makefile for backup

CMake in a nutshell

CMakeLists.txt

```
add_library(midas STATIC midas.cxx odb.cxx ...)  
target_include_directories(midas PUBLIC ${PROJECT_SOURCE_DIR}/include)  
...
```

```
add_executable(mlogger mlogger.cxx ...)  
target_link_libraries(mlogger midas)
```

target "mlogger" inherits
include directory

```
add_subdirectory(progs)  
add_subdirectory(examples)
```

CMakeLists.txt

```
add_executable(mhttpd ...)
```

CMakeLists.txt

```
add_executable(frontend ...)
```

Things which have changed

- Build midas (library, programs AND examples):
\$ mkdir build; cd build
\$ cmake .. (\$ cmake3 ..)
\$ make
\$ make install
- Changed directory structure:
src/ library sources
progs/ midas programs
build/ temporary build files
lib/ midas library (was linux/lib)
bin/ executables (was linux/bin)
- Install to **/usr/local** not supported any more

Some tips and tricks

- Multi-thread build:

- export MAKEFLAGS=-j\$(`nproc`+1) [Linux]
- export MAKEFLAGS=-j\$(`sysctl -n hw.ncpu`+1) [Mac]
- improves build time by ~5x on 8-core machine

- Build type:

- Default: “RelWithDebInfo” [-O2 -g]
- \$ make CMAKE_BUILD_TYPE=Debug [-g]
- \$ make CMAKE_BUILD_TYPE=Release [-O3]
- Change permanently with “ccmake ..”

Editing CMake cache with “ccmake ..”

```
build -- ccmake .. -- 100x50
Page 1 of 2
CMAKE_AR /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault
CMAKE_BUILD_TYPE RelWithDebInfo
CMAKE_COLOR_MAKEFILE ON
CMAKE_CXX_COMPILER /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault
CMAKE_CXX_FLAGS
CMAKE_CXX_FLAGS_DEBUG -g
CMAKE_CXX_FLAGS_MINSIZEREL -Os -DNDEBUG
CMAKE_CXX_FLAGS_RELEASE -O3 -DNDEBUG
CMAKE_CXX_FLAGS_RELWITHDEBINFO -O2 -g -DNDEBUG
CMAKE_C_COMPILER /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault
CMAKE_C_FLAGS
CMAKE_C_FLAGS_DEBUG -g
CMAKE_C_FLAGS_MINSIZEREL -Os -DNDEBUG
CMAKE_C_FLAGS_RELEASE -O3 -DNDEBUG
CMAKE_C_FLAGS_RELWITHDEBINFO -O2 -g -DNDEBUG
CMAKE_EXECUTABLE_FORMAT MACHO
CMAKE_EXE_LINKER_FLAGS
CMAKE_EXE_LINKER_FLAGS_DEBUG
CMAKE_EXE_LINKER_FLAGS_MINSIZE
CMAKE_EXE_LINKER_FLAGS_RELEASE
CMAKE_EXE_LINKER_FLAGS_RELWITH
CMAKE_EXPORT_COMPILE_COMMANDS OFF
CMAKE_INSTALL_NAME_TOOL /opt/local/bin/install_name_tool
CMAKE_INSTALL_PREFIX /midas
CMAKE_LINKER /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault
CMAKE_MAKE_PROGRAM /usr/bin/make
CMAKE_MODULE_LINKER_FLAGS
CMAKE_MODULE_LINKER_FLAGS_DEBU
CMAKE_MODULE_LINKER_FLAGS_MINS
CMAKE_MODULE_LINKER_FLAGS_RELE
CMAKE_MODULE_LINKER_FLAGS_RELW
CMAKE_NM /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault
CMAKE_OBJCOPY CMAKE_OBJCOPY-NOTFOUND
CMAKE_OBJDUMP /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault
CMAKE_OSX_ARCHITECTURES
CMAKE_OSX_DEPLOYMENT_TARGET
CMAKE_OSX_SYSROOT /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platf
CMAKE_RANLIB /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault
CMAKE_SHARED_LINKER_FLAGS
CMAKE_SHARED_LINKER_FLAGS_DEBU
CMAKE_SHARED_LINKER_FLAGS_MINS
CMAKE_SHARED_LINKER_FLAGS_RELE
CMAKE_SHARED_LINKER_FLAGS_RELW

CMAKE AR: Path to a program.
Press [enter] to edit option Press [d] to delete an entry CMake Version 3.14.5
Press [c] to configure
Press [h] for help Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently On)
```

Building your frontend out-of-tree

```
cmake_minimum_required(VERSION 3.0)
project(frontend)

# Check for MIDASSYS environment variable
if (NOT DEFINED ENV{MIDASSYS})
    message(SEND_ERROR "MIDASSYS environment variable not defined.")
endif()

set(CMAKE_CXX_STANDARD 11)
set(MIDASSYS $ENV{MIDASSYS})

if (${CMAKE_SYSTEM_NAME} MATCHES Linux)
    set(LIBS -lpthread -lutil -lrt)
endif()

add_executable(frontend frontend.cpp)

target_include_directories(frontend PRIVATE ${MIDASSYS}/include)
target_link_libraries(crfe ${MIDASSYS}/lib/libmfe.a ${MIDASSYS}/lib/libmidas.a ${LIBS})
```

Building your frontend in-tree

```
add_subdirectory(your_frontend)
```

```
add_executable(frontend frontend.cpp)  
target_link_libraries(frontend mfe midas ${LIBS})
```

Versioning with other packages

- In complex online environments, you need certain software versions to work with each other
 - online software
 - ROOT
 - perl 5
 - mysql
 - midas
- Versions evolve over time
- To go back in time, you have to manually combine right versions
- This can be automatized with git submodules

MEG software Wiki:

Minimum requirement of ROOT

from 1/Mar/2014	v5-34-17
from 27/Mar/2013	v5-34-00
from 1/Mar/2012	v5-32-00
from 1/Mar/2011	v5-30-00
from 1/Mar/2010	v5-26-00
from 18/Mar/2009	v5-22-00
from 1/Mar/2008	v5-18-00
from 1/May/2007	v5-14-00
from 1/Jan/2007	v5-12-00

Midas as a submodule

```
$ git submodule add git@butbucket.org:tmidas/midas
```

```
<your experiment>  
  /online  
  /analyzer  
  /root  
  /midas → fe0aa8d438a25157fb6
```

- Make sure your project compiles with current midas
- Committing your project commits a certain midas revision
- Checking out a specific review gets automatically the compatible midas version
- Compile recursively with CMake

```
$ git checkout 3ef0d3a  
$ git pull --recurse-submodules
```

CMakeLists.txt

```
add_subdirectory(midas)
```

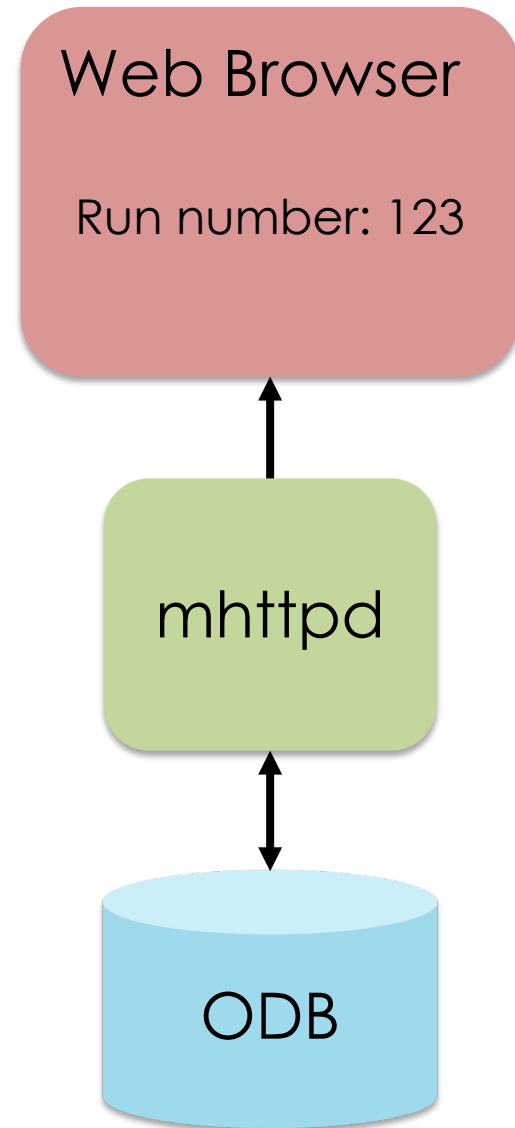
Custom Pages

Old Custom pages (reminder)

```
<html>  
Run number: <odb path="/Runinfo/Run number">  
</html>
```

```
db_get_value("/Runinfo/Run number")
```

```
/Runinfo/Run number: 123
```



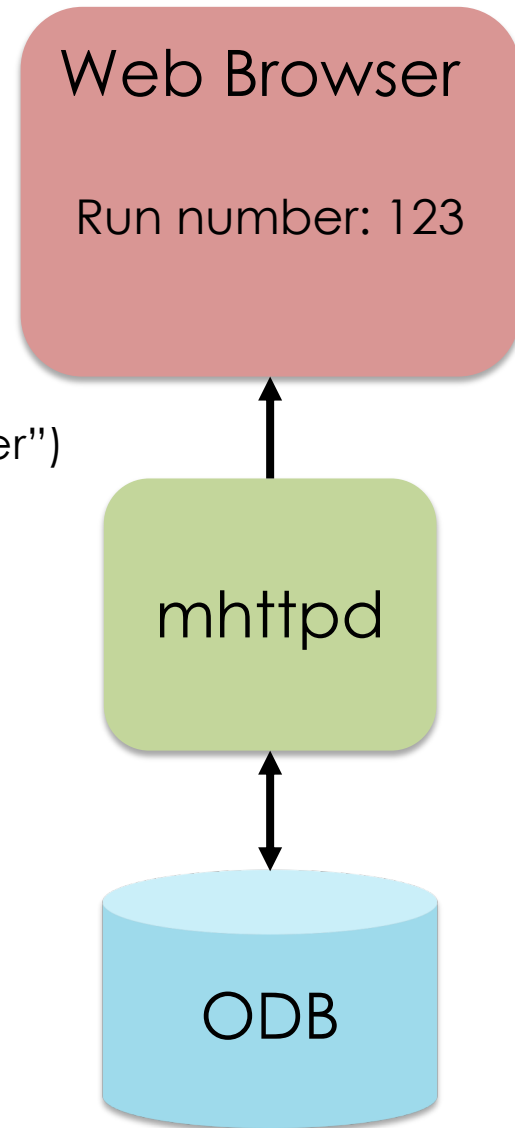
Old Custom pages (reminder)

```
mjsonrpc_db_get_value(["/Runinfo/Run  
number"]).then(function(rpc) {  
  document.write("Run number: " + rpc.result.data[0]);  
})
```

```
mjsonrpc_db_get_value("/Runinfo/Run number")
```

```
db_get_value("/Runinfo/Run number")
```

```
/Runinfo/Run number: 123
```



New Custom Pages

- Pages are independent of mhttpd:
Changes do not require to re-compile mhttpd
- mhttpd becomes a simple file web server and RPC server to access midas functions from web pages
- Use full potential of HTML5: interactive web pages à la Google Mail or Google Maps
- “Standard” mhttpd pages can be customized easily

Standard Status Page

☰ Online Alarms: None 6 Aug 2019, 10:00:06 GMT-7

Status
Transition
ODB
Messages
Chat
Elog
Alarms
Programs
Buffers
History
MSCB
Sequencer
Config
Help
JSHistory

Run Status

Run 5 Running <input type="button" value="Stop"/>	Start: Mon Aug 5 10:04:33 2019	Running time: 23h55m33s
<input type="button" value="Alarms: On"/>	<input type="button" value="Restart: Off"/>	Data dir: /online/

1565110375 09:52:55.814 2019/08/06 [ODBEEdit,INFO] Program ODBEdit on host localhost started

Equipment

Equipment +	Status	Events	Events[/s]	Data[MB/s]
Trigger	Sample Frontend@localhost	5.146M	89.1	0.005
Periodic	Sample Frontend@localhost	52473	1.0	0.000

Logging Channels

Channel	Events	MB written	Compr.	Disk Level
#0: run00005.mid.lz4	5198353	131.103	44.7%	87.7%
Lazy Label	Progress	File Name	# Files	Total

Clients

mhttpd [localhost] ODBEEdit [localhost]	Sample Frontend [localhost]	Logger [localhost]
--	-----------------------------	--------------------

Modified Status Page

Online Alarms: None 6 Aug 2019, 9:59:47 GMT-7

Status
Transition
ODB
Messages
Chat
Elog
Alarms
Programs
Buffers
History
MSCB
Sequencer
Config
Help
JSHistory

Run Status

Run 5	Start: Mon Aug 5 10:04:33 2019	Running time: 23h55m15s
Running <input type="button" value="Stop"/>	Alarms: On	Restart: Off
Data dir: /online/		

1565110375 09:52:55.814 2019/08/06 [ODBEdit,INFO] Program ODBEdit on host localhost started

Trigger rate:88.5

Equipment

Equipment +	Status	Events	Events[/s]	Data[MB/s]
Trigger	Sample Frontend@localhost	5.144M	88.5	0.005
Periodic	Sample Frontend@localhost	52456	1.0	0.000

Logging Channels

Channel	Events	MB written	Compr.	Disk Level
#0: run00005.mid.lz4	5196637	131.103	44.7%	87.7%

Lazy Label	Progress	File Name	# Files	Total
------------	----------	-----------	---------	-------

Clients

```
<tr>  
<td colspan="6" align="center">  
  Trigger rate:<span name="modbvalue" data-odb-path="/Equipment/Trigger/Statistics/Events per sec." data-format="f1"></span>  
</td>  
</tr>
```

mjsonrpc_xxx functions

≡ Online Alarms: None 6 Aug 2019, 10:04:50 GMT-7

Status
Transition
ODB
Messages
Chat
Elog
Alarms
Programs
Buffers
History
MSCB
Sequencer
Config
Help
JSHistory

MIDAS Javascript functions

This web page shows examples of how to use most MIDAS javascript and JSON-RPC functions. For more information, please read the MIDAS documentation: <https://midas.triumf.ca/MidasWiki/index.php/Mhttpd.js>

Index

```
mjsonrpc_set_url(url);
mjsonrpc_db_get_values(paths);
mjsonrpc_db_copy(paths);
mjsonrpc_db_paste(paths, values);
mjsonrpc_db_ls(paths);
mjsonrpc_db_create(specs);
mjsonrpc_db_resize(paths, new_lengths);
test db_resize_string(paths, new_lengths, new_string_lengths);
mjsonrpc_db_key(paths);
test db_rename
test db_link
test db_reorder
mjsonrpc_db_delete(paths);
mjsonrpc_cm_msg(message);
test cm_msg_facilities();
test cm_msg_retrieve(facility, time, min_messages);
test cm_exist()
test cm_shutdown()
test start_program()
test al_trigger_alarm()
test al_reset_alarm()
test get_alarms()
test jrpc()
ODBEdit(path);
```

modb* JavaScript Scheme

```
mjsonrpc_db_get_value(["/Runinfo/Run number"]).then(function(rpc) {  
    document.getElementById('run_number').innerHTML = rpc.result.data[0];  
}).catch(function(error) {  
    mjsonrpc_error_alert(error);  
});
```

```
<body class="mclass" onload="mhttpd_init('myPage');>
```

...

```
<div name="modbvalue" data-odb-path="/Runinfo/Run number"></div>
```


Complete example (reminder)

```
<body class="mcss" onload="mhttpd_init('Test',  
1000)">
```

```
<div id="mheader"></div>
```

```
<div id="msidenav"></div>
```

```
<div id="mmain">
```

```
  <table class="mtable">
```

```
    <tr>
```

```
      <th colspan="2"  
class="mtableheader">Status</th>
```

```
    </tr>
```

```
    <tr>
```

```
      <td style="width: 200px">
```

```
        Run number:
```

```
      </td>
```

```
      <td>
```

```
        <div name="modbvalue" data-odb-  
path="/Runinfo/Run number" data-odb-  
editable="1"></div>
```

```
      </td>
```

```
    </tr>
```

```
      <div name="modbbar" style="width:  
500px" data-odb-path="/Runinfo/Run number"  
data-max-value="10"
```

```
        data-color="lightgreen"></div>
```

```
    </td>
```

```
  </tr>
```

```
  <tr>
```

```
    <td>
```

```
      <button name="modbbutton"  
class="mbutton" data-odb-path="/Runinfo/Run  
number" data-odb-value="1">Set run  
        number to 1
```

```
      </button>
```

```
    </td>
```

```
  </tr>
```

```
  </table>
```

```
</div>
```

```
</body>
```

```
</html>
```

Resulting HTML page

The screenshot displays a web interface with a top navigation bar and a main content area. The top bar includes a hamburger menu icon, the text 'online', the date and time 'Tue Jul 25 2017 14:14:43 GMT-0700 (PDT)', and a 'Help' link. A vertical sidebar on the left contains a 'Standard menu' with items: Status, Start, Transition, ODB, Messages, Chat, Alarms, Programs, History, MSCB, Sequencer, Config, Example, and Help. The main content area features a 'Status' panel with a blue header. Below the header, it displays: 'Run number: 5', 'Last run start: Mon Jul 17 15:53:12 2017', 'Last run stop: Mon Jul 17 15:53:23 2017', and 'Indicator: 5' with a green progress bar. At the bottom of the status panel are three buttons: 'Set run number to 1', 'Set run number to 5', and 'Set run number to 10'. Two blue arrows point to the top bar and the sidebar, labeled 'Messages and alarms' and 'Standard menu' respectively.

Messages and alarms

online Tue Jul 25 2017 14:14:43 GMT-0700 (PDT) Help

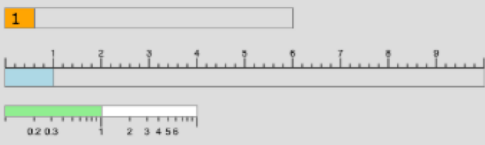
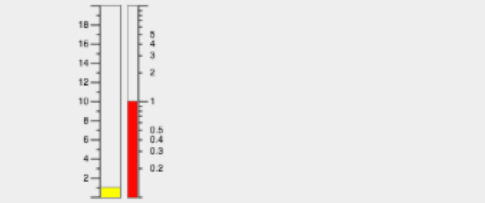
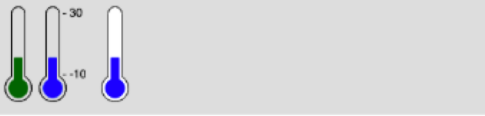
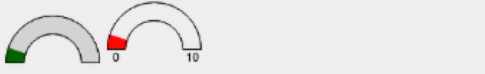
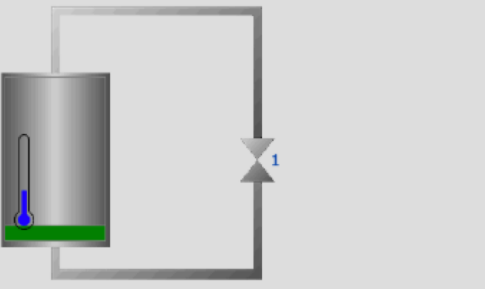
Status

Run number:	5
Last run start:	Mon Jul 17 15:53:12 2017
Last run stop:	Mon Jul 17 15:53:23 2017
Indicator:	5

Set run number to 1 Set run number to 5 Set run number to 10

Standard menu

Status

Run number:	1
Last run start:	Thu Feb 7 10:35:31 2019
Last run stop:	Thu Feb 7 10:40:36 2019
Check box:	<input checked="" type="checkbox"/>
Color box:	<input type="checkbox"/>
Horizontal bars:	
Vertical bars:	
Thermometer:	
Gauges:	
	
<input type="button" value="Set run number to 1"/> <input type="button" value="Set run number to 5"/> <input type="button" value="Set run number to 10"/>	

modbvalue

modbcheckbox
modbbox

modbhbar + mhaxis

modbvbar + mvaxis

modbthermo

modbgauge

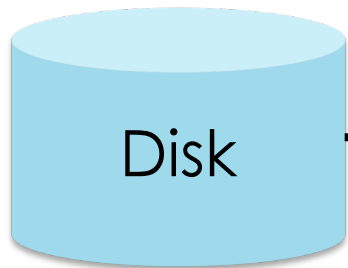
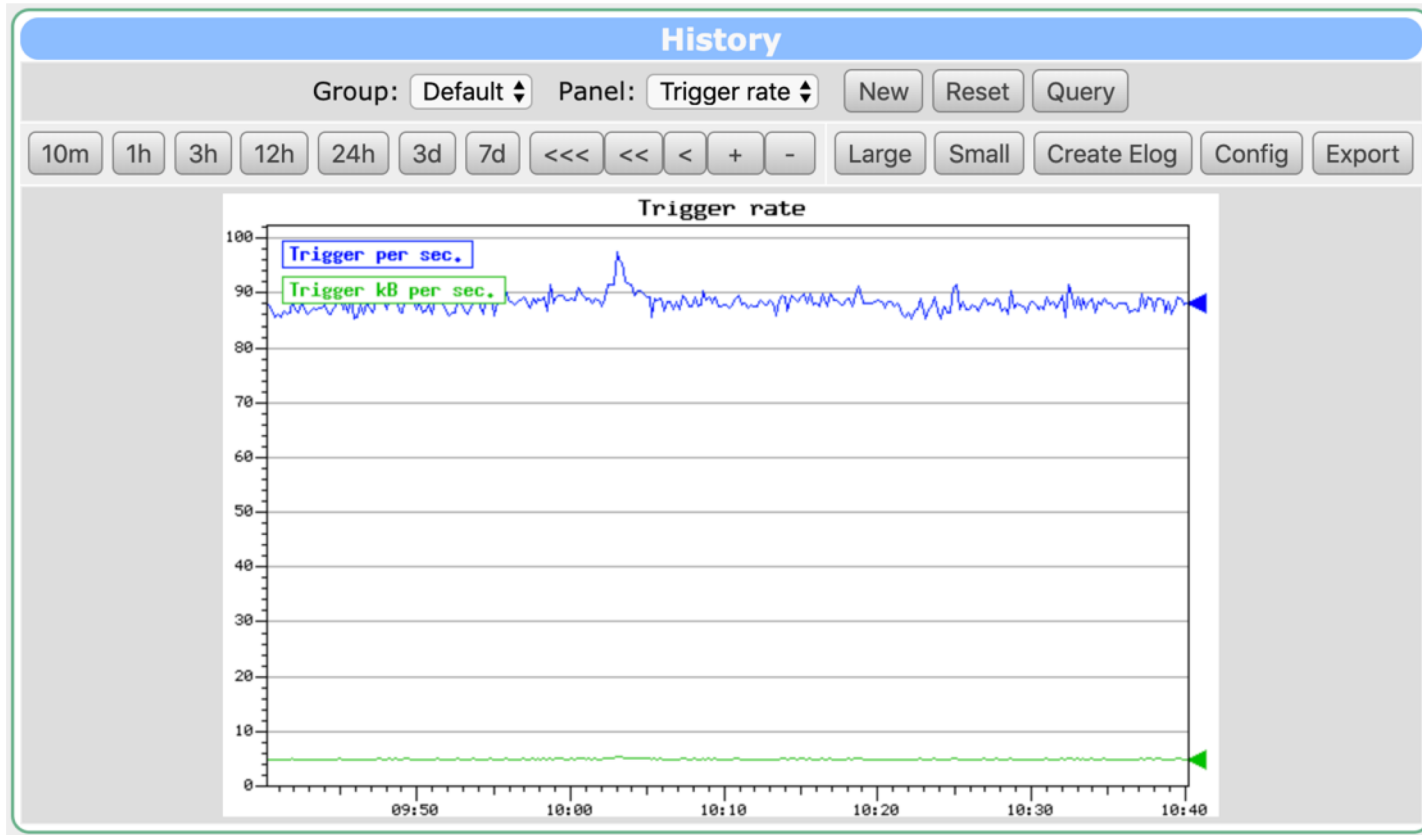
modbthermo + modbvbar
on top of image

modbbutton

Conversion of standard midas pages

Page	Status
Status	OK (Shouyi Ma)
Transition	OK
ODB	TBD
Messages	OK
Chat	OK
Elog	OK
Alarms	OK
Programs	OK
Buffers	New (KO)
History	WIP
Sequencer	TBD
Config	OK
Help	TBD

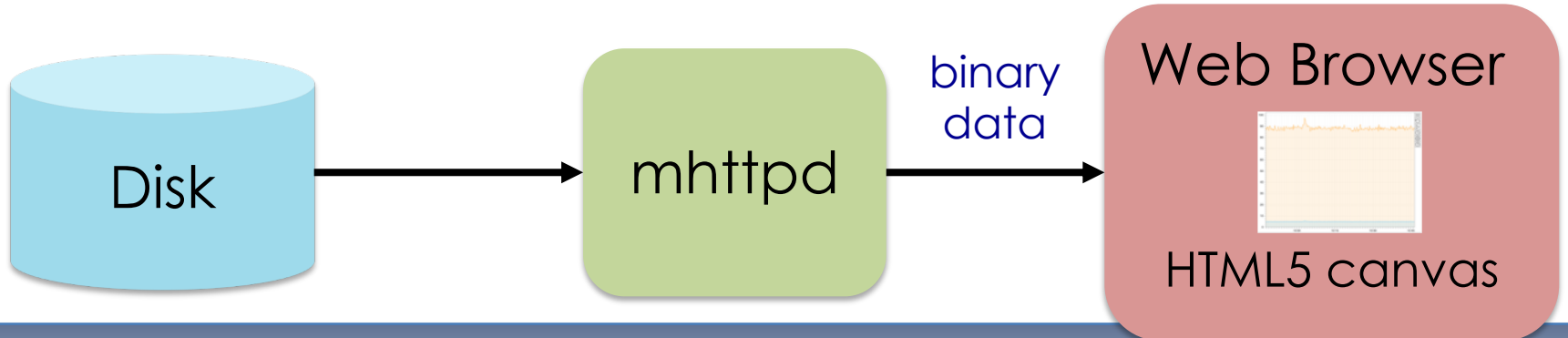
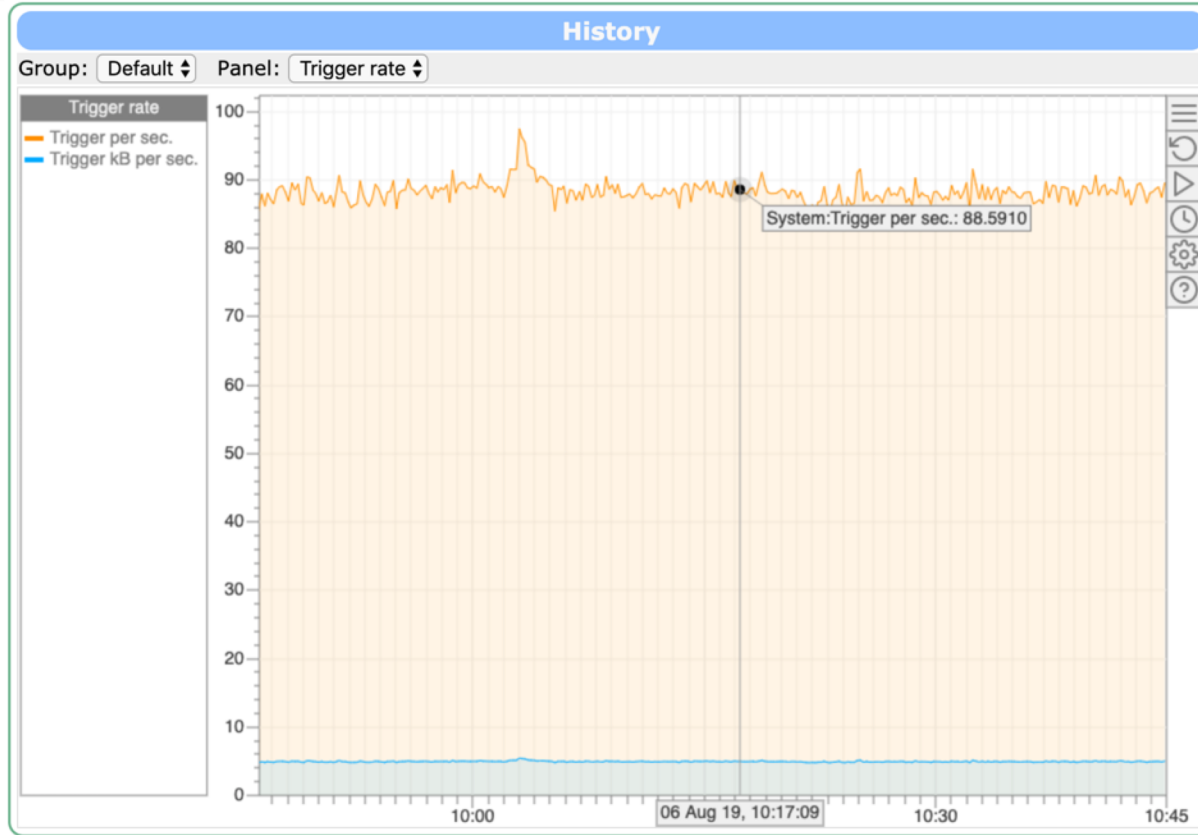
Old History



static GIF
image



New History



Interactivity

- Live scrolling
- Panning and zooming (à la Google Maps)
- Inspect values with cursor
- Dynamic loading of data
 - Update only pulls new data points since last update
 - Un-zooming pulls old data on request
 - Switching to binary data (JS typed arrays) for speed improvements
- Custom page:

```
<div data-name="mjshistory"  
  data-group="Default"  
  data-panel="Trigger rate"></div>
```

Midas and C++

Switch to C++

- Around June 2019
midas has been switched to C++
- Should we settle on C++ 11 ?
- Some minor changes in user code:
<https://midas.triumf.ca/elog/Midas/1526>
- Converted most device drivers, few are missing (mainly type casts)
- This offers new possibilities → next talk
- This talk: C++ interface to ODB

Long standing problem

ODB

```
/Equipment/Trigger/Common
Event ID      1
Trigger mask  0
Buffer        SYSTEM
Source        0
Format        MIDAS
...
```

C program

```
typedef struct {
    int event_id;
    int trigger mask;
    char buffer[32];
    int source;
    char format[32];
    ...
} COMMON;

COMMON c;
c.event_id = 1;
```

```
[/Equipment/Trigger/Common]
Event ID = WORD : 1
Trigger mask = WORD : 0
Buffer = STRING : [32] SYSTEM
Type = INT : 2
Source = INT : 0
Format = STRING : [8] MIDAS
Enabled = BOOL : y
```

- Need ASCII string to create ODB entries
- Use `db_get_record()` to link C structure to ODB
- Issues: order of variables, size of strings, ...

New Idea

- Inspired by Niels Lohmann: JSON for modern C++
<https://github.com/nlohmann/json>

```
#include <nlohmann/json.hpp>

using json = nlohmann::json;

int main()
{
    json j = {
        { "pi", 3.141 },
        { "happy", true },
        { "answer", {
            { "everything", 42 }
        }},
        { "name", "midas" },
        { "list", {1, 0, 2} }
    };

    j["pi"] = 3.14156;
    std::cout << j["pi"] << std::endl;
}
```

- Just one header file
- Vanilla C++11
- JSON object almost feel like native C++ literal
- Use JSON like an STL container
- Automatic type deduction

New C++ ODB API "mdata"

```
#include <mdata.hpp>

using mdata = midas::mdata;

int main()
{
    cm_connect_experiment(...);

    mdata d("/Equipment/Trigger/Common");

    std::cout << d["Format"] << std::endl;

    d["Event ID"] = 1;

    d["Format"].addListener([](mdata &d) {
        std::cout << d << std::endl;
    });
}
```

"d" contains a local copy of ODB subtree

send data to ODB

hot-links via lambda

Some additional thoughts

- Do not break current C API, just add new one on top
`#include <mdata>`
- Let compiler deduce data types
`d["value1"] = 1.23; // double TID_DOULBE`
`d["value2"] = 1.23f; // float TID_FLOAT`
- Nested and array data
`mdata d("/Equipment/Trigger");`
`d["Common"]["Event ID"] = 1;`
`std::string s = d["Settings"]["Names"][15];`

Some additional thoughts

- Bunch write operations

```
d.startTransaction();  
d["a"] = 1;  
d["n"] = "hello";  
d.commitTransaction();
```

- Serialization to JSON files

```
ofstream f;  
f.open("example.json");  
f << d << endl;  
f.close();
```

- Access ODB dumps and databases

```
mdata d("file:run123.mid");  
mdata d("couchdb:midas.triumf.ca/db");
```

- Any other idea?