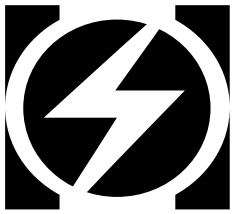
OSHMI
Open Substation HMI



Configuration Manual

Version 6.12

Contents

HMI Features

"Full Graphics" HMI (vector based), i.e. with lossless zoom.

Operates in redundant mode (primary/secondary).

Quick and easy configuration and development of screens.

IEC60870-5-104, DNP3, MODBUS, OPC UA/DA, S7 and ICCP protocol drivers.

Uses standardized Web technologies (HTML, Javascript, SVG), plus popular open source tools like SQLite, NGINX HTTP Server, PHP and LUA.

Powerful Inkscape+SAGE SVG open source graphics editor.

Runs on Windows and Linux PC platform (through Wine), in multiple browsers: Chromium (default), Chrome, Firefox and Safari. Can be accessed by tablets and smartphones (IOS and Android).

Screen Viewer: zoom and pan functions, time machine.

Event Viewer: event list with millisecond resolution, panic and history functions.

Tabular Viewer: visualization of points available per substation/bay.

Alarms Viewer: displays alarms, with filters by substation and priority.

Trend Viewer: shows a real time analog measurement plot.

Curves Viewer: queries the historical archives and plot charts.

Recording of historical data. PostgreSQL and Grafana available as add-ons.

Specialized treatment for state points, alarms, and events.

Bounds checking of measures (maximum and minimum with hysteresis).

Inhibition of alarms. Acknowledgment in 2 stages (acknowledged and eliminated).

Support for annotations (safety card) with command block.

Dedicated Shell, replaces the standard Windows Shell, allows access only to the functions of the HMI.

Lua Scripting language for the server environment. Javascript language for screen scripts.

Unlimited number of viewers, and can be opened on the local machine (where the server wheel) or on remote machines. No limit on machines and monitors.

No arbitrary limit on the number of points.

Integration with Excel (dynamic worksheet), PowerBI and Tableau.

Operator Training Features:

- Real time data simulator: allows to simulate measures, digital states and commands.
- Historical archives player: allows you to simulate (reconstitute) situations already occurred and stored.

Requirements

Minimum Hardware:

- IBM/PC x86 compatible computer.
- Intel Core i3 processor, 4GB RAM, HD 240GB.
- Ethernet 100MBPS.
- 22” LCD monitor with 1920x1080 resolution.

Strongly Recommended:

- Intel Core i7 (or other CPU w/ 4 or more cores), 8GB RAM, 240GB SSD.
- Video board with 2 independent monitor outputs.
- 2 x 27” 2560x1440 or Ultra HD LED monitors.

Recommended Operating Systems:

- Windows 10 Pro/Home/Enterprise 32/64 bits.
- Linux Mint and others (requires manual setup).

Deprecated Operating Systems:

- Windows 7 32/64 bits. Still compatible, but not recommended for new projects.
- Windows 8/8.1 32/64 bits. Still compatible, but not recommended for new projects.

Not anymore compatible Operating Systems:

- Windows XP and Vista. Those are possible only up to OSHMI version 3.17 and Chromium 50.

Other utility software:

- VS Code, Notepad++, PSPAD, Geany or other text editor.

Directory Structure

\oshmi\bin – executable files.

webserver.exe – realtime data webserver

hmishell.exe – replacement for the Windows Shell to lock an HMI machine.

mon_proc.exe – process monitor (software watchdog).

menu.bat – special options menu.

QTester104.exe – IEC60870-5-104 client protocol driver.

iccp_client.exe – ICCP client protocol driver.

opc_client.exe – OPC UA/DA client driver.

s7client.exe – Siemens S7 client protocol driver.

modbus.exe – Modbus/TCP client driver.

dnp3.exe – DNP3 client driver.

*.dll – necessary libraries.

\oshmi\conf – config files (files here are not overwritten on updates).

oshmi_config_manager.xlsm - Excel worksheet for configuration management.

hmi.ini – hmi main config file.

point_list.txt – point list file.

point_calc.txt – point calculations file.

mon_proc.ini – config file for mon_proc.

hmishell.ini – configuration file for HMIShell.

qtester104.ini – configuration of QTester104.

opc_client.conf – configuration of OPC Client.

s7client.conf – configuration of S7 Client.

dnp3.ini – configuration of DNP3 driver.

modbus_queue.ini – configuration of Modbus driver.

iccp_config.txt – configuration of ICCP driver.

config_viewers.js – configuration of Viewers.

nginx_access_control.conf – configure allowed client machines.

nginx_http.conf – configure http access.

nginx_httpd.conf – configure https secure access and client certificates.

\oshmi\conf_templates – config files templates (overwritten by OSHMI updates).

\oshmi\i18n – Internationalization files.

messages_i18n.js – messages for the web interface

hmishell_i18n.ini – messages for HMIShell.

\oshmi\htdocs – HTML, javscript and PHP files.

screen.html e websage.js – Screen Viewer.

dlginfo.html e dlgcomando.html – Point and command access dialogs.

events.html – Events Viewer (realtime and historic).

tabular.html – Tabular Viewer.

trend.html – Trend Viewer.

\oshmi\svg – Screen files (files here are not overwritten on updates).

screen_list.js – list of available screens (Screen Viewer).

*.svg – SVG screens.

\oshmi\db – SQLite database files.

soe.sl3 – events table (not overwritten on updates).

dumpdb.sl3 – point state dump table (not overwritten on updates).

hist.sl3 – historic data table (not overwritten on updates).

notes.sl3 – database table for non-blocking annotations.

db_maintenance.bat – script for database files checking and optimizing.

\db_cold*.* - cold (empty) database files and SQL build scripts.

\oshmi\browser and \oshmi\browser-data – Chromium browser files.

\oshmi\inkscape – Inkscape SVG display editor.

\oshmi\docs – Documentation files.

\oshmi\etc – Registry files, dynamic Excel sheet, Services control and other auxiliary files.

\oshmi\charts – Vega charts.

\oshmi\extprogs – Externally download-able software.

\oshmi\fonts – Font files.

\oshmi\logs – Log files (not overwritten on updates).

\oshmi\nginx_php – NGinx and PHP files.

\oshmi\linux – Linux specific scripts.

\oshmi\scripts – Lua scripts (not overwritten on updates).

Installation on Windows

1. Install *Windows* OS (7, 8 or 10 – 32 or 64 bits - Home/Pro/Business).
2. Login as an Windows administrator user.
3. Deactivate *Windows Firewall* (recommended, not mandatory).
   Deactivate Automatic Updates (recommended, not mandatory).
   Uninstall all unnecessary software, specially manufacturer's bloat-ware.
   Configure the system for maximum performance.
   Deactivate screen saver.
   Deactivate monitor suspend function.
   Deactivate all energy economy functions.
4. If possible, disable all anti-virus software that can slow down system performance (Windows embedded tools like Defender are in general lightweight and do not impact system performance).
5. Deactivate *UAC (recommended, not mandatory)*.
   Deactivate the following services *(recommended, not mandatory)*:
   - Windows Search;
   - Ready Boost;
   - Super Fetch.
6. Windows 10: disable all privacy options, Cortana and background app processing (recommended, not mandatory).
7. Install OSHMI executing the file ***oshmi_setup_xxx.exe***.
8. Optional: execute ***c:\oshmi\extprogs\download_external_progs.bat***.
9. Edit the point configuration file ***point_list.txt*** and the calculations file (***point_calc.txt***), if necessary.
10. Configure the file ***c:\oshmi\conf\hmi.ini.***
    Configure one of the protocol drivers:
    c:\oshmi\conf\qtester104.ini
    c:\oshmi\conf\iccp_config.txt
    c:\oshmi\conf\dnp3.ini
    c:\oshmi\conf\modbus_queue.ini
11. Screens must be put in ***c:\oshmi\svg*** and listed in ***c:\oshmi\svg\screen_list.js***.
12. Configure process that will run (at least ***webserver.exe*** and one protocol driver) in ***mon_proc.ini*** or to run as Windows services in ***c:\oshmi\etc*** batch files.

The “hmi.ini” file

```
; HMI.INI OSHMI CONFIGURATION FILE

;--------------------------------------------------------------------------------------
; Initialization config
[RUN]

; Delay in seconds to launch apps
; Default = 15
;DELAY=15

; Interval in seconds to launch apps
; Default = 5 ;
;INTERVAL=5

; Protocol driver executable file
; default = "qtester104.exe"
;PROTOCOL_DRIVER="qtester104.exe"

; Apps to run at initialization
; default = empty
;RUN00="browser\chrome.exe" --user-data-dir=c:\oshmi\browser-data --disable-popup-blocking --process-per-
site --no-sandbox --no-proxy-server
--app=http://127.0.0.1:51909/htdocs/screen.html?SELTELA=svg/tela_cin.svg
;RUN01="browser\chrome.exe" --user-data-dir=c:\oshmi\browser-data --disable-popup-blocking --process-per-
site --no-sandbox --no-proxy-server --app=http://127.0.0.1:51909/htdocs/events.html

; Hides or not the webserver.exe window, 1=hide, 0=don't hide,
; Default=0.
HIDE=0

; Simulation mode (0=disable simulation, 1=enable local simulation, 2=simulation in master mode,
3=simulation in slave mode)
; Default = 0
SIMULATION=1

; Use internal beep speaker (1=yes, 0=no).
; Default = 0
;BEEP_INTSPEAKER=0

; Viewers command line (updated by the installer)
; Normally only needs to be edited to run on Linux or for changing the browser on Windows.
EVENTS_VIEWER="c:\oshmi\browser\chrome.exe --user-data-dir=c:\oshmi\browser-data --bopt
--app=http://127.0.0.1:51909/htdocs/events.html"
TABULAR_VIEWER="c:\oshmi\browser\chrome --user-data-dir=c:\oshmi\browser-data --bopt
--app=http://127.0.0.1:51909/htdocs/tabular.html"
SCREEN_VIEWER="c:\oshmi\browser\chrome --user-data-dir=c:\oshmi\browser-data --bopt
--app=http://127.0.0.1:51909/htdocs/screen.html"
TREND_VIEWER="c:\oshmi\browser\chrome --user-data-dir=c:\oshmi\browser-data --bopt
--app=http://127.0.0.1:51909/htdocs/trend.html"
CURVES_VIEWER="c:\oshmi\browser\chrome --user-data-dir=c:\oshmi\browser-data --bopt
--app=http://127.0.0.1:51909/htdocs/histwebview/histwebview.php"
DOCS_VIEWER="c:\oshmi\browser\chrome --user-data-dir=c:\oshmi\browser-data --bopt
--app=http://127.0.0.1:51909/htdocs/listdocs.php"
LOGS_VIEWER="c:\oshmi\browser\chrome --user-data-dir=c:\oshmi\browser-data --bopt
--app=http://127.0.0.1:51909/htdocs/listlogs.php"

; Browser command line keys (not updated by the installer)
; Recommended for low memory systems: "--process-per-site --no-sandbox --disable-popup-blocking --no-proxy-
server --bwsi --disable-extensions --disable-sync --no-first-run"
; Recommended for high performance systems: "--disable-popup-blocking --no-proxy-server --bwsi --disable-
extensions --disable-sync --no-first-run"
; Default (low memory)= "--process-per-site --no-sandbox --disable-popup-blocking --no-proxy-server --bwsi
--disable-extensions --disable-sync --no-first-run"
;BROWSER_OPTIONS="--process-per-site --no-sandbox --disable-popup-blocking --no-proxy-server --bwsi --
disable-extensions --disable-sync --no-first-run"

;--------------------------------------------------------------------------------------
; Redundancy configuration
[REDUNDANCY]

; IP Address of the other machine
; Default = empty (no redundancy)
;OTHER_HMI_IP=192.168.1.2

; Use the hour of the other machine to set local clock (1=yes, 0=no)
; Default = 0
;ACCEPT_TIME=0

; Interval in seconds to send keep alive to the other machine (default = 5)
;SEND_TIME_PERIOD=5

; Port for event and annotation synchronization
; Default: 51909
;HTTP_PORT=51909
```

```
;-------------------------------------------------------------------------------------------
[WEBSERVER]
; Max number of events shown in Events Viewer
; Default = 150
;MAX_EVENTS=150

; Priority up to that is shown for events on panic mode (Events Viewer)
; Default = 3
;PANIC_PRIORITY=3

; Enable double transition alarm (1=alarm, 0=no alarm)
; Default = 0
;DOUBLE_TRANSITION_ALARM=0

; Remote IP Address list, comma separated, * = everyone can access.
; Default = empty (normally it's empty)
; This is valid only for webserver.exe and do not affect NGINX.
;REMOTE_CLIENTS=""

; Adjust Timezone for PHP
; See list of supported timezones: http://www.php.net/manual/en/timezones.php
;TIMEZONE="America/Recife"

; MicroHTTPd server daemon start mode (1=single thread, 2=thread per connection)
; Default = 1
;HTTPD_MODE=1

; Origin for the http CORS header Access-Control-Allow-Origin to responses (use empty string to remove
header)
; Default = "*"
;CORS_ORIGIN="*"

;-------------------------------------------------------------------------------------------
[HIST]
; Do record historic data (SQLITE): 1=yes, 0=no
; Default = 1
;RECORD=1

; Dead band scale: 100 = normal (100%), 200 = doubles dead band = less data will be recorded
; Default = 100
;DEADBAND_FACTOR=100 ;

; Number of days to keep events and historical data
; Default=36
;LIFETIME=36

; Enable SQL text files generation to feed a PostgreSQL database.
; 1 = Enable, 0 = Disable
; Default = 0
;DB_POSTGRESQL=0

;-------------------------------------------------------------------------------------------
[JSON]
; JSON UDP Protocol driver interface configuration

; UDP port for data update (where OSHMI listens for JSON UDP messages), Default=9100 Disable=0
;UDP_JSON_PORT=9100
; UDP port destination for OSHMI UDP JSON command messages, Default=9101 Disable=0
;UDP_JSON_PORT_CMD=9101

; IP address for each endpoint devices
; Default endpoints are the local address (127.0.0.1) and the specified OTHER_HMI_IP
;UDP_JSON_ENDPOINT1=127.0.0.1
;UDP_JSON_ENDPOINT2=192.168.1.2
```

The “point_list.txt” File

This text file configures the list of points (TAGS) that will be used in the system. It must be encoded UTF-8 or pure ASCII. Spaces are the field separator, so there must be no spaces inside each field (except for the last field that is quoted). This file can be managed and updated also using the Excel worksheet “oshmi_config_manager.xlsm” that is self-documented and easier to edit.

Columns description:

POINT_NUMB = point number (integer, 32 bits), must be unique.

ADDR = protocol address (integer 24 bits, 3 octets), the value zero means ADDR=POINT_NUMB. Must be unique for the same RTU.

ID = point ID (tag) (up to 29 chars), must be unique.

TYP = Point type “D”: digital or “A”: analog (1 char).

MESSAGE = state descriptors (OFF/ON) (separated by a mandatory “/”, up to 29 chars total, up to 24 by state). Unity of measure for analog points (up to 10 chars). SPACES NOT ALLOWED HERE.

ALM = code of the MESSAGE, only for documentary purposes (integer, 3 digits).

EQ = type of equipment state or measurement code (integer, 3 digits).

INF = complimentary info code (integer, 3 digits).

Special types (EQ/INF):

Information type	*EQ*	*INF*
Unspecified type	0	0
kV	1	0
A	2	0
MW	3	0
MVAr	4	0
MVA	8	0
Hz	9	0
Transformer Tap	16	0
Temperature	19	0
Breaker State	27	0
Switch State	28	0
Fault Distance (km)	32	8

OR = origin code (0=normal, 1=calculated, 6=manual, 7=command) (integer, 1 digit).

UN = Digitals and Commands: not used (must be 1 character, can be a “-” dash).

For Modbus commands: “M” = change the driver UDP listen port to 8097.

For analog points:

“t” = delay for alarm of 30s.

“T” = delay for alarm of 60s.

“u” = delay for alarm of 5min.

“U” = delay for alarm of 10min.

Other characters = no delay on alarms.

RTU = RTU protocol link address (integer, 3 digits).

ASDU = protocol ASDU type code (integer, 3 digits).

KCONV1 = conversion factor 1.

For analog points: A factor from “A.x + B” conversion.

For digital points, “-1” inverts state.

For commands:

1=with select, 0=no select.

KCONV2 = conversion factor 2.

For analog points: B factor from “A.x + B” conversion.

For digital points, “-1” create a time stamp when change detected.

For commands, duration: 0=unspecified, 1=short, 2=long, 3=persistent.

SUPCMD = supervised point number relative to a command point (integer) (each command must have a corresponding supervised point). When the point is not a command (origin != 7), this field must be “0” (zero).

DC = Decimals (integer, 2 digits)

Number of decimal places for analog measures (0=no, -1=1 place, -2=2 places or -3=3 places), only for documentary purposes.

Number of decimal places for analog events (10=no, 11=1 place, 12=2 places or 13=3 places).

For digital points, it's the alarm state:

0 = alarmed only on state OFF, e.g: communication state health.

1 = alarmed only on state ON, e.g: equipment failure.

2 = alarmed both when changed to states ON and OFF, e.g: switch breaker state.

3 = pure event, only matters the transition from OFF to ON, e.g: protection operation.

PR = alarm priority (integer, 1 digit), 0=most urgent, 9=least urgent.

INIVAL = an initial value for the point. For commands it is a point number to a interlocking point (OFF=liberated, ON=interlocked) that can block a command, a minus signal inverts the interlocking logic.

“SUBST~BAY~DESCRIPTION” = substation name, bay and description of point (up to 29 ~ 29 ~ 79 characters, max of 99 in total);

WARNING: The space character (ASCII 32) is the field separator so it can't be used inside fields except for the last that is delimited by (double) quotation marks.

The “point_calc.txt” file

Allows to calculate points based on predefined compiled formulas.

The calculated point and all the parcels must be present also in the “point_list.txt” file.

Fields:

PTNUM = number of the calculated point

PARC = point number of the parcel point

COD_FORM = numeric code of the formula

FORMULA = optional documentary formula name

ID_POINT = optional documentary ID (tag) of the calculated point

ID_PARC = optional documentary ID (tag) of the parcel point

Example:

```
SUBX-SUBY-MVA [5050] =
       APPARENT_POWER ( SUBX-SUBY-MW [1855] ,  SUBX-SUBY-MVAR [1856] )
```

Contents of “point_calc.txt”:

```
PTNUM PARCEL   COD_FORM  FORMULA  ID_POINT      ID_PARC
05050  01855        03         PA   SUBX-SUBY-MVA SUBX-SUBY-MW
05050  01856        03         PA   SUBX-SUBY-MVA SUBX-SUBY-MVAR
```

Available formulas:

```
COD_FORM    FORMULA PARCELS            DESCRIPTION
01          CURRENT     3              ((sqrt(((P1*P1)+(P2*P2))))/P3)*1000/sqrt(3)
03          AP          2              sqrt((P1*P1)+(P2*P2))
09          API         2              (P1*P2*sqrt(3))/1000
04          SUM         n              P1+P2+...+Pn
06          AND         n              P1 and P2 and … Pn (“and” digital)
07          OR          n              P1 or P2 or … Pn (“or” digital)
10          NEGSUM      n              -P1-P2...-Pn
15          DIF         2              P1-P2
50          CHOOSED     2              Best of 2 digital points
51          CHOOSEA     2              Best of 2 analog points
```

Obs.: for digital calculations, use kconv2=-1 in “point_list.txt” to generate a events.

WARNING: The space character (ASCIII 32) is the field separator so it can't be used inside fields.

The screen_list.js file

This file lists the available SVG screens to the operator in the Screen Viewer.

This is a *Javascript* file with the grouped list of screens marked up as HTML attributed to a *Javascript* variable `optionhtml`.

You must edit the markup to list your screens.

The option value must be the file name in the svg folder in this form: '*svg/scren_file1.svg*'.

The text after the <*option*> tag is the text displayed in the screen menu.

The <*optgroup*> tag can be used to group screens. The label parameter describes the group.

There are keyboard shortcuts to screens of 2 types:

- Numeric shortcuts 1,2,3,4,5,6,7,8,9 and 0: these are fixed shortcuts for the first 10 screens. They can be shown in the description text between the “[“ and “]” characters as [1] for the first screen and [0] for the tenth screen.
- Alphabetic shortcuts (A-Z): can be attributed to any screen marking the description text with the chosen key shortcut between the “{“ and “}” characters.

The point character can be used to align the menu.

Markup example:

- optionhtml = "\

```
<option id='SELTELA_OPC1' selected disabled='disabled'>Choose a screen...</option>\
<optgroup label='Substation X'>\
  <option value='svg/subst_x1.svg'>SUBST X 1.........[1]{X}</option>\
  <option value='svg/subst_x2.svg'>SUBST X 2.........[2]{C}</option>\
</optgroup>\
<optgroup label='Substation Y'>\
  <option value='svg/subst_y.svg' >SUBST Y...........[3]{Y}</option>\
</optgroup>\
<optgroup label='Substation Z'>\
  <option value='svg/subst_z.svg' >SUBST Z...........[4]{Z}</option>\
</optgroup>\
";
```

Warning: there can't be spaces after the “\” character.

Client mode configuration

In this mode a machine is configured to be a client of a server HMI.

A Client HMI doesn't need a point list database nor SVG screens. it obtains all the viewers displays from the server. The only process needed is the *HMIShell.exe* or even without it the displays can be opened by shortcuts directly in the browser or through the *index.html* page.

Proceed a normal OSHMI setup and then configure the *hmishell.ini* and the *mon_proc.ini* files.

In the *conf/mon_proc.ini* just put *HIDE=1* and comment every other options.

In the *conf/hmishell.ini* file configure the IP addresses for the (up to) 2 servers:

```
[IHMSHELL]
...
SERVER1=10.63.3.62
SERVER2=10.63.3.63
...
```

Also it is possible to customize server addresses according to the OS logged-in username.

```
[IHMSHELL]
...
SERVER1_USERNAME=10.63.3.1
SERVER2_USERNAME=10.63.3.2
...
SERVER1_OTHERUSERNAME=10.63.5.1
SERVER2_OTHERUSERNAME=10.63.5.2
...
```

In the server machines you must allow access to client machines in the *c:\oshmi\conf\nginx_access_control.conf* file. Example:

```
# IP-based access control
# allow local access only by default
allow 127.0.0.1;
# to allow more clients configure the following option
# allow _IP_hmi_client_;
allow 10.63.3.77; # client machine 1
allow 10.63.3.78; # client machine 2
deny all;
```

Data Client mode configuration

In this mode a machine is configured to be a data only client of a server HMI.

A Data Client HMI doesn't need a point list database but it must have the SVG screen files. It obtains only data from the server. The only process needed is the *HMIShell.exe* or alternatively displays can be opened by shortcuts directly in the browser or through the *index.html* page.

This mode makes possible extremely low bandwidth communications with the remote server. This is ideal for satellite, GPRS and other applications that require very low bandwidth comms.

Data can be encrypted (using https/client certificates) and compressed by the Nginx remote server (see the *Securing OSHMI Servers* section).

Proceed a normal OSHMI client setup (see previous section *Client Mode Configuration*).

Additionally, configure your data server in the following file:

C:\oshmi\htdocs\pntserver.js

Configure the line bellow pointing to your server (protocol http or https, ip address and port):

var ServerPath = 'http*(s)*://*server_ip_addr*:*port*/htdocs/';

Screen editor - Inkscape+SAGE (SCADA Animation Graphic Editor)

Original Inkscape SAGE manual: http://www.integraxor.com/doc/ug/sage.html.

Inkscape Online Book: http://tavmjong.free.fr/INKSCAPE/MANUAL/html/

The *Inkscape+SAGE* editor is used to edit OSHMI SVG screens. SCADA animations and XSAC language representation in the SVG file, originally designed for the *Integraxor* animation engine were adapted to be used with the OSHMI Animation Engine. The attributes and the XSAC language are the same, but the meaning of the attributes is different. To understand the meaning and utilization of the attributes with the OSHMI Screen Viewer Animation Engine, this reference must be consulted and the original meanings and documentation must be ignored!

Please configure each new screen with 2400 x 1500 pixels (File | Document Properties | Page | Page Size | Custom Size). This is a reference size, the actual drawing can be larger.

Screens must be saved with the native (Inkscape SVG, *.svg) format.

To edit SCADA properties of an object, right-click the mouse and choose *Object Properties*.

Follow below a list of attributes that can be utilized.

Get – for text objects only, retrieves and shows a point floating point value.

In the *Tag* field, put the point number or tag to be retrieved. The fields *Alignment* and *Type* are ignored.

There are 3 ways to format values obtained by the “Get” directive. When the text of the object contains the “|” (pipe) character, it is used convention C. When the text contains the “%” character, it is be used convention A. In all other cases it is used convention B.

A) Printf convention. To format values use the standard C language printf convention in the text of the object (e.g: “%5.2f”). For analog values, use “%f”. For string values, use “%s”. For a complete printf convention reference, see here and here. This convention can be used to format number and string values.

B) d3.format convention. The d3.format convention can be used. For a reference see here and here. The character “~” should be used in place of “%” when d3.format percent convention is necessary. The default locale is US English to change it, call d3.formatLocale from a script. This convention can be used to format only numeric values.

C) Boolean convention. For Boolean values, use “off_text|on_text|failed_text”, to show custom texts based on the tag value and quality.

It's possible to represent flow direction with an arrow in place of the value signal using the following codes (positioned where you want the arrow to be shown):

'u^': up pointing arrow for positive values (down for negative values);

'd^': down pointing arrow for positive values (up for negative values);

'r^': right pointing arrow for positive values (left for negative values);

'l^': left pointing arrow for positive values (right for negative values);

'a^': shows only the absolute value.

Examples of formatting, considering the value = -23.456:

format text	*shown text*
%6.2f	-23.47
%08.3f	-023.456
%1.0f	-23
%5.2fu^	23.47↓
l^%.1f	→23.1

Examples of formatting, considering the value = 123456789.123:

format text	*shown text*
s	123.456789123M
.3s	123M

The time of the last alarm for the point can be obtained in place of the value. When the alarm is acknowledged the time is removed, but if the current state is “abnormal” the time is maintained. In this case the *Tag* field must be TMPxxxxx where xxxxx is the point number and the formatting text must be specified with the *printf* “%s” option.

For digital points can be used “OFF_TXT|ON_TXT” syntax for displaying text messages for the OFF an ON states.

Color: changes the color of objects according to limits for the value of points. Applies to text, rectangles, circles and paths.

Each line int the list of limits contains the following fields:

- *Tag*: tag or point number;
- *Limit*: value limit, the color will be used for values equal or greater the the limit;
- *Color Name/Code*: desired color.

The last true condition will be considered and the other ignored.

The field *Limit* can have also some special values:

- a - for alarm not yet acknowledged
- n - for not normal value
- c - for frozen value (not changing for some time)
- f - for failed value

For digital points the IEC60870-5 double point will be tested for the following conditions:

- 0 – indeterminate or intermediate state
- 1 – off state
- 2 – on state
- 3 – indeterminate state
- 128 – indeterminate or intermediate state plus invalid value
- 129 – off state plus invalid value
- 130 – on state plus invalid value
- 131 – indeterminate state plus invalid value

The colors are the SVG colors (named or #RRGGBB). “none” is the transparent color.

A single color value will be used as fill and stroke colors. To specify different fill and stroke separate 2 color values by a “|” (pipe) character. Example: “red|green” = red fill and green stroke.

A void fill color like in “|yellow” affects only the stroke keeping the fill is unaltered.

A void stroke color like in “black|” affects only the fill keeping the stroke is unaltered (recommended for text).

There are color shortcuts defined in *conf/config_viewers.js*:

- "-clr-bgd" – shortcut for the background color;
- "-clr-tbr" – shortcut for the toolbar color;
- "-clr-01" – first user defined shortcut (ScreenViewer_ColorTable[1]);
- "-clr-02" – second user defined shortcut (ScreenViewer_ColorTable[2]);
- …
- … up to 99 user defined color shortcuts.

These colors shortcuts can be used also inside Vega visualization specifications.

To interpolate colors between 2 values, use @color in the color field (fill or stroke) in the final line.

Example:

To make fill colors that varies continuously between white and red proportionally to values between 0 and 10 (for point 100).

```
Tag     Value   Color
100     0       white
100     10      @red|
```

In the field *Color Name/Code* it's possible to change a SVG attribute instead of the color with the “attrib:” prefix. There must be a space after “attrib:”. Examples:

```
attrib: opacity=0.5;
attrib: style=font-size:20px;fill:red;
```

Also in the field *Color Name/Code* it's possible to run a *Javascript* short script with the “script:” option. There must be a space after “script:”.

The function $W.Animate can be used to animate objects. The first parameter is the object to be animated (“thisobj” reprresents the current object); the second is the animation type ('animate', 'set', 'animateTransform', 'animateColor' or 'animateMotion'); the third is the animation options. Examples:

```
script: $W.Animate( thisobj, 'animate', {'attributeName': 'x', 'from': 0, 'to': 10, 'fill':
'freeze', 'repeatCount': 5, 'dur': 5 } ); // animates on axis x, from 0 to 10 seconds,
repeats 5 times.
script: $W.Animate( thisobj, 'animate', {'attributeName': 'width', 'from': 45, 'to': 55,
'repeatCount':5,'dur': 1 } ); // animates width between 45 and 55, 5 times in 1 second.
script: $W.Animate( thisobj, 'animate', {'attributeName': 'width', 'values': '45;55;45',
'repeatCount':5,'dur': 1 } ); // animates width for the values 45,55 and  45, 5 times in 1
second.
```

Supports all SVG attributes, as in:

http://www.kevlindev.com/tutorials/basics/animation/js_dom_smil/index.htm

http://www.w3.org/TR/SVG/animate.html

To load and change images dinamically use the function “$W.LoadImage” as this:

```
script: $W.LoadImage( thisobj, 'clipart/modem.png');
```

Bar: changes the height of the object.

In the *Tag* field, put the point number or tag.

The fields *Min* and *Max* represents the expected spread of point values. The height of the object will be 100% of its size when the value of the point is equal to the *Max* value and 0% when equal to the value defined in the *Min* field.

Changes in different directions can be obtained by just rotating the object.

Opacity: changes the opacity of the object according to the value of a point.

In the *Tag* field, put the point number or tag.

The fields *Min* and *Max* represents the expected spread of point values. The opacity of the object will be 100% (totally solid) when the value of the point is equal to the *Max* value and 0% (totally transparent) when equal to the value defined in the *Min* field.

For digital points consider the value 0 for the *ON* state and 1 for the *OFF* state (least significant bit of double point). For *Min*=0 and *Max*=1 the object will be solid for the *OFF* state and disappear for the *ON* state. With *Min*=1 and *Max*=0 the reverse effect will be obtained.

Rotate: Rotates the object according to value.

The *Tag* field must be the point number or tag.

Fill the fields *Max* and *Min* according to value range of point. When the point reaches the value of *Max* the object will rotate 360 degrees. The object will not rotate when the point have the value of *Min*. The rotation is clockwise when *Max* > *Min*, to invert the rotation direction, let *Min* be greater than *Max*.

Tooltips: A text that will be shown when the mouse is over the object. The fields *Line 1* to *Line 5* can be filled with the lines of text to be presented. *Size* and *Style* are ignored.

The tooltips can contain *Javascript* code between "!EVAL" and "END" marks. Use "V[nnn]" or WebSAGE.getValue("tag" or point number) to obtain point values in the *Javascript* expression. The expression will be evaluated and the resulting value of it will be shown. What is out of the "!EVAL" and "END" marks will be presented. Clone object can receive the "V[%n]" to obtain the point value of the cloned object (see the "Clone" attribute).

Example: Consider a point with number 123 with a value of 22.1 and a point 456 with a value of 10.5.

Line 1: "AL11+AL12=!EVAL V[123]+V[456] !END MW", will present: "AL11+AL12 = 32.6 MW".

Slider: Allows to move the object in a straight line according to the value of the associated point.

The *Tag* field must be the point number or tag. The fields *Max* and *Min* must be filled with the desired range of variation for the point. The object must be cloned (Edit | Clone | Create Clone or ALT+D). The original object defines the initial position (this position will be reached when the value is equal to *Min*). The clone object must be moved to the final position (the position to be reached when the value is equal to *Max*). Movement in the reverse direction can be obtained using switching the values of *Min* and *Max*.

Zoom: Allows to define a zoom region that is extended to the full viewer when clicked. This attribute will only works for rectangle object. This must be the top object and have a opacity >0 like 10%.

Script: Associates a *Javascript* script to a event or defines a Vega specification.

Available events:

mouseup: release the mouse button.

mousedown: mouse click.

mouseover: mouse cursor entering object.

mousemove: mouse cursor moving over object.

mouseout: mouse cursor leaving object.

keydown: key pressed.

exec_once: execute a script one time only after the screen is loaded and parsed.

exec_on_update: execute a script when data is updated.

Vega specification markup options:

vega: old style Vega 1/2 specification. In the first line of the script must be written the tag list comma separated. In the next line either a URL to a specification or the specification itself beginning with a "{" char.

vega4: new style Vega 3/4/5 specification. In the first line of the script must be written the tag list comma separated. In the next line either a URL to a specification or the specification itself beginning with a "{" char.

vega-lite: vega-lite specification. In the first line of the script must be written the tag list comma separated. In the next line either a URL to a specification or the specification itself beginning with a "{" char.

vega-json: old style Vega 1/2 specification with no tags associated. In the first line of the script must be put a URL to a specification or the specification itself beginning with a "{" char.

vega4-json: new style Vega 3/4/5 specification with no tags associated. In the first line of the script must be put a URL to a specification or the specification itself beginning with a "{" char.

It is possible to set the number of minutes to retrieve for historical data (except for **vega-json)** putting the pipe character and a number after the tags list (ex.: "TAG1|15"). If number of minutes is negative, plot from the last round time (for the range).

See Vega project site for tools and documentation of syntax: https://vega.github.io/vega/.

In the Vega specification ("data"/ "values" section), use the following markup:

"PNT#1" to retrieve the current value of the fist point in the point list;

"TAG#1" to retrieve the tag of the fist point in the point list;

"LMI#1" to retrieve the inferior limit of the fist point in the point list;

"LMS#1" to retrieve the superior limit of the fist point in the point list;

"FLG#1" to retrieve the qualifier flags of the fist point in the point list;

"FLR#1" to retrieve the failure of the fist point in the point list;

"SUB#1" to retrieve the substation of the fist point in the point list;

"BAY#1" to retrieve the bay description of the fist point in the point list;

"DCR#1"to retrieve the description of the fist point in the point list;

"HIS#1" to retrieve the historical curve of the fist point in the point list;

For scripts, the function ***$W.Animate()*** and *thisobj* variable can be used to animate objects:

```
$W.RemoveAnimate(thisobj); // remove previous animations
$W.Animate( thisobj, "animate", {"attributeName": "x", "from": 208 ,"to": 300, "repeatCount":
5, "dur": 5} ); // animate on axis x
$W.Animate( thisobj, "animate", {"attributeName": "y", "from": -301 ,"to":-400,
"repeatCount": 5, "dur": 5} ); // animate on axis y
```

It's recommended to use `$W.RemoveAnimate(thisobj)` before creating a new animation to avoid cumulative animations.

Other useful function allows to toggle the visibility of a object and also apply a translation to it:

```
$W.ShowHideTranslate( 'id_of_object', x, y );
```

To make a object invisible immediately after a screen load, use the Inkscape XML editor, go to the root SVG node and create a "*onload*" event with the following code:

```
$W.ShowHideTranslate( 'id_of_object', 0, 0 );
```

This will run after a page load and will hide the object.

The function ***$W.directCommandExec(point, value)*** can be used to execute commands from scripts (no confirmation will be required). Point parameter is the point number of command. Value parameter is the value of the command (can be a number for analog commands or "ON" and "OFF" for digital commands.

The function ***W$.makeDraggable(obj)*** can be used to make an object draggable.

Text: This attribute will display predefined texts associated to ranges of point values. Works only for text objects.

The *Tag* field must be filled with the point number or tag.

The list of *Tag Value*'s and associated *Tag Text*'s must be created with ascending order of value. The value of the point will be tested against the list of *Tag Value*'s to be grater than or equal to each of it. The last true condition will cause the associated text to be presented.

Use a *Tag Value* of "f" to test for invalid values and "a" for an unacknowledged alarm.

For digital points consider the IEC60870-5 double point values 0, 1, 2, 3, 128, 129, 130 and 131 as shown previously in the *Color* attribute.

Clone or Faceplate: This powerful concept permits to replicate groups of animated objects associating each replica to different points (or different set of points).

A model group of objects can be created associating the all the point tags of animations of object(s) to a indirection like "%n" (use "%m", "%p", etc. to use more tag indirections in the same model). Group all related objects that will compose the model. Then use the *Clone* attribute to resolve the indirections in the object with the list of *Variable*'s and *Value*'s. *Variable* is the indirection variable like "n" (here you must not put a "%", just the character of the variable) and *Value* is the point number associated to it. So all animations in the grouped objects will be related to the values of the indirected point. Copy and paste the grouped object and change the point numbers of the indirected(s) variable(s) to obtain new objects with the same animations linked to other points.

Popup: Allows to control the access point dialog when the object is clicked.

The field *Source* can be:

- A point number – point to opened in the point access dialog when the object is clicked.
- "block" - blocks the point access dialog when the object is clicked.
- "notrace" - allow point access dialog when object is clicked but do not highlight the object when accessed.
- "preview:URL"**:** presents a preview of the URL when mouse is over the object. The *width* an *height* parameters define the preview window size.

Set: used to configure the following special functions:

#exec or **#exec_once** in the field *Tag* - executes once the script entered in the field *Source*.

#exec_on_update in the field *Tag* - executes the script entered in the field *Source* each time data is updated (changed).

#copy_xsac_from in the field *Tag* - copies the XSAC attributes from other model object(s) to the current object. Use the field *Source* to indicate the ID of the model object. Multiple ID's of model objects can be entered in the field *Source* separating them with commas. The other fields are ignored.

#set_filter in the field *Tag* - defines a filter (by the point ID) for the data presented in the *Alarm Box*, Use the field *Source* to enter the text of the filter. The other fields are ignored.

#arc in the field *Tag* – draws a donut chart. The point number or tag must be in the *Source* field. In the *Prompt* field there are four parameters separated by commas: minimum value (normally zero), maximum value (for a 360 degree arc), the inner circle radius and the animation time in ms.

#radar in the field *Tag* – for a rectangle object, defines a radar (spider) graphic. List the points in the *Source* field separated by commas. The field *Prompt* can be used to change the configuration of the chart, by applying conventions from https://github.com/alangrafu/radar-chart-d3. Write the attribute names using double quotes. Ex:

```
{ “levels”:5, “maxValue”:200, “axisText”: false }
```

#vega, #vega4, #vega-json, #vega4-json or **#vega-lite** in the field *Tag* - defines a Vega (version 1/2 or version 3/4/5) or VegaLite chart. List the points in the *Source* field separated by commas. You can set the number of minutes to retrieve for historical data (except for **#vega-json)** putting the pipe character and a number after the tags list in the Source field (ex.: “38038|15”). If number of minutes is negative (**#vega4**), plot from the last round time (for the range). The field *Prompt* must contain the Vega chart specification (JSON code that must begin with a “’{”) or a URL link to a file (ex.: “../charts/stacked.json”).
See Vega project site for tools and documentation of syntax: https://vega.github.io/vega/.

In the Vega specification (“data”/ “values” section), use the following markup:

"PNT#1" to retrieve the current value of the fist point in the point list;
"TAG#1" to retrieve the tag of the fist point in the point list;
"LMI#1" to retrieve the inferior limit of the fist point in the point list;
"LMS#1" to retrieve the superior limit of the fist point in the point list;
"FLG#1" to retrieve the qualifier flags of the fist point in the point list;
"FLR#1" to retrieve the failure of the fist point in the point list;
“SUB#1” to retrieve the substation of the fist point in the point list;
“BAY#1” to retrieve the bay description of the fist point in the point list;
“DCR#1”to retrieve the description of the fist point in the point list;
“HIS#1” to retrieve the historical curve of the fist point in the point list;

Obs.: Special codes to obtain other point attributes instead of the point value in the XSAC animations, nnn represents the point number:

- **!ALMnnn** = returns "1" for the unacknowledged alarm or abnormal state and "0" for the acknowledged and normal state.
- **!ALRnnn** = returns "1" for the unacknowledged alarm and "0" for acknowledged or not alarmed state.
- **!TMPnnn** = returns the time of the last alarm for the point.
- **!SLIM***nnn* = returns the upper analog limit.
- **!ILIM***nnn* = returns the lower analog limit.
- **!TAG***nnn* = returns the point tag (ID).
- **!DCR***nnn* = returns the point description.
- **!STON***nnn* = returns the text for the ON state of digital points.
- **!STOFF***nnn* = returns the text for the OFF state of digital points.
- **!STVAL***nnn* = returns the text for current state of digital points.
- **!EVAL** *expression* = evaluates a *Javascript* expression (use $V(*"tag" or point number)* or WebSAGE.getValue("tag" or point number) to obtain point values in the expression).

Open: used to open new pages, screens, previews or trend graphic plot:

For *Source Type* = URL:

- Field *Source* = new:URL: open new page with URL contents.
- Field *Source* = preview:URL: open preview box with URL contents.
- Field *Source* = screen_name: go to another screen (from the screen list).

For rectangle objects and *Field Type* = *Tag* draw a line with trend graphic plot inside the rectangle with the rectangle stroke width and color. Parameters:

- Field *Source* = point number.
- Field *X-position* = interval of historic plot in minutes, if 0 (zero) plot all available values, if > 0 return the mean value of each period.
- Field *Y-position* = vertical offset value.
- Field *Width* = horizontal (X) range in seconds, plot time window continuously moved, current value plotted to the right edge. If negative, plot from the last round time (for the range), current value plotted from left to right, restart plot when current date bigger than final time for the last range (can be used to plot values ahead of time, like forecasts).
- Field *Height* = vertical (Y) range for point values.

Hint to debug scripts in SVG screen files:

- include the keyword “debugger;” at the beginning of the script.
- open the screen in the Screen Viewer.
- press F12 to open the browser Developer Tools and then F5 to reload.
- The Chromium browser will stop execution when it find the introduced “debugger;” breakpoint. Use the execution control keys F10, F11, F9, F8 to forward execution.

Screen templates using tag prefixes

It is possible to reuse SVG screens using tag prefixing for cases when data is repeatable from different origins, e.g. when there is an IED applied multiple times.

All repeatable tags in the screen must be prefixed with the markup $$#1 (... $$#20, up to 20 prefixes).

Lets say there are the following tags for IED's in the point list:

IED1:
IED1_DATA1
IED1_DATA2
IED1_DATA3

IED2:
IED2_DATA1
IED2_DATA2
IED2_DATA3

Screens can be tagged as:

$$#1_DATA1
$$#1_DATA2
$$#1_DATA3

When opening the screen it must be passed a parameter IDPREFIX1 (... IDPREFIX20) identifying the prefix.

Opening this shortcut will open the screen with data from IED1:

../htdocs/screen.html?IDPREFIX1=IED1_&HIDETB=1&SELTELA=../svg/prefixed-screeen.svg

Opening this shortcut will open the screen with data from IED2:

../htdocs/screen.html?IDPREFIX1=IED2_&HIDETB=1&SELTELA=../svg/prefixed-screeen.svg

And so on, so that you can have the same screen for any number of repeatable IED.

Lua Script Language (for calculations, automatons and logic real time processing)

The *Lua* 5.1 (http://www.lua.org) scripting language is integrated in *OSHMI* to allow server side scripts to be run over the real time database.

The file "*c:\oshmi\scripts\script.lua*", hosts the user script that will be run once in the initialization of the system (*webserver.exe*) and subsequently the function *ScriptCycle* will be called periodically (each 2 seconds as default).

A set of functions is available to use with *Lua* interfacing the real time database. New functions can be created in the (*webserver.exe)* lua_u.cpp module source code.

The *Lua* language syntax is documented online at *http://www.lua.org/manual/5.1/* and *http://www.lua.org/pil/index.html*, also in the file "*c:\oshmi\docs\ lua_reference_manual.pdf*".

It was included the useful *BitOp* (http://bitop.luajit.org/) extensions for bit operations.

The *webserver.exe* does have a "*Script*" button that opens an interactive *Lua* environment that can be used to debug and test code.

```
-- HMI functions available:
--
-- hmi_print : print values on script dialog
--             args: comma separated strings
--
-- hmi_isprimary : returns true if hmi is in primary state
--
-- hmi_settimer: adjust cycle time for function ScriptCycle
--               args: time in ms
--
-- hmi_activatebeep: beeps the alarm sound
--
-- hmi_getpoint: obtains point info
--               args: point number
--               returns: analog value (float) or digital double (0,1=OFF,2=ON,3)
--                        fail state (1=failed, 0=normal)
--                        point type (1=analog, 0=digital)
--                        substituted? (1=yes, 0=no)
--
-- hmi_writepoint: writes on point
--                 args: point number, value, fail state
--                 returns: 0 if ok, 1 if point not found
--
-- hmi_blkpoint: blocks command of supervised point
--               args: point number, description message (annotation)
--               returns: 0 if ok, 1 if point already annotated, 2 if point not found
--
-- hmi_unblkpoint: unblocks command of supervised point, only if same annotation message
--                 args: point number,description message (annotation)
--                 returns: 0 if ok, 1 if point already annotated, 2 if point not found
--
-- hmi_sendcmd: sends command
--              args: command point number, value
--              returns: 0 if ok, 3 if point not found, 4 if blocked, nil if wrong arg number
```

Viewers Configuration File “config_viewers.js”

This file allows the user to configure customizable some aspects of the viewers.

The file c:\oshmi\conf\config_viewers.js is not overwritten when OSHMI is updated so the user configurations is preserved. To change an option, uncomment and edit.

```
//// Viewers Configuration Parameters
//
//// list of colors representing priorities for the alarms/tabular/events viewers
//var ColorOfPriority = [ "red", "yellow", "goldenrod", "plum", "silver", "silver", "silver", "silver",
"silver", "silver", "silver" ];
//// list of colors representing the first up to last substation for the alarms/tabular/events viewers
//var ColorOfSubstation = [ "cadetblue", "brown", "green", "magenta", "orange", "darkcyan", "goldenrod",
"deepskyblue", "indigo", "lightseagreen" ];
//
//// Events Viewer ------------------------------------------------------------------------------
//var EventsViewer_ToolbarColor = '#BBBBBB'; // toolbar color
//
//var EventsViewer_Font = 'Source Sans Pro,calibri,consolas,arial,helvetica'; // font
//
//var EventsViewer_TableColor = '#E0E0E0'; // table background color
//var EventsViewer_GridColor = '#F0F0F0'; // table grid color
//
//var EventsViewer_AlmTxtColor = 'blue'; // alarmed color
//var EventsViewer_FailTxtColor = 'white'; // failed value color
//var EventsViewer_AckTxtColor = '#484848'; // acknowledged event color
//var EventsViewer_ElimTxtColor = '#B0B0B0'; // eliminated event color (until removed)
//
//var EventsViewer_LineColor = '#E0E0E0'; // line color
//var EventsViewer_LineColorHighight1 = 'gray'; // Type 1 highlighted alarms color (operated protection)
//var EventsViewer_LineColorHighight2 = 'black'; // Type 2 highlighted alarms color (circuit breakers
state)
//
//var EventsViewer_RefreshTime = 15; // refresh time in seconds (digital changes triggers a faster refresh)
//
//// Event time tag configuration
//// 0 = GPS (field time)
//// 1 = local time (time when HMI detected the event)
//// 2 = chosen by operador (defaults to GPS time)
//// 3 = chosen by operador (defaults to local)
//var EventsViewer_TimeGPSorLocal = 2;
//
//var EventsViewer_AllowFilter = 1;  // 0: no filters, 1=operator can set filter by substation
//
//var EventsViewer_Notific = 1;  // 0: disable desktop notifications, 1=enable desktop notifications
//
//// Tabular Viewer -----------------------------------------------------------------------------
//var TabularViewer_ToolbarColor = '#AA9E97'; // toolbar color
//
//var TabularViewer_Font = 'Source Sans Pro,calibri,consolas,arial,helvetica'; // font
//
//var TabularViewer_TableColor = '#DCDCEE'; // table background color
//var TabularViewer_GridColor = '#ECECEE'; // table grid color
//
//var TabularViewer_AlmTxtColor = 'blue'; // alarmed color
//var TabularViewer_FailTxtColor = 'white'; // failed value color
//var TabularViewer_AckTxtColor = '484848'; // acknowledged alarm color
//
//var TabularViewer_LineColor = '#DCDCEE'; // line color
//var TabularViewer_LineColorDestaq1 = 'gray'; // Type 1 highlighted alarms color (operated protection)
//var TabularViewer_LineColorDestaq2 = 'black'; //  Type 2 highlighted alarms color (circuit breakers
state)
//
//var TabularViewer_RefreshTime = 7; // refresh time (seconds)
//
//// Screen Viewer ------------------------------------------------------------------------------
//
//var ScreenViewer_RefreshTime = 5; // refresh time in seconds (digital changes triggers a faster refresh)
//
//// SVG Screen dimensions (must match <svg> tag dimensions on SVG screens)
//// <svg width="2400" height="1500">
//var ScreenViewer_SVGMaxWidth = 2400; // default SVG screen width in pixels
//var ScreenViewer_SVGMaxHeight  = 1500; // default SVG screen height in pixels
//
//var ScreenViewer_Background  = '#DDDDDD'; // background color for Inkscape SAGE SVG screens
//var ScreenViewer_ToolbarColor = 'lightslategray'; // toolbar color
//var ScreenViewer_RelationColor = 'red'; // color for relationated points when shown by [shift+backspace]
//var ScreenViewer_TagFillColor = 'yellow'; // annotation tag fill color
//var ScreenViewer_TagStrokeColor = 'red'; // annotation tag stroke color
//var ScreenViewer_TagInhAlmFillColor = 'khaki'; // inhibited alarm tag fill color
```

```
//var ScreenViewer_TagInhAlmStrokeColor = 'darkblue'; // inhibited alarm tag border color
//var ScreenViewer_DateColor = "white"; // data update time color
//var ScreenViewer_TimeMachineDateColor = "black"; // time machine data update color
//var ScreenViewer_TimeMachineBgColor = "green"; // time machine background color
//var ScreenViewer_AlmBoxTableColor = "#DCDCEE"; // alarm box table color
//var ScreenViewer_AlmBoxGridColor = "whitesmoke"; // alarm box grid color
//var ScreenViewer_BarBreakerSwColor = "steelblue"; // color for DJ, SC and bars
//var ScreenViewer_ShowScreenNameTB = 1; // show screen name on toolbar
//
//// user color palette (use in Inkscape+SAGE color fields as "-cor-05" or "-cor-49")
//var ScreenViewer_ColorTable = new Array();
//ScreenViewer_ColorTable[0]  = 'white'; //
//ScreenViewer_ColorTable[1]  = 'white'; // sc ou dj falha
//ScreenViewer_ColorTable[2]  = 'white'; //
//ScreenViewer_ColorTable[3]  = ScreenViewer_Background; // dj ab
//ScreenViewer_ColorTable[4]  = ScreenViewer_BarBreakerSwColor; // sc ou dj fc
//ScreenViewer_ColorTable[5]  = ScreenViewer_BarBreakerSwColor; // sc ab
//ScreenViewer_ColorTable[6]  = 'cornsilk'; // sc ou dj 00
//ScreenViewer_ColorTable[7]  = 'cornsilk'; // sc ou dj 11
//ScreenViewer_ColorTable[8]  = ScreenViewer_BarBreakerSwColor; // borda do dj ab
//ScreenViewer_ColorTable[9]  = '#AAAAAA'; // unidades de medidas
//ScreenViewer_ColorTable[10] = 'cadetblue'; // outras medida ok
//ScreenViewer_ColorTable[11] = 'red'; // medida fora de faixa
//ScreenViewer_ColorTable[12] = 'white'; // medida ou estado falhado
//ScreenViewer_ColorTable[13] = ScreenViewer_BarBreakerSwColor; // titulo da tela (SE)
//ScreenViewer_ColorTable[14] = ScreenViewer_BarBreakerSwColor; // texto de linha
//ScreenViewer_ColorTable[15] = '#AAAAAA'; // número de equipamento
//ScreenViewer_ColorTable[16] = ScreenViewer_BarBreakerSwColor; // barra 230kV
//ScreenViewer_ColorTable[17] = ScreenViewer_BarBreakerSwColor; // barra 138kV
//ScreenViewer_ColorTable[18] = ScreenViewer_BarBreakerSwColor; // barra de 69kV
//ScreenViewer_ColorTable[19] = ScreenViewer_BarBreakerSwColor; // barra de alimentadores 13kV/23kV
//ScreenViewer_ColorTable[20] = 'cadetblue'; // medida de MW ok
//ScreenViewer_ColorTable[21] = 'cadetblue'; // medida de MVAr ok
//ScreenViewer_ColorTable[22] = 'cadetblue'; // medida de kV ok
//ScreenViewer_ColorTable[23] = 'cadetblue'; // medida de corrente ok
//ScreenViewer_ColorTable[24] = 'cadetblue'; // medida de tap ok
//ScreenViewer_ColorTable[25] = ScreenViewer_BarBreakerSwColor; // cor da barra de 500kV
//ScreenViewer_ColorTable[26] = 'gray'; // texto estados
//ScreenViewer_ColorTable[27] = 'gray'; // grade de estados
//ScreenViewer_ColorTable[28] = 'darkgreen'; // estados off
//ScreenViewer_ColorTable[29] = 'darkred'; // estado on
//ScreenViewer_ColorTable[30] = 'red'; // estado de alarme
//ScreenViewer_ColorTable[31] = 'lightgray'; // quadro de estados
//ScreenViewer_ColorTable[32] = 'black'; // borda do quadro de estados
//ScreenViewer_ColorTable[33] = '#D7D7D7'; // área de operação
//ScreenViewer_ColorTable[34] = 'gray'; // símbolo de aterramento
//ScreenViewer_ColorTable[35] = ScreenViewer_BarBreakerSwColor; // estado normal
//ScreenViewer_ColorTable[36] = 'mediumvioletred'; // estado anormal
//ScreenViewer_ColorTable[37] = 'red'; // estado alarmado
//ScreenViewer_ColorTable[38] = '#999999'; // texto estático
//ScreenViewer_ColorTable[39] = '#DDE8DD'; // quadro de status normal
//ScreenViewer_ColorTable[40] = 'yellow'; // quadro de status alarme
//ScreenViewer_ColorTable[41] = 'deepskyblue'; // medida congelada
//ScreenViewer_ColorTable[42] = 'red'; // medida alarmada
//ScreenViewer_ColorTable[43] = 'cadetblue'; // outras medidas ok
//ScreenViewer_ColorTable[44] = 'red'; // alarm priority 0
//ScreenViewer_ColorTable[45] = 'yellow'; // alarm priority 1
//ScreenViewer_ColorTable[46] = 'orange'; // alarm priority 2
//ScreenViewer_ColorTable[47] = 'fucsia'; // alarm priority 3 (diagnóstico)
//ScreenViewer_ColorTable[48] = '#CCCCCC'; // state box fill color inactive
//ScreenViewer_ColorTable[49] = '#CCCCCC'; // state box border color inactive
//ScreenViewer_ColorTable[50] = '#505050'; // state box active text color
//ScreenViewer_ColorTable[51] = 'lightsteelblue'; // state box fill color ON
//ScreenViewer_ColorTable[52] = 'tan'; // state box fill color OFF
//ScreenViewer_ColorTable[53] = '#888888'; // state box inactive text color
//ScreenViewer_ColorTable[54] = 'red'; // state box operated text color
//ScreenViewer_ColorTable[55] = 'lightsteelblue'; // analog range ok
//ScreenViewer_ColorTable[56] = '#777777'; // analog range out of limits
//ScreenViewer_ColorTable[57] = ScreenViewer_BarBreakerSwColor; // analog indicator ok
//ScreenViewer_ColorTable[58] = 'crimson'; // analog indicator out of limits
//ScreenViewer_ColorTable[59] = '#CCCCCC'; // load rectangle MW or MVA or A
//
//var ScreenViewer_SlideShowInterval = 10; // slide show time in seconds
//var ScreenViewer_EnableTimeMachine = 1; // 1 = enable time machine, 0 = disable time machine
//
//// Command dialog
//var CommandDialog_Background = 'wheat'; // background color
//
//// Other options ----------------------------------------------------------------------
//
//// Type 1 highlighted alarms
//var Visors_AlmTxtHighlight1 = 'OPERADO'; // Alarm text to identify Type 1 highlighted alarms
//// Type 2 highlighted alarms
//var Visors_IDTxtHighlight2    = 'XCBR'; // Text present on ID to identify Type 2 highlighted alarms
//// and
//var Visors_DescrTxtHighlight2 = ':estado'; // Text present on DESCRIPTION to identify Type 2 highlighted
alarms
```

Simulation Mode

The simulation mode is configured by the following option of the *hmi.ini* file:

```
[RUN]
SIMULATION=1
```

After the first installation os the system the default mode is simulated.

Analog measurements are validated and have their initial values slightly changed over time.

Digital points are also validated and respond when their associated commands are issued.

Some alarms are randomly activated.

The "*Simulation*" button of *webserver.exe* opens a interactive dialog box where it's possible to enter new values for points.

To turn off the simulation mode just use:

```
[RUN]
SIMULATION=0
```

Qtester104 – IEC60870-5-104 Protocol Driver

This software scans a *IEC60870-5-104* slave injecting the data in the *webserver.exe* real time database server.

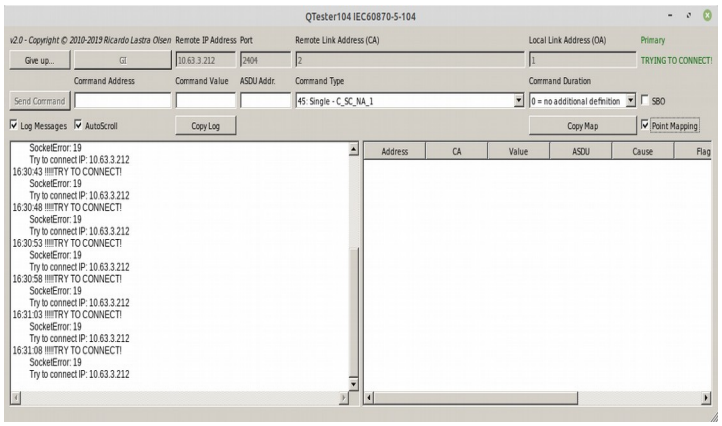
C:/oshmi/qtester104.ini file:

```
[IEC104]

; Local link address
PRIMARY_ADDRESS=1

[RTU1]

; RTU link address
SECONDARY_ADDRESS=2

; RTU IP address
IP_ADDRESS=10.63.3.212

; redundant RTU IP address
;IP_ADDRESS_BACKUP=10.63.3.213

;TCP_PORT=2404

;GI_PERIOD=330

; enable to send commands
ALLOW_COMMANDS=1
```

Only one RTU with redundancy is allowed to be scanned. QTester104 will try to connect to the main RTU, if failed it will try the backup RTU, and so on.

It can operate in redundant mode using 2 server machines, only one will be active and scanning at each time, the other will be in standby until the main scanner is not detected anymore.

Use the “Log Messages” and “Point Mapping” options to debug communications and point database.



Process Monitor (*mon_proc.exe*)

This program is used to monitor the presence of essential processes that need to be active under the Windows OS.

Each process to be monitored must be configured in *mon_proc.ini.*

When a monitored process is not detected it is restarted (MODE=0 and MODE=1). If a process configured to with MODE=1 it is also restarted when not sending the special UDP messages that are expected by *mon_proc.exe*.

It's recommended for a normal HMI to have the *webserver.exe* process monitored.

For a client HMI there are no essential processes, so *mon_proc.exe* is not necessary and the *mon_proc.ini* can be empty.

Example *MON_PROC.INI*:

```
[CONF]
; Hide=0 show mon_proc window, Hide=1 do not show mon_proc window
Hide=0

; EXAMPLE:
;[PROC1]
; File= file name with path
; File=C:\oshmi\bin\webserver.exe ; controled process
; Args= ; command line args
; Exec= ; complete command line (used in place of File+Args) to execute process
; Mode=0 ; only verify process existence
; Mode=1 ; additionally wait for specific UDP messages on port 8081
; RestartCount=12 ; timeout count to restart program (in seconds)

; will run webserver.exe, verify if process exists every 28s and restart it if not found
[PROC1]
File=C:\oshmi\bin\webserver.exe
Args=
Exec=
Mode=0
RestartCount=28
```

Time Tag / Timezone Treatment by OSHMI

Recording and querying of events time tag are treated separately.

Recording:
An incoming event have its field time tag converted to a Unix Time (UTC=GMT) integer in a SQLite table. The time of the reception of the event (local time) is also recorded with the same conversion.

This conversion is accomplished using the daylight saving time (DST) information from the host OS.

The assumption is that the filed is in the same time base and time zone of the HMI server machine.

Queries:
The timezone for queries is configured in the *c:\oshmi\conf**hmi;ini*** file, [WEBSERVER] section, entry TIMEZONE=”UTC”. When commented out this line, the default timezone is “America/Sao_Paulo”.

(List of PHP supported timezones: *http://php.net/manual/timezones.php*).

The GMT time tag is restored converting to this configured timezone.

For queries the OS timezone is not relevant.

So if the OS timezone and the timezone configured in the *c:\oshmi\conf**hmi.ini*** are the same, the time shown in the Events Viewer will be the same received from the field.

If these timezones are different there can be a shift in the hour presented in relation to the field time tag of the original event.

OSHMI Port Utilization Map

Webserver

Real time data HTTP queries: TCP 51908

BDTR local listen: UDP 65280

Listen to data from *iec104m.exe*: UDP 8099

Data send to *iec104m.exe*: UDP 8098

Watchdog messages, send to *mon_proc*: UDP 8081

Listen JSON driver interface: UDP 9100

Send data (commands) to JSON drivers: UDP 9101...9100+n

QTester104

I104M listen: UDP 8098

Data send to I104M/*Webserver.exe*: UDP 8099

Mon_proc

Listen to watchdog update messages: UDP 8081

IEC104M

Listen to *webserver.exe* messages: UDP 8098 (default)

Listen to *webserver.exe* messages: UDP 8097 (optional for Modbus driver)

Listen to *webserver.exe* messages: UDP 8096 (reserved for DNP3 driver)

Data send to *webserver.exe*: UDP 8099

ICCP_client

Redundancy control, listen: UDP 8102

Browser

HTTP requests: TCP 51909

NGINX / PHP

NGINX HTTP server: TCP 51909

PHP-CGI: TCP 9000

OSHMI Linux Install (Wine)

Any modern Linux x86 32 or 64 bit distros can possibly run OSHMI. The tested and recommended distro is Mint 19.0 “Tara” XFCE 32 bits. The instructions here are for the recommended distro, other environments may require different procedures. The HMIShell.exe, Webserver.exe and Inkscape+SAGE modules require Wine 3.xx to run.

Download and install Mint (https://linuxmint.com/download.php).

Login as admin user.

Connect to Internet and install also:

```
sudo apt-get remove thunderbird hexchat transmission-gtk

sudo apt-get update

sudo apt-get dist-upgrade

sudo apt-get install chromium-browser nginx sqlite3 wget curl
php php-fpm php-cgi php-gd php-sqlite3 x11-utils ttf-mscore-
fonts-installer xdotools
```

Install Wine 4.xx, follow instructions here: https://www.tecmint.com/install-wine-on-ubuntu-and-linux-mint/.

```
wget -nc https://dl.winehq.org/wine-builds/Release.key

sudo apt-key add Release.key

sudo apt-add-repository https://dl.winehq.org/wine-builds/
ubuntu/

sudo apt-add-repository 'deb https://dl.winehq.org/wine-builds/
ubuntu/ bionic main'

sudo apt-get update

sudo apt-get install --install-recommends winehq-stable
```

Install updated Winetricks:

```
wget  https://raw.githubusercontent.com/Winetricks/winetricks/
master/src/winetricks

chmod +x winetricks

winetricks corefonts consolas lucida tahoma pptfonts
```

Create an “operator” user and “oshmi” group.

sudo groupadd oshmi

sudo useradd operator

sudo usermod -G oshmi operator

Login as operator.

Run the OSHMI installer with Wine.

```
Wine OSHMI_setup-xxx.exe
cd
ln -s .wine/drive_c/oshmi/ oshmi
cd oshmi
rm -rf browser browser-data nginx_php
rm -rf .db/*.bat
rm -rf bin/mon_proc.exe
```

Put OSHMI stuff under “oshmi” group:

```
su admin
sudo chown -R :oshmi /home/operator/.wine/drive_c/oshmi/*.*
sudo chown -R :oshmi /home/operator/.wine/drive_c/oshmi/*
sudo chmod -R g+w  /home/operator/.wine/drive_c/oshmi/*.*
sudo chmod -R g+w  /home/operator/.wine/drive_c/oshmi/*
sudo chmod -R g+rwxs /home/operator/.wine/drive_c/oshmi/db
sudo chmod +x /home/operator/.wine/drive_c/oshmi/linux/*.sh
sudo chmod +x /home/operator/.wine/drive_c/oshmi/db/*.sh
sudo cp /home/operator/.wine/drive_c/oshmi/linux/nginx/sites-
available/default /etc/nginx/sites-available/
sudo cp  /home/operator/.wine/drive_c/oshmi/linux/
process_*.sh /home/operator/.wine/drive_c/oshmi/db/
sudo usermod -a -G operator www-data
sudo usermod -a -G oshmi www-data
sudo service nginx restart
```

In *hmi.ini*, configure the native Chromium Browser:

```
EVENTS_VIEWER="/usr/bin/chromium-browser --disable-popup-block-
ing --process-per-site --no-sandbox --no-proxy-server --
app=http://127.0.0.1:51909/htdocs/events.html"
TABULAR_VIEWER="/usr/bin/chromium-browser --disable-popup-
blocking --process-per-site --no-sandbox --no-proxy-server --
app=http://127.0.0.1:51909/htdocs/tabular.html"
SCREEN_VIEWER="/usr/bin/chromium-browser --disable-popup-block-
ing --process-per-site --no-sandbox --no-proxy-server --
app=http://127.0.0.1:51909/htdocs/screen.html"
```

```
TREND_VIEWER="/usr/bin/chromium-browser --disable-popup-block-
ing --process-per-site --no-sandbox --no-proxy-server --
app=http://127.0.0.1:51909/htdocs/trend.html"

CURVES_VIEWER="/usr/bin/chromium-browser --disable-popup-block-
ing --process-per-site --no-sandbox --no-proxy-server --
app=http://127.0.0.1:51909/htdocs/histwebview/histwebview.php"

DOCS_VIEWER="/usr/bin/chromium-browser --disable-popup-blocking
--process-per-site --no-sandbox --no-proxy-server --app=http://
127.0.0.1:51909/docs/listdocs.php"

LOGS_VIEWER="/usr/bin/chromium-browser --disable-popup-blocking
--process-per-site --no-sandbox --no-proxy-server --app=http://
127.0.0.1:51909/htdocs/listlogs.php"
```

As “operator” access XFCE Session configuration and add .wine/drive_c/oshmi/linux/autostart.sh to session initialization.

Configure Chromium to allow notifications and popups to the local server.

Exporting Data to MS PowerBI and Tableau using OData

It is possible to export data to MS PowerBI and Tableau using the OData protocol.

OSHMI URL Format for OData:

```
Example URL:
http://127.0.0.1:51909/htdocs/odata.php?FILTER=U1-MTWT,U2-
MTWT&INITDATE=07/11/2017&ENDDATE=07/13/2017&INTERVAL=15
```

Where:

http://127.0.0.1:51909 is the local machine OSHMI server. If the OSHMI server is in other machine just substitute with its IP address.

FILTER=U1-MTWT,U2-MTWT is the tag list filter. You can use point numbers or a tag list comma separated, or a generic filter like KAW2TR3* that will query all tags initiated by KAW2TR3.

INITDATE=07/11/2017 is the initial date in format *mm/dd/yyyy*.

ENDDATE=07/13/2017 is the end date in format *mm/dd/yyyy*.

Omit dates to get only the latest values for the selected points.

INTERVAL=15 is the sample interval in minutes. Use INTERVAL=0 or omit this parameter to get all available data.

MS PowerBI:

Download, install and create an account for PowerBI (https://powerbi.microsoft.com).

Choose “Get Data” from PowerBI the menu, select “OData Feed”.

Type in the URL field:

```
http://127.0.0.1:51909/htdocs/odata.php?FILTER=U1-MTWT,U2-
MTWT&INITDATE=07/11/2017&ENDDATE=07/13/2017&INTERVAL=15
```

Maybe there will be needed to edit the query to change columns data types:

TIMESTAMP to Date/Time

VALUE to Number

FAILED to True/False

POINT_KEY to Integer.

Tableau:

Download, install and create an account for Tableau (https://public.tableau.com).

Create a new Data Source and choose OData.

Type in the “Server” field:

```
http://127.0.0.1:51909/htdocs/odata.php?FILTER=U1-MTWT,U2-
MTWT&INITDATE=07/11/2017&ENDDATE=07/13/2017&INTERVAL=15
```

Exporting Data to MS PowerBI using JSON

It is possible to export data to MS PowerBI using the JSON format.

OSHMI URL Format for JSON:

```
Example URL:
http://127.0.0.1:51909/htdocs/json?FILTER=U1-MTWT,U2-
MTWT&INITDATE=07/11/2017&ENDDATE=07/13/2017&INTERVAL=15&DATEFORMAT=1
```

Where:

http://127.0.0.1:51909 is the local machine OSHMI server. If the OSHMI server is in other machine just substitute with its IP address.

FILTER=U1-MTWT,U2-MTWT is the tag list filter. You can use point numbers or a tag list comma separated, or a generic filter like KAW2TR3* that will query all tags initiated by KAW2TR3.

INITDATE=07/11/2017 is the initial date in format *mm/dd/yyyy*.

ENDDATE=07/13/2017 is the end date in format *mm/dd/yyyy*.

Omit dates to get only the latest values for the selected points.

INTERVAL=15 is the sample interval in minutes. Use INTERVAL=0 or omit this parameter to get all available data.

DATEFORMAT=1 formats date textually, DATEFORMAT=0 formats dates as unix timestamps.

MS PowerBI:

Download, install and create an account for PowerBI (https://powerbi.microsoft.com).

Choose “Get Data” from PowerBI the menu, select “JSON File”.

Type in the name field:

```
http://127.0.0.1:51909/htdocs/json.php?FILTER=U1-MTWT,U2-
MTWT&INITDATE=07/11/2017&ENDDATE=07/13/2017&INTERVAL=15&DATEFORMAT
=1
```

Edit the query, select “list”, choose “convert to table”, press “OK”, click on the split button (next to “Column1”), click “OK”.

Change columns data types:

TIMESTAMP to Date/Time

VALUE to Number

FAILED to True/False

POINT_KEY to Integer.

Close and apply query.

Exporting Data to Tableau using Web Data Connector (JSON)

It is possible to export data to Tableau (including the free Tableau Public version) using the JSON format with help from the Web Data Connector OSHMI tool.

Download, install and create an account for Tableau (https://public.tableau.com).

Create a new Data Source and choose “Web Data Connector”.

In the URL field enter:

```
http://127.0.0.1:51909/htdocs/tableau_wdc.php
```

Replace the IP address if needed, hit ENTER.

Type tags or point numbers in the “Tags” field, you can use suggest as you type feature. The list of tags must be comma separated.

Select initial and end dates for a period or erase dates to get the latest value for each point only.

Select the period of sampling in minutes or put 0 to retrieve all the available data.

Hit the “Get OSHMI Data” button.

Hit “Refresh Data” button.

Go to “Sheet 1” or other sheet.

PostgreSQL / Timescale Add-on

This out-of-the-box preconfigured add-on requires 64bit Windows. It is possible to use 32 bit Windows and Linux versions of PostgreSQL and Timescale, but this will require manual installation and configuration.

Download oshmi_postgresql_addon.zip from OSHMI repository, unzip to c:\oshmi\PostgreSQL\. The installation is managed by the BIGSQL package manager:

https://www.openscg.com/bigsql/docs/pgcli/pgcli/

The Timescale (https://www.timescale.com/) extension is already installed and configured. Tables SOE and HIST are already created as *hypertables*, partitioned by day.

Configure OSHMI to create SQL files for PostgreSQL in c:\oshmi\conf\hmi.ini, section [HIST]:

```
[HIST]
; Do record historic data (SQLITE): 1=yes, 0=no
; Default = 1
; Faz gravação do histórico: 1=sim, 0=não
;RECORD=1

; Dead band scale: 100 = normal (100%), 200 = doubles dead band = less data will be recorded
; Default = 100
; Escala de banda morta percentual: 100 = normal (100%), 200 = duplica banda morta = menos
dados serão gravados
;DEADBAND_FACTOR=100 ;

; Number of days to keep events and historical data
; Default=36
;LIFETIME=36

; Enable SQL text files generation to feed a PostgreSQL database.
; 1 = Enable, 0 = Disable
; Default = 0
DB_POSTGRESQL=1
```

Admin user=postgres, password=oshmi.

See the file c:\oshmi\PostgreSQL\install.txt to access the definition and comments of the available tables and views.

To use an external server, edit the files in c:\oshmi\db: process_pg_dumpdb.bat, process_pg_hist.bat, process_pg_soe.bat to change the IP addresses of parameter -h from 127.0.0.1 to the IP of the PostgreSQL server. Put the database credentials in the “%APPDATA%\postgresql\pgpass.conf” file.

References:

- https://www.postgresql.org
- https://www.postgresql.org/docs/9.1/libpq-pgpass.html
- https://www.timescale.com
- https://www.openscg.com/bigsql/

Grafana Add-on

This out-of-the-box preconfigured add-on requires 64bit Windows and the PostgreSQL/Timscale add-on.

It is also possible to use Linux versions of Grafana, but this will require manual installation and configuration.

First download oshmi_grafana_addon.zip from OSHMI repository, unzip to c:\oshmi\grafana\.

Admin user=admin, password=oshmi.

Default user=oshmi, password=oshmi.

The PosqgreSQL user “grafana” (password=”oshmi”) does have only access to the views “catalog”, “seq_events”, “realtime_data” and “historical_data”.

See the file c:\oshmi\PostgreSQL\install.txt to access the definition and comments of the tables and views.

References:

https://grafana.com/

http://docs.grafana.org/

OSHMI JSON Protocol Driver Interface

This interface makes possible to feed data to OSHMI from other systems or protocol adapter drivers. Also commands can be passed from OSHMI to these other systems. For example an IOT device can send data to and receive controls from OSHMI displays using simple JSON UDP messages.

See HMI.ini [JSON] section for configuration options of ports and endpoints.

Data update messages

Data messages are in pure text JSON format as:

```
[
{ "tag" : "SUB1-MEAS-KV",
  "value" : 123.45
},
{ "tag" : "SUB2-MEAS-KV",
  "value" : 678.3
},
{ "tag" : "SUB1-STATUS-XCBR",
  "value" : true
}
]
```

The same message can be delivered in a compact form as :

```
{ "SUB1-MEAS-KV": 123.45,
  "SUB2-MEAS-KV": 678.3,
  "SUB1-STATUS-XCBR": true }
```

The more verbose format admits the following parameters:

“tag” - string, identifier for the point

“point_key” - integer, numeric identifier for the point

“address” - physical address for the point

It must be specified one of the three “tag”, “point_key” or “address”.

“value” - number or bool, current value for the point. Analog values are floating point numbers and binary states are coded as booleans (true/false).

“failed” - boolean, specify the quality of the measurement (true=bad, false=good, when absent it is considered as good).

“timetag” - field time tag (unix timestamp).

“ms” - millisecond of the field time tag.

When the point received by OSHMI is not configured in the point_list.txt file, the point will be created in the real-time database on-the-fly.

It must be considered the network characteristics for UPD packet safe delivery. For local ethernet networks the packet MTU is normally 1500 bytes, so the JSON message size can be up to 1472 bytes.

For WAN networks, the MTU can be smaller, so it is considered a safe size of 508 bytes for the JSON message.

Data messages must be sent to OSHMI using port 9100.

Command Message

Command Messages are sent from OSHMI to a UDP listener on port 9101.

Command points must be configured in the point_list.txt file.

Command messages have complete information about the command to be executed (some information may be irrelevant to the application):

```
 { "tag": "SUB1-STATUS-XCBR-K",
   "point_key": 64158,
   "address": 64158,
   "rtu": 0,
   "asdu": 0,
   "sbo": true,
   "value": 0,
   "logic_val": false,
   "action": "Turn_Off"
}
```

“rtu” - integer – RTU or IED address or number identifier.

“asdu” - integer - Type of ASDU to be used in the protocol.

“sbo” - boolean – Select before operate option (true=use SBO).

“logic_val” - boolean – Value as boolean (true/false).

“action” - string - Desired action of command, textually.

Command Ack Message

Command Ack Messages can optionally be sent to OSHMI to confirm command acceptance by the protocol driver. This message is similar to a data message and is also sent to OSHMI UDP port 9100.

```
{ "tag" : "SUB1-STATUS-XCBR-K",
  "value" : true,
  "failed" : false
}
```

“tag”, “point_key” or “address”: repeat here the tag, point_key or address received in the command message.

“value”: repeat here the value received in the command message.

“failed”: a boolean that represents the acceptance of the command (“failed”=true: command not accepted, “failed”=false: command accepted.

Debugging JSON Messages

To debug UDP JSON messages in OSHMI, use the JSON button of webserver.exe and mark the “Log” checkbox to see text messages received and sent.

Examples of JSON UDP Messages using Windows Powershell

(From https://gist.github.com/PeteGoo/21a5ab7636786670e47c)

```
function Send-UdpDatagram
{
      Param ([string] $EndPoint,
      [int] $Port,
      [string] $Message)

      $IP = [System.Net.Dns]::GetHostAddresses($EndPoint)
      $Address = [System.Net.IPAddress]::Parse($IP)
      $EndPoints = New-Object System.Net.IPEndPoint($Address, $Port)
      $Socket = New-Object System.Net.Sockets.UDPClient
      $EncodedText = [Text.Encoding]::ASCII.GetBytes($Message)
      $SendMessage = $Socket.Send($EncodedText, $EncodedText.Length, $EndPoints)
      $Socket.Close()
}
Send-UdpDatagram -EndPoint "127.0.0.1" -Port 9100 -Message '[{"tag": "KOR1TR1-2MTWT",
"value": 17.445 }]'
```

To see command messages, use a UDP listener such as this:

(From https://github.com/sperner/PowerShell/blob/master/UdpServer.ps1)

```
param( $address="Any", $port=9101 )

try{
    $endpoint = new-object System.Net.IPEndPoint( [IPAddress]::$address, $port )
    $udpclient = new-object System.Net.Sockets.UdpClient $port
}
catch{
    throw $_
    exit -1
}
Write-Host "Press ESC to stop the udp server ..." -fore yellow
Write-Host ""
while( $true )
{
    if( $host.ui.RawUi.KeyAvailable )
    {
        $key = $host.ui.RawUI.ReadKey( "NoEcho,IncludeKeyUp,IncludeKeyDown" )
        if( $key.VirtualKeyCode -eq 27 )
        {   break   }
    }

    if( $udpclient.Available )
    {
        $content = $udpclient.Receive( [ref]$endpoint )
        Write-Host "$($endpoint.Address.IPAddressToString):$($endpoint.Port) $
([Text.Encoding]::ASCII.GetString($content))"
    }
}
$udpclient.Close( )
```

Securing OSHMI servers

OSHMI server can be set up for encrypted secure access just by configuring the Nginx server for HTTPS, using client certificates and optionally user authentication. This method of access is standards based so it is very safe (bank level crypto), it is also convenient and lightweight.

The detailed configuration of the Nginx server is out of scope from this document. Please consult the provided references and Nginx documentation. Look also for hints in the OSHMI Nginx default files (commented sections).

https://fardog.io/blog/2017/12/30/client-side-certificate-authentication-with-nginx/

https://arcweb.co/securing-websites-nginx-and-client-side-certificate-authentication-linux/

https://www.ssltrust.com.au/help/setup-guides/client-certificate-authentication

https://www.digitalocean.com/community/tutorials/how-to-set-up-password-authentication-with-nginx-on-ubuntu-14-04

https://www.digitalocean.com/community/tutorials/how-to-set-up-http-authentication-with-nginx-on-ubuntu-12-10

Client machines must have the client certificates correctly installed.

It is provided at c:\oshmi\etc\autoselectclientcert.reg a Windows registry entry template to allow for the Chromium browser to auto select Client Certificates. Edit them to point to your OSHMI servers and certificate issuer.