Install and configure the monitoring of EVN Jumping JIVE and IVS Seamless Auxiliary Data (operation, diagnostic, and analysis data)

— Alexander Neidhardt 2019/04/30 00:29

The operational data from IVS or other telescopes (e.g. the EVN) are interesting for operation, diagnostic, and analysis. Originally, the real-time data were sent for dedicated antennas by a NASA Field System PC process into an incoming folder using SCP. A new way uses HTTP pages generated by e-RemoteCtrl software on the NASA Field System PC. The central monitoring server fetches the HTML pages and searches values and tags included. The incoming data are managed by ZABBIX (and SysMon) to present them. A copy of the web page of each individual NASA Field System site is also accessible on the central monitoring server. A web archive can be used to store the data history over years (in individual formats) on the system monitoring PC.

A general explanation of the originally written white paper can be found here:

ivstaskforceseamlessauxiliarydata.pdf

0) Download and install data sender on the NASA Field System Computer

- Download archives:
  - The software can currently be downloaded from a dropbox folder: https://www.dropbox.com/sh/efxmrs9klbgfg20/AABFxHVHRfxik5vGn_-7_pNsa?dl=0
  - The software can also be downloaded from the NASA Field System FTP servers (the address will be published)
  - or: request the software from Wettzell Observatory (A. Neidhardt)
- You will get a tar-ball using the naming convention <year><month><day>_eremotectrl_deliverable.tar.gz (where <year><month><day> is the time tag of the current version) or <release>_eremotectrl_deliverable.tar.gz (where <release> is a static release number)
- Unpack the archive into /usr2/st as user "prog" (or user "root")

-
  ```
  cd /usr2/st
  gunzip <year><month><day>_eremotectrl_deliverable.tar.gz
  tar -xvf <year><month><day>_eremotectrl_deliverable.tar.gz
  ```

- You should get a new folder "eremotectrl_deliverable"
- Change into the new folder and build the code

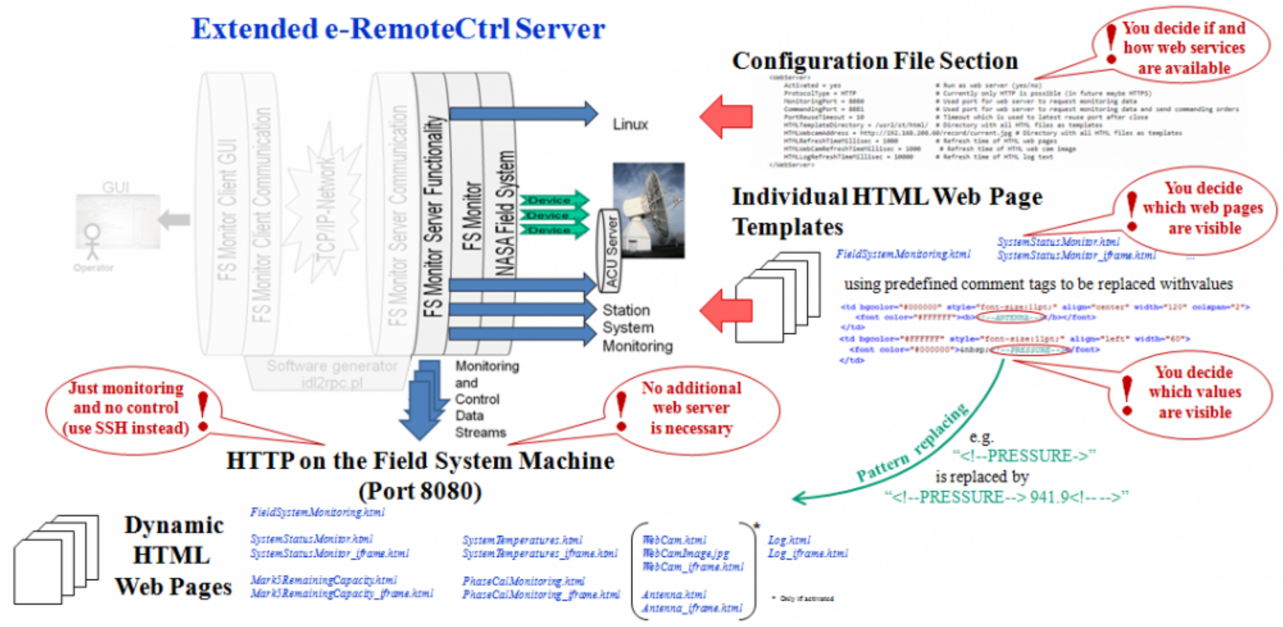
-
  ```
  cd /usr2/st/eremotectrl_deliverable/make
  make build
  su
  make install
  ```

```
exit
```

- After building, you can continue with the configuration.

1) The data sender on the NASA Field System Computer: e-RemoteCtrl software

- The e-RemoteCtrl Software was extended with a web server functionality. It is perfomant enough to server a few different users but cannot be compared to specialized web servers, like Apache. The server reads template files containing HTML code and key tags as comments. The key tags are predefined identifiers. The rest of the HTML structure can freely be changed.
- The main page is “FieldSystemMonitoring.html”. It contains a frame. All single frame patches consists of two files, e.g. “SystemStatusMonitor_iframe.html” and “SystemStatusMonitor.html”. The IFRAME-file is used to create a page which updates in predefined time intervals (e.g. every second) avoiding white flashing of different browsers during reload (e.g. Chrome, MS Internet Explorer, etc.). The second web page contains the real data and presentation structures. The following scheme gives an impression on how it works.


Extended e-RemoteCtrl Server
Configuration File Section
You decide if and how web services are available
Linux
GUI
Operator
FS Monitor Client GUI
FS Monitor Client Communication
TCP/IP-Network
FS Monitor Server Communication
FS Monitor Server Functionality
FS Monitor
NASA Field System
Device
Device
Device
ACU Server
Station System Monitoring
Software generator idl2rpc.pl
Monitoring and Control Data Streams
Individual HTML Web Page Templates
You decide which web pages are visible
FieldSystemMonitoring.html
SystemStatusMonitor.html
SystemStatusMonitor_iframe.html
using predefined comment tags to be replaced withvalues
You decide which values are visible
Just monitoring and no control (use SSH instead)
No additional web server is necessary
HTTP on the Field System Machine (Port 8080)
Pattern replacing
e.g. “<!--PRESSURE->” is replaced by “<!--PRESSURE--> 941.9<!---->”
Dynamic HTML Web Pages
FieldSystemMonitoring.html
SystemStatusMonitor.html
SystemStatusMonitor_iframe.html
Mark5RemainingCapacity.html
Mark5RemainingCapacity_iframe.html
SystemTemperatures.html
SystemTemperatures_iframe.html
PhaseCalMonitoring.html
PhaseCalMonitoring_iframe.html
WebCam.html
WebCamImage.jpg
WebCam_iframe.html
Antenna.html
Antenna_iframe.html
Log.html
Log_iframe.html
* Only if activated

- To enable the web server part, adapt the configuration file /usr2/st/eremotectrl_deliverable/config/eremotectrl.conf using an editor. The important parts are:

```
    <Telescope>
       IVSStationCode = XY
    </Telescope>
```

- and

```
    <WebServer>
       Activated = yes                          # Run as web server
(yes/no)
       ProtocolType = HTTP                      # Currently only HTTP
is possible (in future maybe HTTPS)
       MonitoringPort = 8080                    # Used port for web
```

```
server to request monitoring data
        CommandingPort = 8081                         # Used port for web
server to request monitoring data and send commanding orders
        PortReuseTimeout = 10                         # Timeout which is used
to latest reuse port after close
        HTMLTemplateDirectory = /usr2/st/html/  # Directory with all
HTML files as templates
        #HTMLWebcamAddress = http://webcam..../current.jpg # Directory
with all HTML files as templates
        HTMLRefreshTimeMillisec = 1000          # Refresh time of HTML
web pages
        HTMLWebCamRefreshTimeMillisec = 200     # Refresh time of HTML
web cam image
        HTMLLogRefreshTimeMillisec = 5000      # Refresh time of HTML
log text
    </WebServer>
```

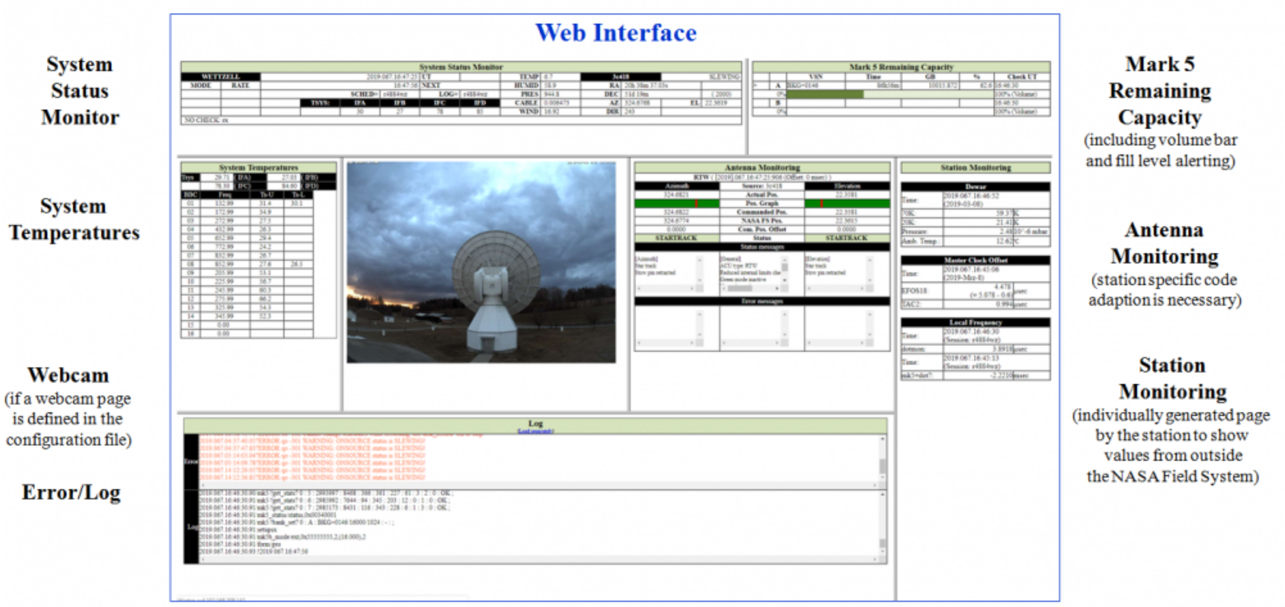
- Now you can start the server (Attention: Restricted prots like 80 can only be acquired by root processes, so that you must start the server as user "root").

-
  ```
   /usr2/st/eremotectrl_deliverable/bin/ercd
  /usr2/st/eremotectrl_deliverable/config/eremotectrl.conf
  ```

- If everything works, you should see some output lines in the shell.
- Create a Linux startup script /etc/init.d/ercd and set the automatic start during boot time

-
  ```
   sudo update-rc.d ercd defaults
  ```

- If you open a web browser and connect to the IP address of the NASA Field System adding the port defined in the configuration file (e.g. http://fspc:8080) you should see the remote control web GUI. Attention: SHTTP is currently not supported.

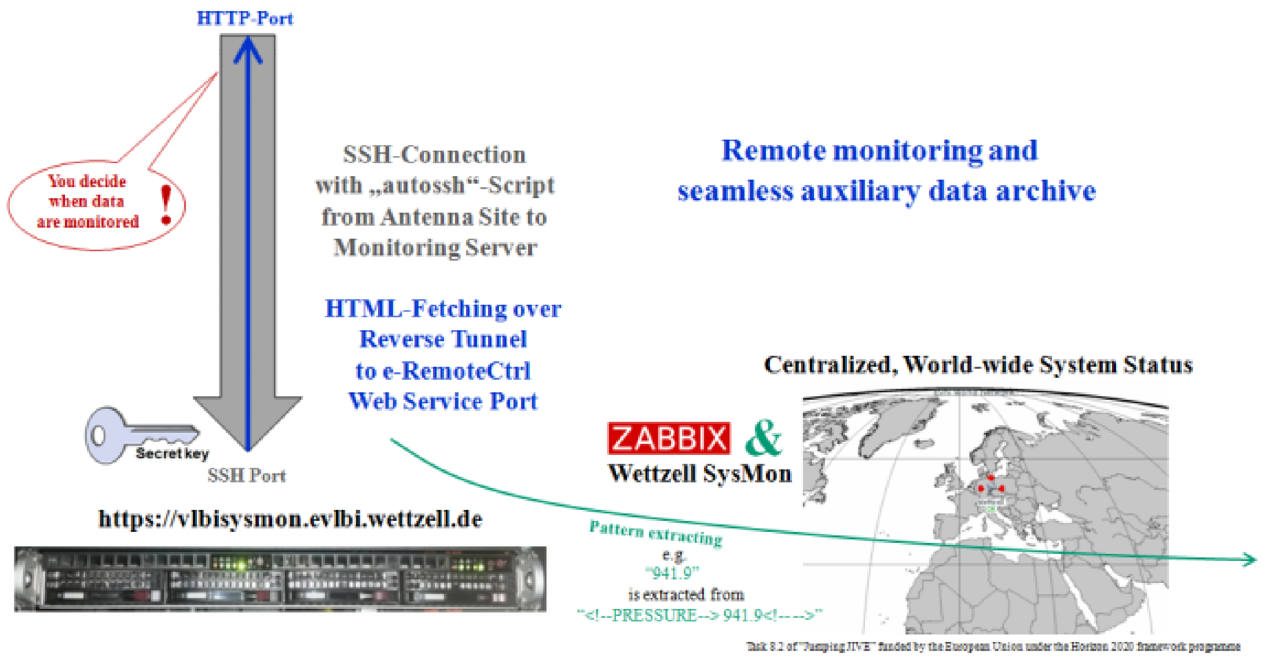

Web Interface
System Status Monitor
System Temperatures
Webcam (if a webcam page is defined in the configuration file)
Error/Log
Mark 5 Remaining Capacity (including volume bar and fill level alerting)
Antenna Monitoring (station specific code adaption is necessary)
Station Monitoring (individually generated page by the station to show values from outside the NASA Field System)

-
- Attention: The antenna monitoring can only be done if the station specific code of the e-RemoteCtrl software is programmed. An additional Phase Cal Monitoring (not shown in the figure above) is only filled with values if Phase Cal extraction is activated in the NASA Field System, so that the specific log lines can be extracted by the e-RemoteCtrl software (usually on

Mark4 systems).

2) Data transfer from the NASA Field System Computer to the Central Monitoring Server

- While the web pages are only availbale in the local network, they can also be used to support international services, like EVN, IVS, and IVS Task Force for Seamless Auxiliary Data. The web pages are fetched and copied to a central archive and are available on the Internet via an HTTPS URL (with or without password access). The web pages are not publicly available and are just used to support the services. Additionally, most of the parameters are tracked and monitored, so that graphs can be plotted from historic data.
- If you want to support these services please request an SSH key for the SSH connection, a port numbers for HTTP (and optionally ports for ZABBIX and SSH-oprin), and user account data from Wettzell observatory (A.Neidhardt).
- To enable a secure way of data exchange, the NASA Field System PC must open an SSH connection to the central monitoring server https://vlbisysmon.evlbi.wettzell.de. The SSH connection is used to open one reverse tunnel to the HTTP server port (and maybe others to a ZABBIX agent or for a remote control connection to the "oprin" program used to send commands to the NASA Field System; but currently only the monitoring is discussed here). The processing uses the following structure:


HTTP-Port
You decide when data are monitored !
SSH-Connection with „autossh"-Script from Antenna Site to Monitoring Server
HTML-Fetching over Reverse Tunnel to e-RemoteCtrl Web Service Port
Remote monitoring and seamless auxiliary data archive
Centralized, World-wide System Status
ZABBIX & Wettzell SysMon
Secret key
SSH Port
https://vlbisysmon.evlbi.wettzell.de
Pattern extracting
e.g. "941.9" is extracted from "<!--PRESSURE--> 941.9<!---->"
Task 8.2 of "Jumping JIVE" funded by the European Union under the Horizon 2020 framework programme

-
- The port number for HTTP are used as local endpoint of the reverse tunnel on the central monitoring server.
- Using the key file you can run the following script to start "autossh"
- Please take care that "autossh" is installed or install it if it is not yet available (as user "root"):

  -
    ```
    apt-get install autossh
    ```

-
  ```
  #!/bin/bash
  # This script starts and stops autossh if called from a Linux start script
  # ./autossh_run.sh <ssh_key> <list_of_reverse_tunnels>
  ```

```
<ssh_server_address>
# e.g.
# ./autossh_run.sh key -R22224:127.0.0.1:22 -R8084:127.0.0.1:8080
test.ip

ME=`basename "$0"`
ABSOLUTE_PATH_TO_ME="$(cd "$(dirname "${BASH_SOURCE[0]}")" &&
pwd)/$(basename "${BASH_SOURCE[0]}")"
ARGS=("$@")
ssh_server=${ARGS[-1]}
ssh_key=${ARGS[0]}
USER="oper"

arguments=""
for arg in "$@"
do
    arguments="$arguments $arg"
done

if [ $# -lt 2 ] ; then
    echo "./autossh_run.sh <key-file> [<ssh-arguments>]{+} <ssh-
server>"
    echo "     <key-file>          : keyfile including path
(mandatory)"
    echo "     <ssh-arguments>     : optional SSH argument list, e.g.
port forwarding string like -R2222:127.0.0.1:22"
    echo "     <ssh-server>        : IP-address/-alias of SSH server
(mandatory)"
    exit 1
fi

sigkill()
{
    echo -e "$ME: Received signal KILL/QUIT/TERM/INT"
    mypid=`ps ax | grep $ssh_server | grep autossh  | grep -v 'grep' |
grep -v $ME | awk '{print $1}'`
    number_of_found_processes=`ps ax | grep $ssh_server | grep autossh
| grep -v 'grep' | grep -v $ME | awk '{print $1}' | wc -w`
    if [ $number_of_found_processes -gt 0 ] ; then
        kill $mypid
    fi
    exit
}

# Define signal handler
trap 'sigkill' KILL
trap 'sigkill' QUIT
trap 'sigkill' TERM
trap 'sigkill' INT
```

```
# Start program
/usr/lib/autossh/autossh -M 0 -o "ServerAliveInterval 30" -o
"ServerAliveCountMax 3" -N -T -X -l ${USER} -i $arguments &
COMMAND="echo ${arguments} | grep -o -E 'R[0-9]+' | grep -o -E '[0-9]+'
| head -1"
CHECKPORT=`eval $COMMAND`
COMMAND="date +%s"
STARTTIME_SEC=`eval $COMMAND`
while [ 1 ] ; do
    if [[ -n $CHECKPORT ]]; then
        COMMAND="/usr/bin/timeout 21 /usr/bin/ssh -o ConnectTimeout=20
-l ${USER} -i ${ssh_key}  ${ssh_server} 'netstat -lntu | grep
${CHECKPORT}'"
        PORTCHECK_RESULT=`eval $COMMAND`
        COMMAND="date +%s"
        CHECKTIME_SEC=`eval $COMMAND`
        COMMAND="expr ${CHECKTIME_SEC} - ${STARTTIME_SEC}"
        TIMEINTERVAL_SEC=`eval $COMMAND`
       if [[ -z $PORTCHECK_RESULT ]]; then
            if [ "$TIMEINTERVAL_SEC" -gt "20" ]; then
                echo "autossh is not running"
                COMMAND="ps ax | grep autossh | grep -v grep | grep -v
${ME} | grep ${CHECKPORT} | grep -o -E '^[ ]*[0-9]+'"
                AUTOSSH_PID=`eval $COMMAND`
                if [[ -n $AUTOSSH_PID ]]; then
                    /bin/kill -9 $AUTOSSH_PID
                fi
                COMMAND="ps ax | grep ssh | grep -v grep | grep -v
${ME} | grep ${CHECKPORT} | grep -o -E '^[ ]*[0-9]+'"
                SSH_PID=`eval $COMMAND`
                if [[ -n $SSH_PID ]]; then
                    /bin/kill -9 $SSH_PID
                fi
                echo "=> restart"
                /usr/lib/autossh/autossh -M 0 -o "ServerAliveInterval
30" -o "ServerAliveCountMax 3" -N -T -X -l ${USER} -i $arguments &
                COMMAND="date +%s"
                STARTTIME_SEC=`eval $COMMAND`
            fi
        else
            COMMAND="date +%s"
            STARTTIME_SEC=`eval $COMMAND`
        fi
    fi
    sleep 1
done
```

- Start the script with the following argument list

-
  ```
  autossh_run.sh ./<keyfile> -R<port>:127.0.0.1:8080
  ```

```
vlbisysmon.evlbi.wettzell.de
```

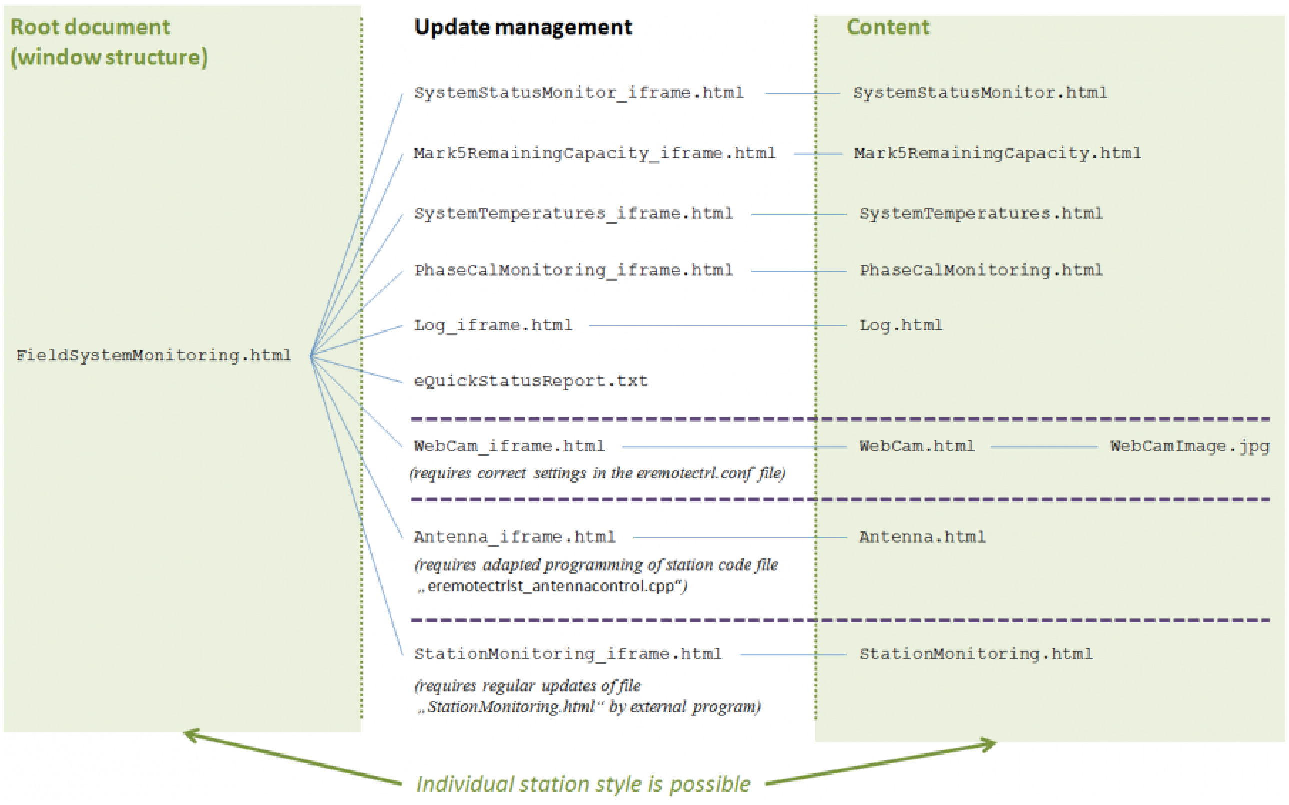
- Replace <keyfile> with the path to the received key file and <port> with the received port number
- Create a Linux startup script /etc/init.d/autossh_run and set the automatic start during boot time

-
  ```
  sudo update-rc.d autossh_run defaults
  ```

- Inform the Wettzell observatory about the opened tunnel. The rest is done on the central monitoring server.

3) Used data formats for the transfer

- The web server supports the following pages (Original: 20180726webpagestructure.pptx)



Root document (window structure)
Update management
Content
FieldSystemMonitoring.html
SystemStatusMonitor_iframe.html
SystemStatusMonitor.html
Mark5RemainingCapacity_iframe.html
Mark5RemainingCapacity.html
SystemTemperatures_iframe.html
SystemTemperatures.html
PhaseCalMonitoring_iframe.html
PhaseCalMonitoring.html
Log_iframe.html
Log.html
eQuickStatusReport.txt
WebCam_iframe.html
WebCam.html
WebCamImage.jpg
(requires correct settings in the eremotectrl.conf file)
Antenna_iframe.html
Antenna.html
(requires adapted programming of station code file „eremotectrlst_antennacontrol.cpp“)
StationMonitoring_iframe.html
StationMonitoring.html
(requires regular updates of file „StationMonitoring.html“ by external program)
Individual station style is possible

-
- The root document “FieldSystemMonitoring.html” contains frames and defines the general structure. It is not necessary to support all related pages. Missing pages are ignored. It might look like this:

    -
      ```
      <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
      "http://www.w3.org/TR/html4/loose.dtd">
      <HTML>
          <FRAMESET rows="18%,47%,35%">
              <FRAMESET cols="70%,30%">
                  <FRAME src="SystemStatusMonitor_iframe.html"
      name="SystemStatusMonitor" scrolling="no">
                  <FRAME src="Mark5RemainingCapacity_iframe.html"
      ```

```
name="Mark5RemainingCapacity" scrolling="no">
        </FRAMESET>
        <FRAMESET cols="20%,15%,35%,30%">
            <FRAME src="SystemTemperatures_iframe.html"
name="SystemTemperatures" scrolling="no">
            <FRAME src="PhaseCalMonitoring_iframe.html"
name="PhaseCalMonitoring" scrolling="no">
            <FRAME src="WebCam_iframe.html" name="WebCam_iframe"
scrolling="no">
            <FRAME src="Antenna_iframe.html" name="Antenna_iframe"
scrolling="no">
        </FRAMESET>
        <frame src="Log_iframe.html" name="Log_iframe"
scrolling="no">
    </FRAMESET>
</HTML>
```

- The IFRAME pages for the update management use Javascript to update related content documents regularly. They have a special timing to avoid the flashing during the update of web contents in browsers like Chrome, MS Internet Explorer, etc. (Firefox automatically avoid the white fading). The IFRAME code looks like this:

  -
    ```
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
    <HTML>
      <HEAD>
        <SCRIPT>
          var site = "Mark5RemainingCapacity.html";
          var ReloadWebpageStarted = 0;
          function load_webpage_foreground () {
             document.getElementsByName('IFRAME_FOREGROUND')[0].src =
    site;
          }
          function load_webpage_background () {
             document.getElementsByName('IFRAME_BACKGROUND')[0].src =
    site;
          }
          function trigger_load_webpage_foreground () {
             window.setTimeout("load_webpage_foreground ()", <!--
    ERC::SAMPLING_REFRESHTIME_MSEC-->);
          }
          function reload_webpage ()
          {
              if (ReloadWebpageStarted == 1)
              {
                  location.reload();
              }
              ReloadWebpageStarted = 1;
              load_webpage_background ();
              window.setTimeout("reload_webpage()", 600000); // Relaod
    ```

```
every 10 minutes
      }
    </SCRIPT>
  </HEAD>
  <BODY onload="reload_webpage ()" >
    <DIV style="position: absolute; top: 1px; left: 1px; z-index:
1; width: 100%; height: 100%">
      <IFRAME name="IFRAME_BACKGROUND" id="IFRAME_BACKGROUND"
             scrolling="no"
             frameborder="0"
             height="100%" width="100%"
             style="visibility: visible;"
             onload="trigger_load_webpage_foreground()">
      </IFRAME>
    </DIV>
    <DIV style="position: absolute; top: 1px; left: 1px; z-index:
2; width: 100%; height: 100%">
      <IFRAME name="IFRAME_FOREGROUND" id="IFRAME_FOREGROUND"
             scrolling="no"
             frameborder="0"
             height="100%" width="100%"
             style="visibility: visible;"
             onload="load_webpage_background()">
      </IFRAME>
    </DIV>
  </BODY>
</HTML>
```

- The IFRAME documents for the update management support the following replacement key tags, which are automatically replaced with values from the NASA Field System or configuration file by the eremotectrl server:
  - **<!–ERC::REFRESHTIME_MSEC–>** - Is replaced by the “HTMLRefreshTimeMillisec” (including a randomized jitter) from the configuration file “eremotectrtl.conf”; standard refresh time
  - **<!–ERC::SAMPLING_REFRESHTIME_MSEC–>** - Is replaced by the “HTMLRefreshTimeMillisec” / 2 (including a randomized jitter) to guarantee the update sampling time “HTMLRefreshTimeMillisec” from the configuration file “eremotectrtl.conf”; standard sampling time (suggested)
  - **<!–ERC::LOG_REFRESHTIME_MSEC–>** - Is replaced by the “HTMLLogRefreshTimeMillisec” (including a randomized jitter) from the configuration file “eremotectrtl.conf”; log refresh time (suggested)
  - **<!–ERC::LOG_SAMPLING_REFRESHTIME_MSEC–>**- Is replaced by the “HTMLLogRefreshTimeMillisec” / 2 (including a randomized jitter) to guarantee the update sampling time “HTMLLogRefreshTimeMillisec” from the configuration file “eremotectrtl.conf”; log sampling time
  - **<!–ERC::WEBCAM_REFRESHTIME_MSEC–>** - Is replaced by the “HTMLWebCamRefreshTimeMillisec” (including a randomized jitter) from the configuration file “eremotectrtl.conf”; webcam refresh time
  - **<!–ERC::WEBCAM_SAMPLING_REFRESHTIME_MSEC–>** - Is replaced by the “HTMLWebCamRefreshTimeMillisec” / 2 (including a randomized jitter) to guarantee the update sampling time “HTMLWebCamRefreshTimeMillisec” from the

configuration file “eremotectrtl.conf”; webcam sampling time (suggested)

- The HTML content documents can be designed individually. They support the following replacement key tags (tags are not document related and can appear in any web page where they are also replaced; but ZABBIX scripts use just the firtst hit of a pattern to extract values; therefore, usually <!–ERC::ANTENNA–>, for example, appears in the page for the System Status Monitor and so on), which are automatically replaced with values from the NASA Field System by the eremotectrl server. The tags are not obligatory. Not defined tags are ignored.
    - **<!–ERC::TIMECOVERED–>** - Should be included to each web page. It is replaced by the time tag from the NASA Field System and used as time-tagging for the ZABBIX/SysMon monitoring. The value of this tag is within a HTML comment and not visible.
    - **System Status Monitor:**
        - **<!–ERC::ANTENNA–>** - Antenna name
        - **<!–ERC::TIME–>** - Current NASA Field System time
        - **<!–ERC::HALT–>** - Scheduleing halt status
        - **<!–ERC::TEMPERATURE–>** - Meteorological temperature in degree Celcius
        - **<!–ERC::SOURCE–>** - Current source
        - **<!–ERC::TRACKINGSTATE–>** - Current tracking state (tracking/slewing)
        - **<!–ERC::NEXTTIME–>** - Time of next activity
        - **<!–ERC::HUMIDITY–>** - Meteorological humidity in percent
        - **<!–ERC::RA–>** - Right ascension of current source
        - **<!–ERC::MODE–>** - Mode
        - **<!–ERC::RATE–>** - Rate
        - **<!–ERC::SCHEDULE–>** - Current schedule
        - **<!–ERC::LOG–>** - current log file name
        - **<!–ERC::PRESSURE–>** - Meteorological pressure in hPas
        - **<!–ERC::DEC–>** - Declination of current source
        - **<!–ERC::CATALOGYEAR–>** - Year of astronomical catalog used
        - **<!–ERC::NAME_IF1–>** - Name of the intermediate frequency (IF) channel 1
        - **<!–ERC::NAME_IF2–>** - Name of the intermediate frequency (IF) channel 2
        - **<!–ERC::NAME_IF3–>** - Name of the intermediate frequency (IF) channel 3
        - **<!–ERC::NAME_IF4–>** - Name of the intermediate frequency (IF) channel 4
        - **<!–ERC::CABLE–>** - Cable value
        - **<!–ERC::AZIMUTH–>** - Azimuth of current source
        - **<!–ERC::ELEVATION–>** - Elevation of current source
        - **<!–ERC::IF1–>** - Value of system temperatur of IF channel 1
        - **<!–ERC::IF2–>** - Value of system temperatur of IF channel 2
        - **<!–ERC::IF3–>** - Value of system temperatur of IF channel 3
        - **<!–ERC::IF4–>** - Value of system temperatur of IF channel 4
        - **<!–ERC::WINDSPEED–>** - Wind speed in km/h
        - **<!–ERC::WINDDIRECTION–>** - Wind direction in degree
        - **<!–ERC::CHECKS–>** - Activated NASA Field System checks
    - **Mark 5 Remaining Capacity**
        - **<!–ERC::SELECTED_MODULEA–>** - Marker to show that module A is used
        - **<!–ERC::VSN_MODULEA–>** - VSN number of module A
        - **<!–ERC::TIME_MODULEA–>** - Remaining recording time with the current setting of module A
        - **<!–ERC::REMAININGGB_MODULEA–>** - Remaining volume in GByte setting of module A
        - **<!–ERC::REMAININGPERCENT_MODULEA–>** - Remaining volume in percent setting of module A

    - **<!–ERC::CHECKTIME_MODULEA–>** - Time of latest check setting of module A
    - **<!–ERC::REMAININGPERCENT_FILLGRAPH_MODULEA–>** - Bar graph showing the fill state of module A
    - **<!–ERC::SELECTED_MODULEB–>** - Marker to show that module B is used
    - **<!–ERC::VSN_MODULEB–>** - VSN number of module B
    - **<!–ERC::TIME_MODULEB–>** - Remaining recording time with the current setting of module B
    - **<!–ERC::REMAININGGB_MODULEB–>** - Remaining volume in GByte setting of module B
    - **<!–ERC::REMAININGPERCENT_MODULEB–>** - Remaining volume in percent setting of module B
    - **<!–ERC::CHECKTIME_MODULEB–>** - Time of latest check setting of module A
    - **<!–ERC::REMAININGPERCENT_FILLGRAPH_MODULEB–>** - Bar graph showing the fill state of module B
  - **System Temperatures**
    - **<!–ERC::TSYS_IF1–>** - Value of system temperatur of IF channel 1
    - **<!–ERC::NAME_IF1–>** - Name of the IF channel 1
    - **<!–ERC::TSYS_IF2–>** - Value of system temperatur of IF channel 2
    - **<!–ERC::NAME_IF2–>** - Name of the IF channel 2
    - **<!–ERC::TSYS_IF3–>** - Value of system temperatur of IF channel 3
    - **<!–ERC::NAME_IF3–>** - Name of the IF channel 3
    - **<!–ERC::TSYS_IF4–>** - Value of system temperatur of IF channel 4
    - **<!–ERC::NAME_IF4–>** - Name of the IF channel 4
    - **<!–ERC::CONVERTER_TYPE–>** - Baseband converter type
    - **<!–ERC::FREQUENCY_CH01–>** to **<!–ERC::FREQUENCY_CH16–>** - Current frequency of channel 1 to 16
    - **<!–ERC::TSYS_UPPER_CH01–>** to **<!–ERC::TSYS_UPPER_CH16–>** - Current system temperature of upper channel 1 to 16
    - **<!–ERC::TSYS_LOWER_CH01–>** to **<!–ERC::TSYS_LOWER_CH16–>** - Current system temperature of lower channel 1 to 16
  - **Phase Cal Monitoring**
    - **<!–ERC::AMPLITUDE_CH01–>** to **<!–ERC::AMPLITUDE_CH16–>** - Phase calibration amplitude of channel 1 to 16
    - **<!–ERC::PHASE_CH01–>** to **<!–ERC::PHASE_CH16–>** - Phase calibration phase of channel 1 to 16
    - **<!–ERC::TIME_CH01–>** to **<!–ERC::TIME_CH16–>** - Time of latest check of phase calibration of channel 1 to 16
  - **Antenna Monitoring** (if the station programming in the eremotectrl server should be avoided, Antenna Monitoring can be replaced by the Station Monitoring considering the inclusion of the replacement tags of the Antenna Monitoring in the way, described for Station Monitoring)
    - **<!–ERC::ACU_ANTENNA_NAME–>** - Antenna name in the antenna control unit
    - **<!–ERC::ACU_ANTENNA_TIME–>** - Antenna time of the antenna control unit
    - **<!–ERC::ACU_SOURCE–>** - Source in the antenna control unit
    - **<!–ERC::ACU_AZIMUTH_POSITION–>** - Actual azimuth position in the antenna control unit
    - **<!–ERC::ACU_ELEVATION_POSITION–>** - Actual elevation position in the antenna control unit
    - **<!–ERC::ACU_AZIMUTH_POSITION_GRAPH–>** - Position graph for azimuth position in the antenna control unit (using the NASA Field System control file

        "antenna.ctl" for limits)
        - **<!-ERC::ACU_ELEVATION_POSITION_GRAPH->** - Position graph for elevation position in the antenna control unit (using the NASA Field System control file "antenna.ctl" for limits)
        - **<!-ERC::ACU_AZIMUTH_POSITION_COMMANDED->** - Commanded azimuth position in the antenna control unit
        - **<!-ERC::ACU_ELEVATION_POSITION_COMMANDED->** - Commanded elevation position in the antenna control unit
        - **<!-ERC::ACU_AZIMUTH_POSITION_NASAFS->** - Current azimuth position in the NASA Field System
        - **<!-ERC::ACU_ELEVATION_POSITION_NASAFS->** - Current elevation position in the NASA Field System
        - **<!-ERC::ACU_AZIMUTH_POSITION_OFFSET->** - Azimuth offset for offset pointing in the antenna control unit
        - **<!-ERC::ACU_ELEVATION_POSITION_OFFSET->** - Elevation offset for offset pointing in the antenna control unit
        - **<!-ERC::ACU_AZIMUTH_STATUS->** - Status of azimuth axis in the antenna control unit
        - **<!-ERC::ACU_ELEVATION_STATUS->** - Status of elevation axis in the antenna control unit
        - **<!-ERC::ACU_AZIMUTH_STATUS_MSG->** - List of azimuth status messages in the antenna control unit
        - **<!-ERC::ACU_GENERAL_STATUS_MSG->** - List of general status messages in the antenna control unit
        - **<!-ERC::ACU_ELEVATION_STATUS_MSG->** - List of elevation status messages in the antenna control unit
        - **<!-ERC::ACU_AZIMUTH_ERROR_MSG->** - List of azimuth error messages in the antenna control unit
        - **<!-ERC::ACU_GENERAL_ERROR_MSG->** - List of general error messages in the antenna control unit
        - **<!-ERC::ACU_ELEVATION_ERROR_MSG->** - List of elevation error messages in the antenna control unit
    - **Log**
        - **<!-ERC::ERROR_MSG->**
        - **<!-ERC::LOG_MSG->**
    - **Station Monitoring**
        - The Station Monitoring is not managed in eremotectrl and just published by the web server service.
        - There are no predefined station replacement tags. They can be defined individually but should have the following structure:
            - **<!-REPLACEMENT_TAG-> [Value] <!- ->**,
            - where *REPLACEMENT_TAG* is an individual name including the antenna identifier and *[Value]* is the value.
        - If these values should be included to Zabbix/SysMon, it is necessary to do the integration in the host by station staff. An explanation is not yet part of this description.
- The web server of the e-RemoteCtrl software also supports the real-time format for IVS Live:

    -
      ```
      <eQuickStatusInfo>
          Service = <!--ERC::SERVICE_NETWORK-->
      ```

```
    Stationname = <!--ERC::ANTENNA-->
    StationIVSCode = <!--ERC::STATION_LETTERCODE-->
    Schedule = <!--ERC::SCHEDULE-->
    Status = <!--ERC::SYSTEMSTATUS-->
    DateTime = <!--ERC::TIME-->
    TimeNext = <!--ERC::NEXTTIME-->
    Source = <!--ERC::SOURCE-->
    Scan = <!--ERC::SCANNAME-->
    Mark5VSN = <!--ERC::MARK5_VSN-->
    Mark5Volume = <!--ERC::MARK5_CAPACITY-->
    Mark5Used = <!--ERC::MARK5_USEDCAPACITY-->
    RightAscension = <!--ERC::RA-->
    Declination = <!--ERC::DEC-->
    Azimuth = <!--ERC::AZIMUTH-->
    Elevation = <!--ERC::ELEVATION-->
    CableDelay = <!--ERC::CABLE-->
    SystemTemperatureIFA = <!--ERC::IF1-->
    SystemTemperatureIFB = <!--ERC::IF2-->
    SystemTemperatureIFC = <!--ERC::IF3-->
    SystemTemperatureIFD = <!--ERC::IF4-->
    MeteorologyTemperature = <!--ERC::TEMPERATURE-->
    MeteorologyHumidity =  <!--ERC::HUMIDITY-->
    MeteorologyPressure =  <!--ERC::PRESSURE-->
</eQuickStatusInfo>
```

- Values are automatically extracted from the web pages at the central monitoring server.

4) Processing of web pages on the Central Monitoring Server

- If you received and installed all components, the administrator of the central web server will open the accounts. The following operations are then automated on the central web server by the administrator and you will get feedback about.
- Required is only a direct connection between the NASA Field System PC and the central monitoring server. As all web pages are copied (usually every second; can be adapted accordingly to your wishes), there will be **no direct access from other PCs** in the Internet to your local NASA Field System PC. External requests get the copy and are not forwarded. Additionally, only the owner of the requested account with username and password has access to the web pages. All Field System web pages will be stored just for 1 second and are overwritten by the updated pages (depending on the Internet connection). Historic data are stored for 90 days and are then deleted by the ZABBIX house keeping process. But finally, station staff is responsible to follow all regulations and law valid for its country when sending data (e.g. for Webcams etc., avoiding humans on the images). In case of a misues in the sense of German law, the service will be stopped and all data will be deleted. If you would need a legal aggreement according to the General Data Protection Regulation (GDPR) of the European Union (https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679), please inform when requesting an account, so that it can be discussed.
- The processing on the central monitoring server is done by the administrator of the server. The following parts are prepared by him.
- It is necessary to add an additional crontab line using the following commands (to use editor “vi” for editing)

-
  ```
  export EDITOR=vi
  crontab -e
  ```
- The line looks like this (use the correct antenna name and port)

-
  ```
  *    *    *    *    *
  /home/oper/Software/vlbisysmon/bin/fetch_fs_webpage.sh wettzell 8083
  *    *    *    *    *
  /home/oper/Software/vlbisysmon/bin/fetch_fs_webpage.sh wettz13n 8084
  *    *    *    *    *
  /home/oper/Software/vlbisysmon/bin/fetch_fs_webpage.sh wettz13s 8085
  ```
- It calls the script “/home/oper/Software/vlbisysmon/bin/fetch_fs_webpage.sh” which starts the fetching of web page content in the background each time the process is not yet active. The script looks like this:

-
  ```
  SERVICE='fetch_fs_webpage_single.sh'
  PROGPATH='/home/oper/Software/vlbisysmon/bin/'

  if [ $# -lt 2 ]; then
      /bin/echo 0.0
      exit 1
  fi

  AntennaName=$1      # e.g. "wettz13s"
  LocalTunnelPort=$2  # e.g. "8085"

  SERVICETEST="$SERVICE $AntennaName"

  #/bin/sleep 1
  if /bin/ps ax | /bin/grep -v grep | /bin/grep "/bin/bash" | /bin/grep
  "$SERVICETEST" > /dev/null
  then
      /bin/echo 1.0
      #echo "$SERVICE is already running => do not call again"
      exit 1
  else
      /bin/echo 2.0
      #echo "$SERVICE is not running => call it"
      ${PROGPATH}/${SERVICE} "${AntennaName}" "${LocalTunnelPort}" &
  fi

  exit 0
  ```
- The real script fetching the web page content is “/home/oper/Software/vlbisysmon/bin/fetch_fs_webpage_single.sh”, which looks like this:

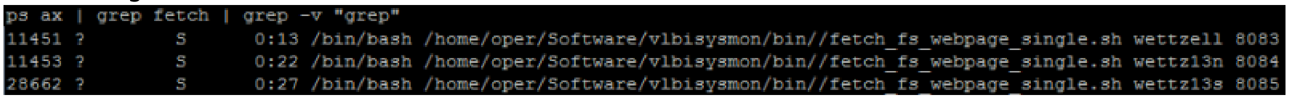
-
  ```
  MONITORING_ARCHIVE_PATH="/var/www/html/monitoring_archive/"
  WEBPAGES_PATH="${MONITORING_ARCHIVE_PATH}fs_web_pages/"
  VERBOSE_ON=1
  ```

```
if [ $# -lt 2 ]; then
    /bin/echo 0.0
    exit 1
fi

AntennaName=$1      # e.g. "wettz13s"
LocalTunnelPort=$2  # e.g. "8085"

echo $WEBPAGES_PATH

while [[ 1 -eq 1 ]] ; do

    StartTime=`/bin/date +%s.%N`
    StartTimeSeconds=`/bin/date +%s`
    WebPageDirectory=`/bin/echo $WEBPAGES_PATH/${AntennaName}`
    TimeOfLastCompleteUpdate=""
    FetchFrameFiles=0

    # Check directory path and create directories if not exist
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Check directiories and create them if they are not
available"
    fi
    if [ ! -d $MONITORING_ARCHIVE_PATH ]; then
        /bin/echo 0.0
        exit 1
    fi
    if [ ! -d $WEBPAGES_PATH ]; then
        /bin/mkdir $WEBPAGES_PATH
        /bin/chmod 777 $WEBPAGES_PATH
        FetchFrameFiles=1
    fi
    if [ ! -d $WEBPAGES_PATH${AntennaName} ]; then
        /bin/mkdir ${WEBPAGES_PATH}${AntennaName}
        /bin/chmod 777 ${WEBPAGES_PATH}${AntennaName}
        FetchFrameFiles=1
    fi

    if [ ! -f ${WebPageDirectory}/TimeOfLastCompleteUpdate.txt ]; then
        FetchFrameFiles=1
    fi

    # Check if complete update of all files should be done
    if [ "$FetchFrameFiles" -eq "0" ]; then
        TimeOfLastCompleteUpdate=`cat
${WebPageDirectory}/TimeOfLastCompleteUpdate.txt`
        if [ `/bin/echo "$StartTimeSeconds - $TimeOfLastCompleteUpdate"
| bc` -gt 600 ] ; then
           FetchFrameFiles=1
        fi
    fi

    # = Main frame web page
==============================================
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Fetch FieldSystemMonitoring.html"
    fi
    if [ "$FetchFrameFiles" -eq "1" -o ! -f
${WebPageDirectory}/FieldSystemMonitoring.html ]; then
        /usr/bin/wget http://127.0.0.1:${LocalTunnelPort}/ -q -O
${WebPageDirectory}/FieldSystemMonitoring.ht
ml.tmp
        /bin/mv ${WebPageDirectory}/FieldSystemMonitoring.html.tmp
${WebPageDirectory}/FieldSystemMonitoring.
html
    fi

    # = System Status Monitor
============================================
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Fetch SystemStatusMonitor(_iframe).html"
    fi
    if [ "$FetchFrameFiles" -eq "1" -o ! -f
${WebPageDirectory}/SystemStatusMonitor_iframe.html ]; then
        /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/SystemStatusMonitor_iframe.html -q
-O ${WebPageDire
ctory}/SystemStatusMonitor_iframe.html.tmp
        /bin/mv ${WebPageDirectory}/SystemStatusMonitor_iframe.html.tmp
${WebPageDirectory}/SystemStatusMonit
or_iframe.html
    fi
    /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/SystemStatusMonitor.html -q -O
${WebPageDirectory}/Syst
emStatusMonitor.html.tmp
    /bin/mv ${WebPageDirectory}/SystemStatusMonitor.html.tmp
${WebPageDirectory}/SystemStatusMonitor.html

    # = Mark 5 Remaining Capacity
========================================
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Fetch Mark5RemainingCapacity(_iframe).html"
    fi
    if [ "$FetchFrameFiles" -eq "1" -o ! -f
${WebPageDirectory}/Mark5RemainingCapacity_iframe.html ]; then
        /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/Mark5RemainingCapacity_iframe.html
-q -O ${WebPageD
irectory}/Mark5RemainingCapacity_iframe.html.tmp
```

```
        /bin/mv
${WebPageDirectory}/Mark5RemainingCapacity_iframe.html.tmp
${WebPageDirectory}/Mark5Remaining
Capacity_iframe.html
    fi
    /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/Mark5RemainingCapacity.html -q -O
${WebPageDirectory}/M
ark5RemainingCapacity.html.tmp
    /bin/mv ${WebPageDirectory}/Mark5RemainingCapacity.html.tmp
${WebPageDirectory}/Mark5RemainingCapacity.ht
ml

    # = System Temperatures
==============================================
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Fetch SystemTemperatures(_iframe).html"
    fi
    if [ "$FetchFrameFiles" -eq "1" -o ! -f
${WebPageDirectory}/SystemTemperatures_iframe.html ]; then
        /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/SystemTemperatures_iframe.html -q -
O ${WebPageDirec
tory}/SystemTemperatures_iframe.html.tmp
        /bin/mv ${WebPageDirectory}/SystemTemperatures_iframe.html.tmp
${WebPageDirectory}/SystemTemperatures
_iframe.html
    fi
    /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/SystemTemperatures.html -q -O
${WebPageDirectory}/Syste
mTemperatures.html.tmp
    /bin/mv ${WebPageDirectory}/SystemTemperatures.html.tmp
${WebPageDirectory}/SystemTemperatures.html

    # = Phase Cal Monitoring
=============================================
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Fetch PhaseCalMonitoring(_iframe).html"
    fi
    if [ "$FetchFrameFiles" -eq "1" -o ! -f
${WebPageDirectory}/PhaseCalMonitoring_iframe.html ]; then
        /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/PhaseCalMonitoring_iframe.html -q -
O ${WebPageDirec
tory}/PhaseCalMonitoring_iframe.html.tmp
        /bin/mv ${WebPageDirectory}/PhaseCalMonitoring_iframe.html.tmp
${WebPageDirectory}/PhaseCalMonitoring
_iframe.html
    fi
    /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/PhaseCalMonitoring.html -q -O
${WebPageDirectory}/Phase
CalMonitoring.html.tmp
    /bin/mv ${WebPageDirectory}/PhaseCalMonitoring.html.tmp
${WebPageDirectory}/PhaseCalMonitoring.html

    # = Web Cam
==========================================================
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Fetch WebCam(_iframe).html and WebCamImage.jpg"
    fi
    if [ "$FetchFrameFiles" -eq "1" -o ! -f
${WebPageDirectory}/WebCam_iframe.html ]; then
        /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/WebCam_iframe.html -q -O
${WebPageDirectory}/WebCam
_iframe.html.tmp
        /bin/mv ${WebPageDirectory}/WebCam_iframe.html.tmp
${WebPageDirectory}/WebCam_iframe.html
    fi
    if [ "$FetchFrameFiles" -eq "1" -o ! -f
${WebPageDirectory}/WebCam.html ]; then
        /usr/bin/wget http://127.0.0.1:${LocalTunnelPort}/WebCam.html -
q -O ${WebPageDirectory}/WebCam.html.t
mp
        /bin/mv ${WebPageDirectory}/WebCam.html.tmp
${WebPageDirectory}/WebCam.html
    fi
    WebCamImageFile=`/bin/sed -n -r 's/.*(WebCamImage\..*)\"\s+.*/\1/p'
${WebPageDirectory}/WebCam.html`
    /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/${WebCamImageFile} -q -O
${WebPageDirectory}/${WebCamIm
ageFile}.tmp
    /usr/bin/convert ${WebPageDirectory}/${WebCamImageFile}.tmp -resize
400x400 ${WebPageDirectory}/${WebCamI
mageFile}.tmp2
    /bin/mv ${WebPageDirectory}/${WebCamImageFile}.tmp2
${WebPageDirectory}/${WebCamImageFile}
    /bin/rm ${WebPageDirectory}/${WebCamImageFile}.tmp

    # = Antenna
==========================================================
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Fetch Antenna(_iframe).html"
    fi
    if [ "$FetchFrameFiles" -eq "1" -o ! -f
${WebPageDirectory}/Antenna_iframe.html ]; then
        /usr/bin/wget
```

```
http://127.0.0.1:${LocalTunnelPort}/Antenna_iframe.html -q -O
${WebPageDirectory}/Anten
na_iframe.html.tmp
        /bin/mv ${WebPageDirectory}/Antenna_iframe.html.tmp
${WebPageDirectory}/Antenna_iframe.html
    fi
    /usr/bin/wget http://127.0.0.1:${LocalTunnelPort}/Antenna.html -q -
O ${WebPageDirectory}/Antenna.html.tmp
    /bin/mv ${WebPageDirectory}/Antenna.html.tmp
${WebPageDirectory}/Antenna.html

    # = Log
==============================================================
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Fetch Log(_iframe).html"
    fi
    if [ "$FetchFrameFiles" -eq "1" -o ! -f
${WebPageDirectory}/Log_iframe.html ]; then
        /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/Log_iframe.html -q -O
${WebPageDirectory}/Log_ifram
e.html.tmp
        /bin/mv ${WebPageDirectory}/Log_iframe.html.tmp
${WebPageDirectory}/Log_iframe.html
    fi
    /usr/bin/wget http://127.0.0.1:${LocalTunnelPort}/Log.html -q -O
${WebPageDirectory}/Log.html.tmp
    /bin/mv ${WebPageDirectory}/Log.html.tmp
${WebPageDirectory}/Log.html

    # = Station Monitoring
===============================================
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Fetch StationMonitoring(_iframe).html"
    fi
    if [ "$FetchFrameFiles" -eq "1" -o ! -f
${WebPageDirectory}/StationMonitoring_iframe.html ]; then
        /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/StationMonitoring_iframe.html -q -O
${WebPageDirectory}/StationMonitoring_iframe.html.tmp
        /bin/mv ${WebPageDirectory}/StationMonitoring_iframe.html.tmp
${WebPageDirectory}/StationMonitoring_iframe.html
    fi
    /usr/bin/wget
http://127.0.0.1:${LocalTunnelPort}/StationMonitoring.html -q -O
${WebPageDirectory}/StationMonitoring.html.tmp
    /bin/mv ${WebPageDirectory}/StationMonitoring.html.tmp
${WebPageDirectory}/StationMonitoring.html

     # = Timetag
```

```
==========================================================
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Create timetag file TimeOfLastCompleteUpdate.txt and
execution time file ExecutionTime.txt"
    fi
    if [ "$FetchFrameFiles" -eq "1" ]; then
        /bin/echo $StartTimeSeconds >
${WEBPAGES_PATH}${AntennaName}/TimeOfLastCompleteUpdate.txt
    fi
    EndTime=`/bin/date +%s.%N`
    ExecutionTime=$(/bin/echo "$EndTime - $StartTime" | bc | awk
'{printf "%f", $0}')
    /bin/echo $ExecutionTime >
${WEBPAGES_PATH}${AntennaName}/ExecutionTime.txt
    sleep 0.15
done
exit 0
```

- If everything works correctly the command “ps ax | grep fetch | grep -v 'grep'” should show something like this:
- 

```
ps ax | grep fetch | grep -v "grep"
11451 ?        S      0:13 /bin/bash /home/oper/Software/vlbisysmon/bin//fetch_fs_webpage_single.sh wettzell 8083
11453 ?        S      0:22 /bin/bash /home/oper/Software/vlbisysmon/bin//fetch_fs_webpage_single.sh wettz13n 8084
28662 ?        S      0:27 /bin/bash /home/oper/Software/vlbisysmon/bin//fetch_fs_webpage_single.sh wettz13s 8085
```

5) Reading data into ZABBIX monitoring on the Central Monitoring Server

- The following operations are prepared by the administrator of the central monitoring system and are then automated. You will get feedback about.
- The data are taken from the web pages fetched from the NASA Field System Computers and stored locally on the Central Monitoring Server
- The extraction is done with the following script “/home/oper/Software/vlbisysmon/bin/getvalue_fs_webpage.sh”:

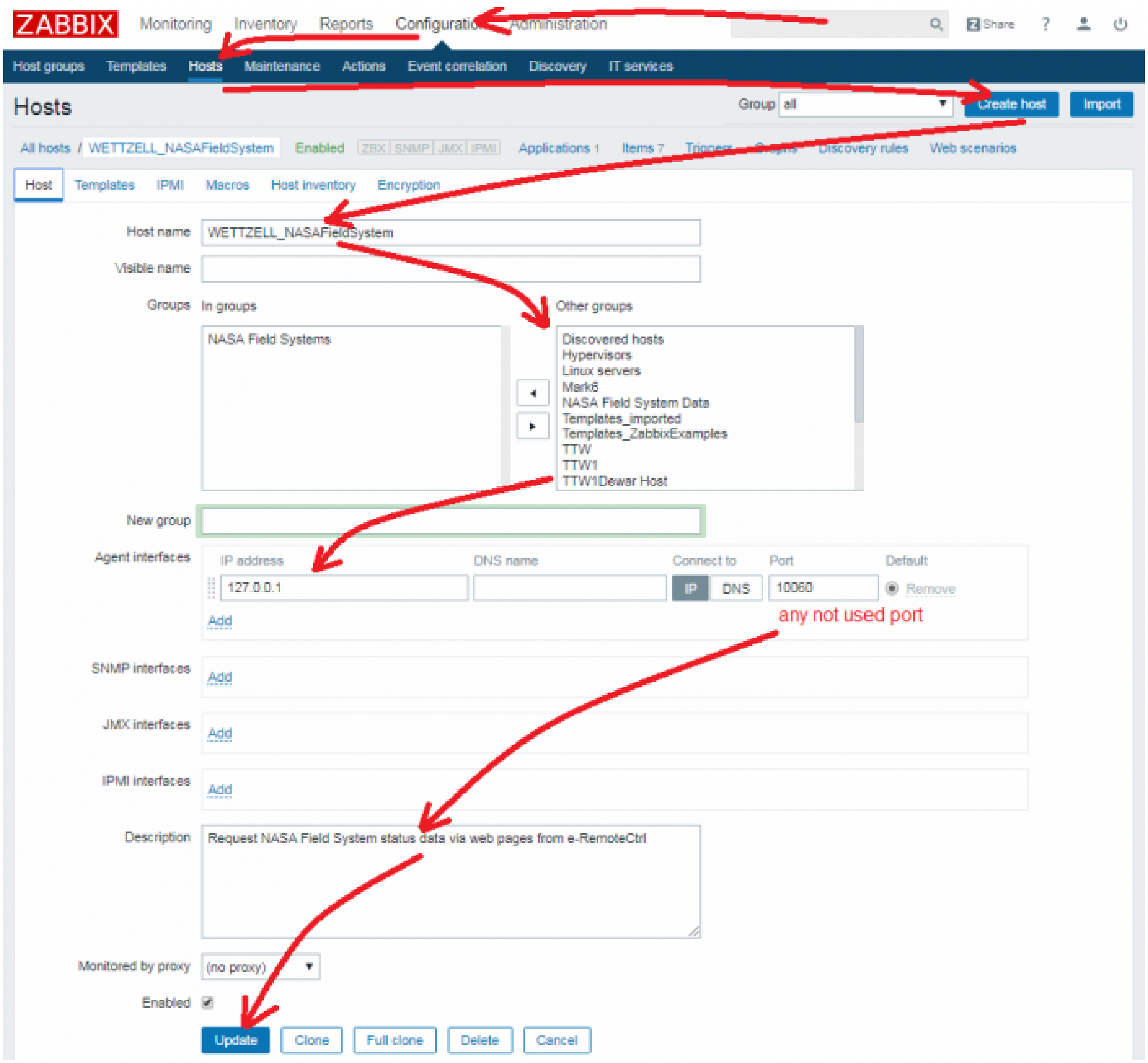
-
  ```
  MONITORING_ARCHIVE_PATH="MONITORING_ARCHIVE_PATH="/var/www/html/monitor
  ing_archive/"
  WEBPAGES_PATH="${MONITORING_ARCHIVE_PATH}fs_web_pages/"
  VERBOSE_ON=1
  HOMEDIR=$(dirname "$0")

  if [ $# -lt 3 ]; then
      echo "1"
      exit 1
  fi

  HOSTNAME=$1
  ANTENNA=$2
  PATTERN=$3
  TIMETAG=""

  # Convert pattern to ZABBIX key
  ```

```
COMMAND="echo ${PATTERN} | sed -r 's/::/./g'"
KEY=`eval $COMMAND`

# Find value and use first match
FILE="SystemStatusMonitor.html"
COMMAND="grep -o '<!--${PATTERN}-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--${PATTERN}-
->\s*\(.*\)\s*<!--\s*-->/\1/'"
VALUE=`eval $COMMAND`
COMMAND="grep -o '<!--ERC::TIMECOVERED-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--ERC::TIMECOVERED-
->\s*<!--\s*\(.*\)\s*-->\s*<!--\s*-->/\1/'"
TIMETAG=`eval $COMMAND`
if [[ -z $TIMETAG ]]; then
    COMMAND="grep -o '<!--ERC::TIME-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--ERC::TIME-
->\s*\(.*\)\s*<!--\s*-->/\1/'"
    TIMETAG=`eval $COMMAND`
fi
if [[ -z $VALUE ]]; then
    FILE="SystemTemperatures.html"
    COMMAND="grep -o '<!--${PATTERN}-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--${PATTERN}-
->\s*\(.*\)\s*<!--\s*-->/\1/'"
    VALUE=`eval $COMMAND`
    COMMAND="grep -o '<!--ERC::TIMECOVERED-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--ERC::TIMECOVERED-
->\s*<!--\s*\(.*\)\s*-->\s*<!--\s*-->/\1/'"
    TIMETAG=`eval $COMMAND`
    if [[ -z $TIMETAG ]]; then
        COMMAND="grep -o '<!--ERC::TIME-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/SystemStatusMonitor.html | sed -e 's/^<!--
ERC::TIME-->\s*\(.*\)\s*<!--\s*-->/\1/'"
        TIMETAG=`eval $COMMAND`
    fi
    if [[ -z $VALUE ]]; then
        FILE="Antenna.html"
        COMMAND="grep -o '<!--${PATTERN}-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--${PATTERN}-
->\s*\(.*\)\s*<!--\s*-->/\1/'"
        VALUE=`eval $COMMAND`
        COMMAND="grep -o '<!--ERC::TIMECOVERED-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--ERC::TIMECOVERED-
->\s*<!--\s*\(.*\)\s*-->\s*<!--\s*-->/\1/'"
        TIMETAG=`eval $COMMAND`
        if [[ -z $TIMETAG ]]; then
            COMMAND="grep -o '<!--ERC::TIME-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/SystemStatusMonitor.html | sed -e 's/^<!--
ERC::TIME-->\s*\(.*\)\s*<!--\s*-->/\1/'"
            TIMETAG=`eval $COMMAND`
        fi
        if [[ -z $VALUE ]]; then
            FILE="Mark5RemainingCapacity.html"
            COMMAND="grep -o '<!--${PATTERN}-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--${PATTERN}-
->\s*\(.*\)\s*<!--\s*-->/\1/'"
            VALUE=`eval $COMMAND`
            COMMAND="grep -o '<!--ERC::TIMECOVERED-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--ERC::TIMECOVERED-
->\s*<!--\s*\(.*\)\s*-->\s*<!--\s*-->/\1/'"
            TIMETAG=`eval $COMMAND`
            if [[ -z $TIMETAG ]]; then
                COMMAND="grep -o '<!--ERC::TIME-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/SystemStatusMonitor.html | sed -e 's/^<!--
ERC::TIME-->\s*\(.*\)\s*<!--\s*-->/\1/'"
                TIMETAG=`eval $COMMAND`
            fi
            if [[ -z $VALUE ]]; then
                FILE="PhaseCalMonitoring.html"
                COMMAND="grep -o '<!--${PATTERN}-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--${PATTERN}-
->\s*\(.*\)\s*<!--\s*-->/\1/'"
                VALUE=`eval $COMMAND`
                COMMAND="grep -o '<!--ERC::TIMECOVERED-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--ERC::TIMECOVERED-
->\s*<!--\s*\(.*\)\s*-->\s*<!--\s*-->/\1/'"
                TIMETAG=`eval $COMMAND`
                if [[ -z $TIMETAG ]]; then
                    COMMAND="grep -o '<!--ERC::TIME-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/SystemStatusMonitor.html | sed -e 's/^<!--
ERC::TIME-->\s*\(.*\)\s*<!--\s*-->/\1/'"
                    TIMETAG=`eval $COMMAND`
                fi
                if [[ -z $VALUE ]]; then
                    FILE="Log.html"
                    COMMAND="grep -o '<!--${PATTERN}-->[^<]*<!-- -->'
${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--${PATTERN}-
->\s*\(.*\)\s*<!--\s*-->/\1/'"
                    VALUE=`eval $COMMAND`
                    COMMAND="grep -o '<!--ERC::TIMECOVERED-->[^<]*<!--
-->' ${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--
ERC::TIMECOVERED-->\s*<!--\s*\(.*\)\s*-->\s*<!--\s*-->/\
1/'"
                    TIMETAG=`eval $COMMAND`
                    if [[ -z $TIMETAG ]]; then
                        COMMAND="grep -o '<!--ERC::TIME-->[^<]*<!-- -
->' ${WEBPAGES_PATH}${ANTENNA}/SystemStatusMonitor.html | sed -e
's/^<!--ERC::TIME-->\s*\(.*\)\s*<!--\s*-->/\1/'"
                        TIMETAG=`eval $COMMAND`
                    fi
                    if [[ -z $VALUE ]]; then
                        FILE="WebCam.html"
                        COMMAND="grep -o '<!--${PATTERN}-->[^<]*<!-- -
->' ${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--${PATTERN}-
->\s*\(.*\)\s*<!--\s*-->/\1/'"
                        VALUE=`eval $COMMAND`
                        COMMAND="grep -o '<!--ERC::TIMECOVERED-
->[^<]*<!-- -->' ${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--
ERC::TIMECOVERED-->\s*<!--\s*\(.*\)\s*-->\s*<!--\s*-
->/\1/'"
                        TIMETAG=`eval $COMMAND`
                        if [[ -z $TIMETAG ]]; then
                            COMMAND="grep -o '<!--ERC::TIME-->[^<]*<!--
-->' ${WEBPAGES_PATH}${ANTENNA}/SystemStatusMonitor.html | sed -e
's/^<!--ERC::TIME-->\s*\(.*\)\s*<!--\s*-->/\1/
'"
                            TIMETAG=`eval $COMMAND`
                        fi
                        if [[ -z $VALUE ]]; then
                            FILE="StationMonitoring.html"
                            COMMAND="grep -o '<!--${PATTERN}-->[^<]*<!-
- -->' ${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--${PATTERN}-
->\s*\(.*\)\s*<!--\s*-->/\1/'"
                            VALUE=`eval $COMMAND`
                            COMMAND="grep -o '<!--ERC::TIMECOVERED-
->[^<]*<!-- -->' ${WEBPAGES_PATH}${ANTENNA}/${FILE} | sed -e 's/^<!--
ERC::TIMECOVERED-->\s*<!--\s*\(.*\)\s*-->\s*<!--
\s*-->/\1/'"
                            TIMETAG=`eval $COMMAND`
                            if [[ -z $TIMETAG ]]; then
                                COMMAND="grep -o '<!--ERC::TIME-
->[^<]*<!-- -->' ${WEBPAGES_PATH}${ANTENNA}/SystemStatusMonitor.html |
sed -e 's/^<!--ERC::TIME-->\s*\(.*\)\s*<!--\s*-->
/\1/'"
                                TIMETAG=`eval $COMMAND`
                            fi
                            if [[ -z $VALUE ]]; then
                               # Tag not found
                               # ============
                               echo "1"
                               exit 1
                            fi
                        fi
                    fi
                fi
            fi
        fi
    fi
fi
```

```
# Manipulate specific values to optimze the ZABBIX representation
if [ "$PATTERN" == "ERC::CABLE" ]; then
    VALUE=$(echo "${VALUE}*1000" | bc)
fi
if [ "$PATTERN" == "ERC::SELECTED_MODULEA" ]; then
    if [ "$VALUE" == ">" ]; then
        VALUE=1
    else
        VALUE=0
    fi
fi
if [ "$PATTERN" == "ERC::SELECTED_MODULEB" ]; then
    if [ "$VALUE" == ">" ]; then
        VALUE=1
    else
        VALUE=0
    fi
fi
if [ "$PATTERN" == "ERC::TIME_MODULEA" ] || [ "$PATTERN" ==
"ERC::TIME_MODULEB" ]; then
    COMMAND="echo '${VALUE}' | sed -e
's/^\([0-9]\+\)\s*h\s*[0-9]\+m\?/\1/'"
    REMAINING_HOURS=`eval $COMMAND`
    COMMAND="echo '${VALUE}' | sed -e
's/^[0-9]\+\s*h\s*\([0-9]\+\)m\?/\1/'"
    REMAINING_MINUTES=`eval $COMMAND`
    VALUE=$(echo "scale=2;
${REMAINING_HOURS}+(${REMAINING_MINUTES}/60.0)" | bc)
fi

# Create time tag information
#echo $TIMETAG # 2018.187.15:10:03
COMMAND="echo '${TIMETAG}' | sed -e
's/^\([0-9]\+\)\.[0-9]\+\.[0-9]\+:[0-9]\+:[0-9]\+/\1/'"
YEAR=`eval $COMMAND`
#echo $YEAR
COMMAND="echo '${TIMETAG}' | sed -e
's/^[0-9]\+\.[0]*\([0-9]\+\)\.[0-9]\+:[0-9]\+:[0-9]\+/\1/'"
DOY=`eval $COMMAND`
DOY=`echo $((--DOY))`
#echo $DOY
COMMAND="echo '${TIMETAG}' | sed -e
's/^[0-9]\+\.[0-9]\+\.\([0-9]\+\):[0-9]\+:[0-9]\+/\1/'"
HOUR=`eval $COMMAND`
#echo $HOUR
COMMAND="echo '${TIMETAG}' | sed -e
's/^[0-9]\+\.[0-9]\+\.[0-9]\+:\([0-9]\+\):[0-9]\+/\1/'"
MINUTE=`eval $COMMAND`
#echo $MINUTE
COMMAND="echo '${TIMETAG}' | sed -e
```

```
's/^[0-9]\+\.[0-9]\+\.[0-9]\+:[0-9]\+:\([0-9]\+\)/\1/'"
SECOND=`eval $COMMAND`
#echo $SECOND

if [[ -z $YEAR ]] || [[ -z $DOY ]] || [[ -z $HOUR ]] || [[ -z $MINUTE
]] || [[ -z $SECOND ]]; then
    TIMESTAMP=$(date +%s)
else
    TIMESTAMP=$(date -d "${DOY} days ${YEAR}-01-01
${HOUR}:${MINUTE}:${SECOND}" +"%s")
fi
#echo $TIMESTAMP

# Send value to ZABBIX
echo "${HOSTNAME} ${KEY} ${TIMESTAMP} ${VALUE}" |
${HOMEDIR}/zabbix_sender -z 127.0.0.1 -T -i - > /dev/null 2> /dev/null

echo "0"
exit 0
```

- The script is called by a ZABBIX host. Each Field System PC requires a separate host in ZABBIX. It must be created manually with the following steps (e.g. for antenna “WETTZELL”).

- 

Hosts

Group: all

All hosts / WETTZELL_NASAFieldSystem | Enabled | ZBX SNMP JMX IPMI | Applications 1 | Items 7 | Triggers | Graphs | Discovery rules | Web scenarios

Host | Templates | IPMI | Macros | Host inventory | Encryption

Host name: WETTZELL_NASAFieldSystem

Visible name:

Groups: In groups: NASA Field Systems

Other groups: Discovered hosts, Hypervisors, Linux servers, Mark6, NASA Field System Data, Templates_imported, Templates_ZabbixExamples, TTW, TTW1, TTW1Dewar Host

New group:

Agent interfaces:

IP address	DNS name	Connect to	Port	Default
127.0.0.1		IP	10060	● Remove

SNMP interfaces: Add

JMX interfaces: Add

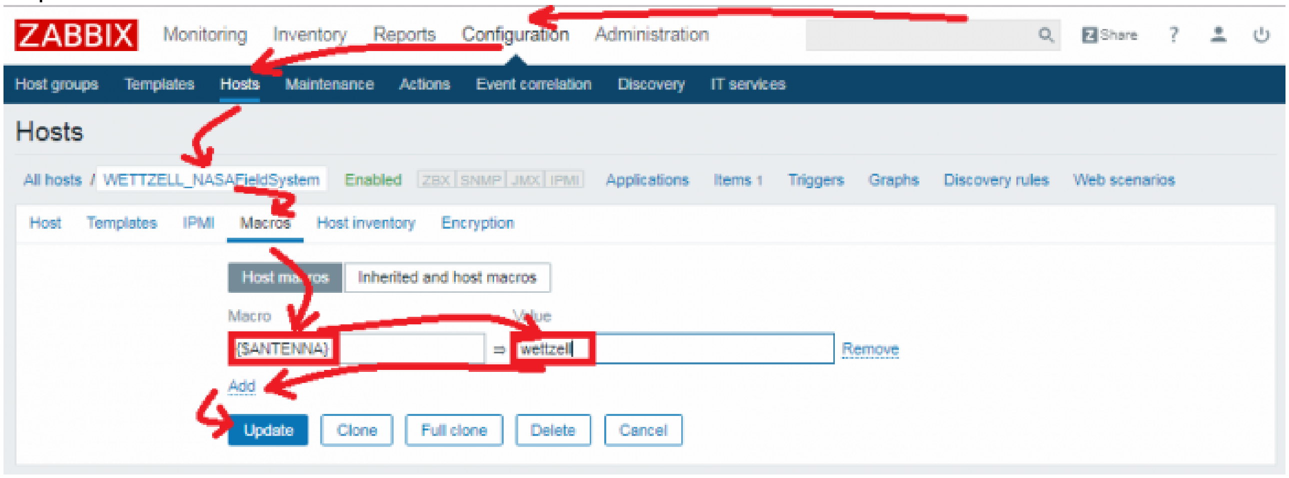
IPMI interfaces: Add

Description: Request NASA Field System status data via web pages from e-RemoteCtrl

Monitored by proxy: (no proxy)

Enabled: ✓

Update | Clone | Full clone | Delete | Cancel

- Important is to create a macro with the antenna name in the host.
- 

Hosts

All hosts / WETTZELL_NASAFieldSystem | Enabled | ZBX SNMP JMX IPMI | Applications | Items 1 | Triggers | Graphs | Discovery rules | Web scenarios

Host | Templates | IPMI | Macros | Host inventory | Encryption

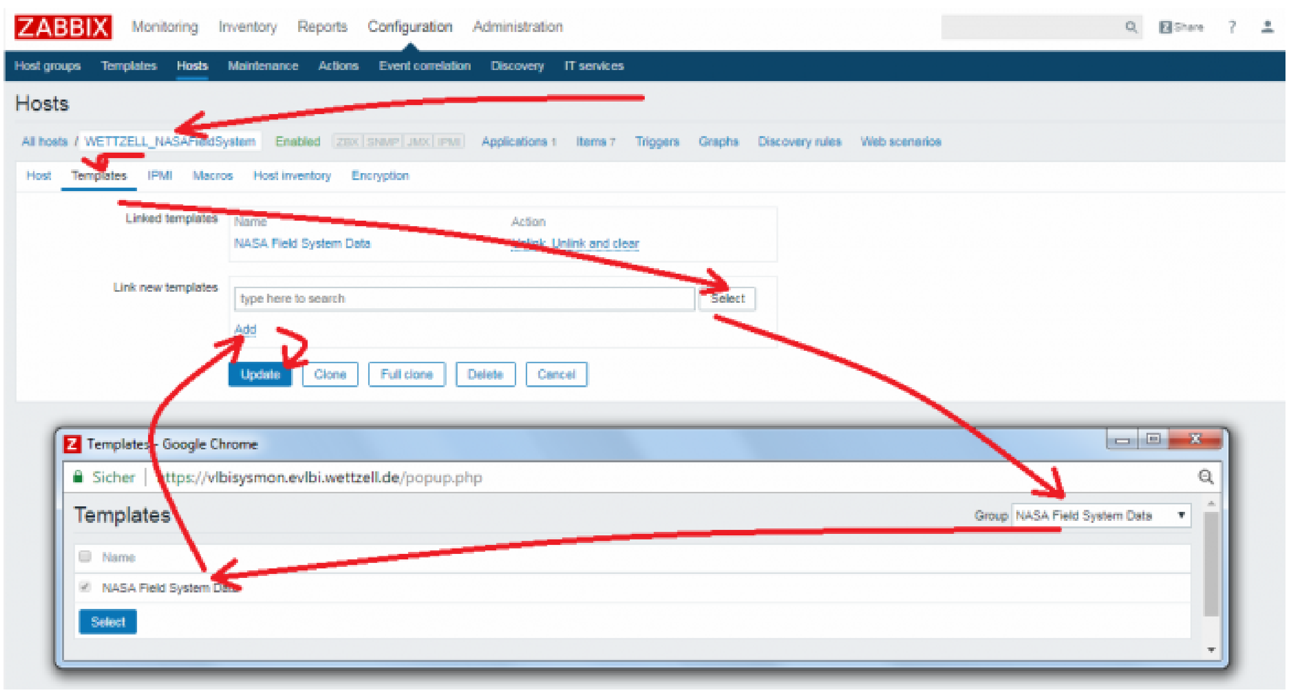
Host macros | Inherited and host macros

Macro		Value	
{$ANTENNA}	⇒	wettzell	Remove

Add

Update | Clone | Full clone | Delete | Cancel

- Now everything is prepared to activate the predefined template to read all available data from the web pages of a NASA Field System. The current version is this: 20180707_zbx_nasa_field_system_data_template.xml

- 

  Hosts

  All hosts / WETTZELL_NASAFieldSystem Enabled ZBX SNMP JMX IPMI Applications 1 Items 7 Triggers Graphs Discovery rules Web scenarios

  Host | Templates | IPMI | Macros | Host inventory | Encryption

Linked templates	Name	Action
	NASA Field System Data	Unlink Unlink and clear

  Link new templates: type here to search [Select]

  Add

  Update Clone Full clone Delete Cancel

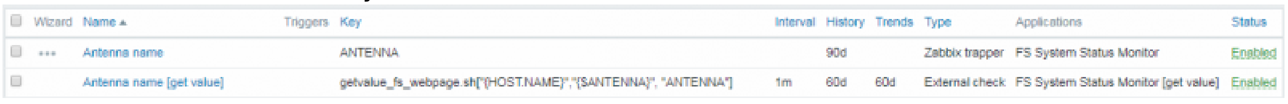
  Templates - Google Chrome

  Sicher | https://vlbisysmon.evlbi.wettzell.de/popup.php

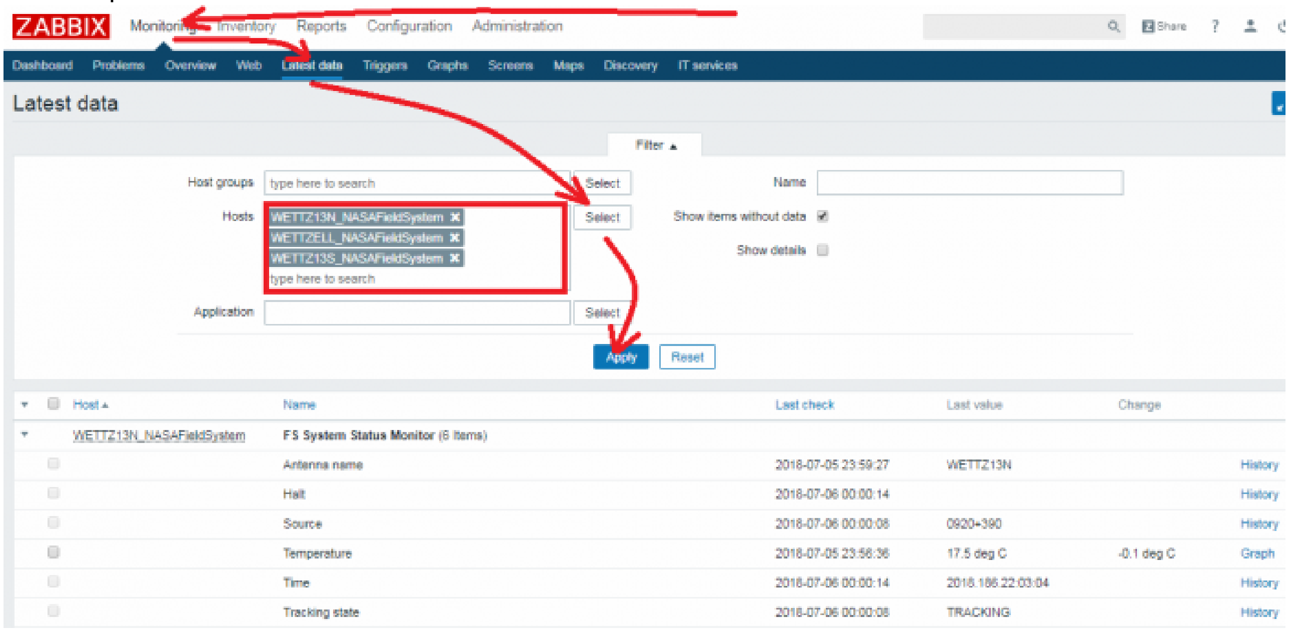
  Templates Group NASA Field System Data

	Name
✓	NASA Field System Data

  Select

- The template creates two items for each value from the NASA Field System: one is to activate the extraction by calling the script "/home/oper/Software/vlbisysmon/bin/getvalue_fs_webpage.sh" as "External check" and the other is the real item value which is filled using "zabbix_sender" in the script taking the real time from the NASA FIeld System.
- 

Wizard	Name ▲	Triggers	Key	Interval	History	Trends	Type	Applications	Status
•••	Antenna name		ANTENNA		90d		Zabbix trapper	FS System Status Monitor	Enabled
	Antenna name [get value]		getvalue_fs_webpage.sh["{HOST.NAME}","{$ANTENNA}", "ANTENNA"]	1m	60d	60d	External check	FS System Status Monitor [get value]	Enabled

- Push "Update" for the host and check the latest data, if the values are monitored.
- 

  Latest data

  Filter ▲

  Host groups: type here to search [Select] Name:

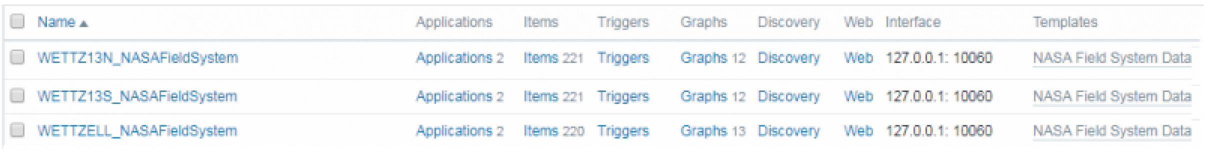
  Hosts: WETTZ13N_NASAFieldSystem ✕ WETTZELL_NASAFieldSystem ✕ WETTZ13S_NASAFieldSystem ✕ type here to search [Select] Show items without data ✓

  Show details

  Application: [Select]

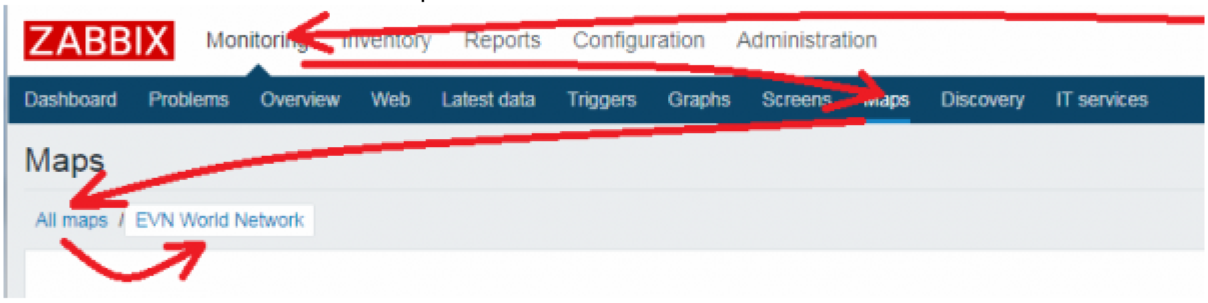
  Apply Reset

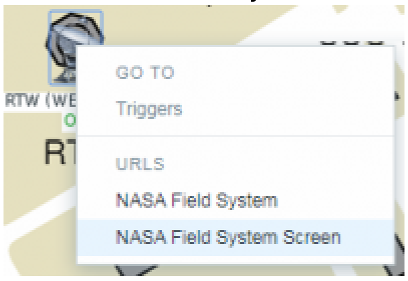
Host ▲	Name	Last check	Last value	Change	
WETTZ13N_NASAFieldSystem	FS System Status Monitor (6 Items)				
	Antenna name	2018-07-05 23:59:27	WETTZ13N		History
	Halt	2018-07-06 00:00:14			History
	Source	2018-07-06 00:00:08	0920+390		History
	Temperature	2018-07-05 23:56:36	17.5 deg C	-0.1 deg C	Graph
	Time	2018-07-06 00:00:14	2018.186.22:03:04		History
	Tracking state	2018-07-06 00:00:08	TRACKING		History

- The current test antennas are
  - 

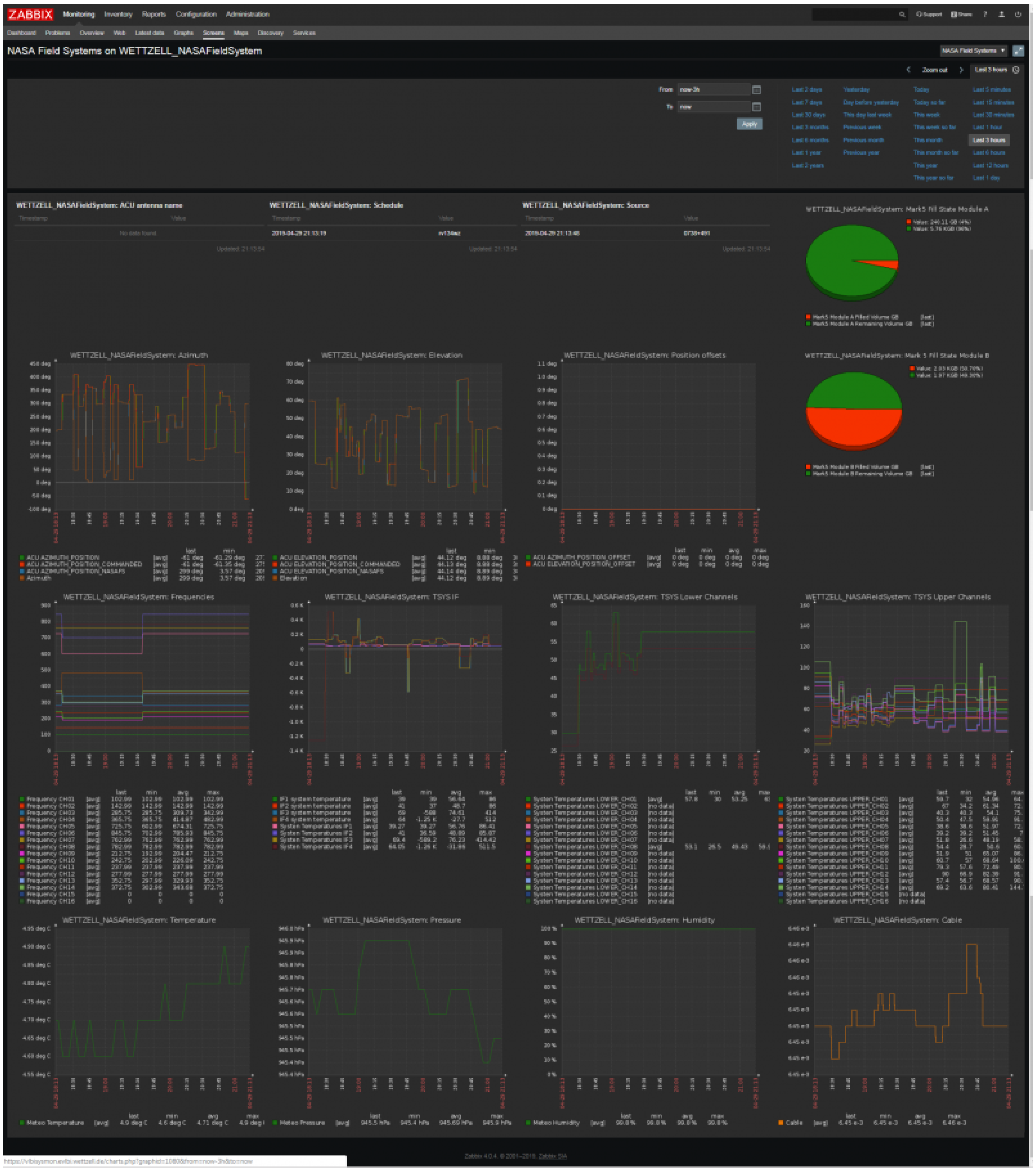
Name ▲	Applications	Items	Triggers	Graphs	Discovery	Web	Interface	Templates
WETTZ13N_NASAFieldSystem	Applications 2	Items 221	Triggers	Graphs 12	Discovery	Web	127.0.0.1: 10060	NASA Field System Data
WETTZ13S_NASAFieldSystem	Applications 2	Items 221	Triggers	Graphs 12	Discovery	Web	127.0.0.1: 10060	NASA Field System Data
WETTZELL_NASAFieldSystem	Applications 2	Items 220	Triggers	Graphs 13	Discovery	Web	127.0.0.1: 10060	NASA Field System Data

6) Accessing data and web pages by antenna staff

- After a positive feedback, you are now able to access the data pages of the monitoring system.
- You received a username and password. It can be used to login to the central monitoring page and to your specific copy of the NASA Field System pages. Just connect to https://vlbisysmon.evlbi.wettzell.de.
- Select the EVN or IVS World Map
- 

- Search and select your antenna to get a drop down menu.
- 

    - Select access to the NASA Field System page (or call the direct access to the individual copies of the NASA Field System pages)https://vlbisysmon.evlbi.wettzell.de/monitoring_archive/fs_web_pages/<antennaname>/FieldSystemMonitoring.html, where <antennaname> must be replaced by the official antenna name also shown in the system status monitor of the NASA Field System)
    - or select the summary page with graphs of the collected data


ZABBIX
Monitoring
Inventory
Reports
Configuration
Administration
Dashboard
Problems
Overview
Web
Latest data
Graphs
Screens
Maps
Discovery
Services
NASA Field Systems on WETTZELL_NASAFieldSystem
Zoom out
Last 3 hours
From now-3h
To now
Apply
WETTZELL_NASAFieldSystem: Mark5 Fill State Module A
WETTZELL_NASAFieldSystem: Azimuth
WETTZELL_NASAFieldSystem: Elevation
WETTZELL_NASAFieldSystem: Position offsets
WETTZELL_NASAFieldSystem: Mark 5 Fill State Module B
WETTZELL_NASAFieldSystem: Frequencies
WETTZELL_NASAFieldSystem: TSYS IF
WETTZELL_NASAFieldSystem: TSYS Lower Channels
WETTZELL_NASAFieldSystem: TSYS Upper Channels
WETTZELL_NASAFieldSystem: Temperature
WETTZELL_NASAFieldSystem: Pressure
WETTZELL_NASAFieldSystem: Humidity
WETTZELL_NASAFieldSystem: Cable
Zabbix 4.0.4. © 2001–2018, Zabbix SIA

WETTZELL_NASAFieldSystem: ACU antenna name

Timestamp	Value
No data found	

Updated: 21:13:54

WETTZELL_NASAFieldSystem: Schedule

Timestamp	Value
2019-04-29 21:13:19	rv134wz

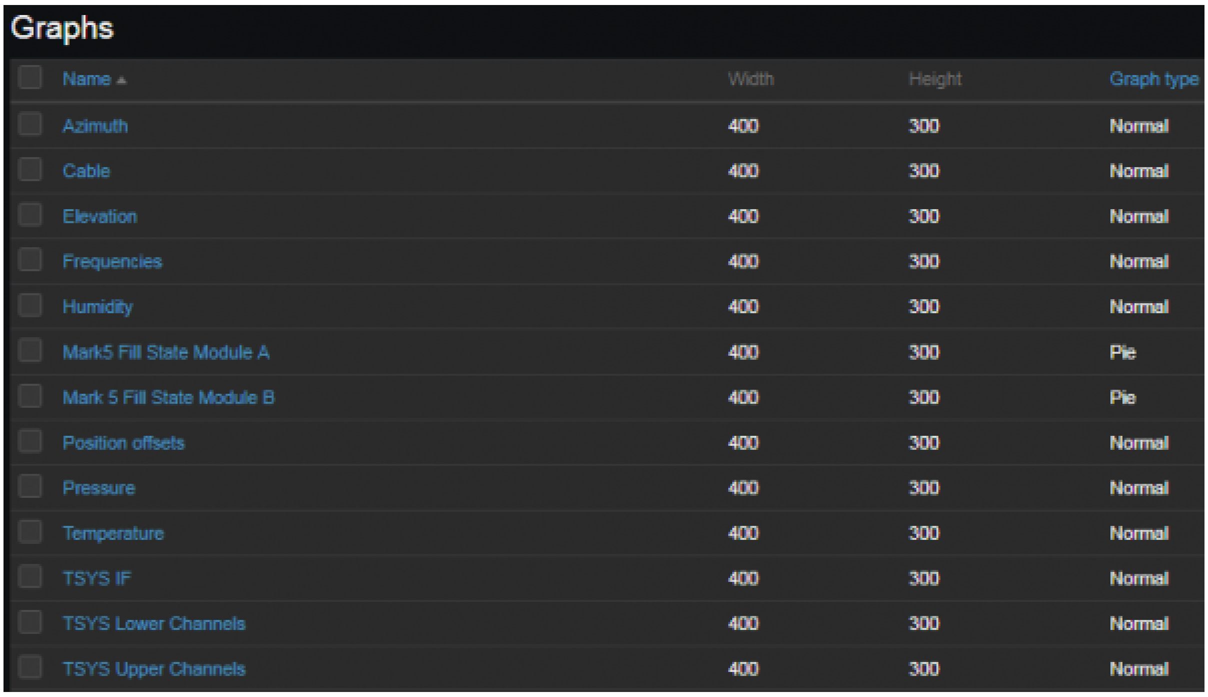
Updated: 21:13:54

WETTZELL_NASAFieldSystem: Source

Timestamp	Value
2019-04-29 21:13:48	0738+491

Updated: 21:13:54

- ○ https://vlbisysmon.evlbi.wettzell.de/charts.php?graphid=1080&from=now-3h&to=now
- ○ Collected are 110 values: see
  20180816valuesavailable_latestdata.pdf
  plus 110 bool values showing if the request of each value succeeded
- ○ The following graphs are predefined

- 

Graphs

Name ▲	Width	Height	Graph type
Azimuth	400	300	Normal
Cable	400	300	Normal
Elevation	400	300	Normal
Frequencies	400	300	Normal
Humidity	400	300	Normal
Mark5 Fill State Module A	400	300	Pie
Mark 5 Fill State Module B	400	300	Pie
Position offsets	400	300	Normal
Pressure	400	300	Normal
Temperature	400	300	Normal
TSYS IF	400	300	Normal
TSYS Lower Channels	400	300	Normal
TSYS Upper Channels	400	300	Normal

7) Adapting station specific parts

- **Adapting the general web page**
  - It is possible to adapt any web page of the e-RemoteCtrl software.
  - There was made a point of keeping the web pages simple using standard HTML. Only IFRAME web files use Javascript sections in a very limited way.
  - The general structure is currently set in the FieldSystemMonitoring.html file in form of frames. If pages should not be shown, they can be left away.
  -
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
    <FRAMESET rows="18%,47%,35%">
        <FRAMESET cols="65%,35%">
            <FRAME src="SystemStatusMonitor_iframe.html"
name="SystemStatusMonitor" scrolling="no">
            <FRAME src="Mark5RemainingCapacity_iframe.html"
name="Mark5RemainingCapacity" scrolling="no">
        </FRAMESET>
        <FRAMESET cols="20%,15%,35%,30%">
            <FRAME src="SystemTemperatures_iframe.html"
name="SystemTemperatures" scrolling="no">
            <FRAME src="StationMonitoring_iframe.html"
name="StationMonitoring" scrolling="no">
            <FRAME src="WebCam_iframe.html" name="WebCam_iframe"
scrolling="no">
            <FRAME src="Antenna_iframe.html" name="Antenna_iframe"
scrolling="no">
        </FRAMESET>
        <frame src="Log_iframe.html" name="Log_iframe"
```

```
scrolling="no">
    </FRAMESET>
</HTML>
```

- If page files do not exist, they are not used.
- The content of each page can be adapted. Adapted pages are visible after restart of the e-RemoteCtrl server. Each time one of the key tag is found it will be replaced by the corresponding values. Missing key tags are not used and do not participate to the monitoring. IFRAME page files "…_iframe.html" should not be changed.
- A sample of a content page is shown here for the case of "Mark5RemainingCapacity.html":
-
```
 <html>
<head>
<!--ERC:: <meta http-equiv="refresh" content="1">  -->
</head>
<body>

<table width="100%" border="1" bgcolor="#FFFFFF" cellpadding="0"
cellspacing="0" align="left">
 <tr>
  <th scope="col" bgcolor="#D6E3BC" colspan="7" style="font-
size:14pt;">Mark 5 Remaining Capacity</th>
 </tr>
 <tr>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="center"
width="50"> <font color="#000000">&nbsp;</font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="center"
width="50"> <font color="#000000">&nbsp;</font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="center"
width="160"> <font color="#000000"><b>VSN</b></font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="center"
width="160"> <font color="#000000"><b>Time</b></font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="center"
width="160"> <font color="#000000"><b>GB</b></font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="center"
width="100"> <font color="#000000"><b>%</b></font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="center"
width="160"> <font color="#000000"><b>Check UT</b></font></td>
 </tr>
 <tr>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="left"
width="50"> <font color="#000000"><!--ERC::SELECTED_MODULEA-
->&nbsp;</font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="center"
width="50"> <font color="#000000"><b>A</b></font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="left"
width="160"> <font color="#000000"><!--ERC::VSN_MODULEA-
->&nbsp;</font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="160"> <font color="#000000"><!--ERC::TIME_MODULEA-
```

```
->&nbsp;</font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="160"> <font color="#000000"><!--ERC::REMEININGGB_MODULEA-
->&nbsp;</font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="100"> <font color="#000000"><!--
ERC::REMEININGPERCENT_MODULEA-->&nbsp;</font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="left"
width="160"> <font color="#000000"><!--ERC::CHECKTIME_MODULEA-
->&nbsp;</font></td>
 </tr>
 <tr>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="50" colspan="2"> 0%</td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="left"
width="50" colspan="4"> <!--
ERC::REMEININGPERCENT_FILLGRAPH_MODULEA--></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="left"
width="50">100% (Volume)</td>
 </tr>
 <tr>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="left"
width="50"> <font color="#000000"><!--ERC::SELECTED_MODULEB-
->&nbsp;</font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="center"
width="50"> <font color="#000000"><b>B</b></font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="left"
width="160"> <font color="#000000"><!--ERC::VSN_MODULEB-
->&nbsp;</font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="160"> <font color="#000000"><!--ERC::TIME_MODULEB-
->&nbsp;</font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="160"> <font color="#000000"><!--ERC::REMEININGGB_MODULEB-
->&nbsp;</font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="100"> <font color="#000000"><!--
ERC::REMEININGPERCENT_MODULEB-->&nbsp;</font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="left"
width="160"> <font color="#000000"><!--ERC::CHECKTIME_MODULEB-
->&nbsp;</font></td>
 </tr>
 <tr>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="50" colspan="2"> 0%</td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="left"
width="50" colspan="4"> <!--
ERC::REMEININGPERCENT_FILLGRAPH_MODULEB--></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="left"
width="50">100% (Volume)</td>
```

```
 </tr>
</table>

 <!--ERC::TIMECOVERED-->



</body>
</html>
```

  - Adapting the HTML code of the web pages gives all possibilities to the station staff. The staff can decide if and which values are shown. Antenna staff can also decide, which style is used for the web pages. Limited is the naming and number of web page files to those defined above.
- **Using Station Monitoring web page**
  - While all other web page files are just read once during startup of the e-RemoteCtrl server and then just updated in the memory, the file “StationMonitoring.html” is read during each request by a browser.
  - It usually does not contain any key tags (except “<!–ERC::TIMECOVERED–>” which is used to give a time reference from the NASA Field System). But it is possible to define individual tags (see in sections above). The file is usually generated externally, e.g. by a station program. The program must write the complete HTML code of the web page and store it in the file “StationMonitoring.html” each time. Therefore, the content is completely individual.
  - An example script writing some HTML output is shown here:
  -
```
 #!/bin/bash
#
=====================================================================
===========================
# Do the generation of Station Monitoring Web Pages
#
=====================================================================
===========================

WHOAMI=`/usr/bin/whoami`

echo "<HTML>"
echo " <HEAD>"
echo " </HEAD>"
echo " <BODY>"
echo " <CENTER>"
echo " <TABLE border=\"1\" bgcolor=\"#FFFFFF\" cellpadding=\"0\"
cellspacing=\"0\" width=\"100%\">"
echo " <TR>"
echo " <TH scope=\"col\" bgcolor=\"#D6E3BC\" style=\"font-
size:14pt;\"> Station Monitoring </th>"
echo " </TR>"
echo " </TABLE>"
echo " </CENTER>"
echo " &nbsp;"
echo " <!--ERC::TIMECOVERED-->"
echo ""
echo " </BODY>"
echo "</HTML>"
```

  - This script can be called as a cron job (use “crontab -e” to edit cron jobs for an regularly activation of the script), to get updates in minute intervals, e.g.:

  -
    ```
     # m h dom mon dow command
     * * * * * /usr2/st_rtw/scripts/CreateStationLog.sh >
    /usr2/st_rtw/eremotectrl/html/StationMonitoring.html
    ```

- **Enabling antenna monitoring (programming with C and C++)**
  - Antenna monitoring can be integrated into Station Monitoring web page or the e-RemoteCtrl specific station code can be used to inject antenna parameter into the remote monitoring output.
  - The specific station code is written in C/C++ and can be found in the server code directory ./srcst. eremotectrlst_antennacontrol.cpp/.hpp is used here for the reporting of antenna specific parameters. The following functions must be filled with program code:

  -
    ```
     unsigned short usSRCSTACUInit (const CSimpleStructuredConf &
    CStationConfiguration,
                                   void * & pvACUInstance)
    {
        std::string strServerIP;
        std::string strServerPort;
        unsigned int uiServerPort = 0;

        pvACUInstance = NULL;

        if (((CSimpleStructuredConf)CStationConfiguration).usGetValue
    ("eremotectrld.StationSpecifics.AntennaControl.ServerIP",
    strServerIP))
        {
            return SRCSTACU_NOK;
        }
        if (((CSimpleStructuredConf)CStationConfiguration).usGetValue
    ("eremotectrld.StationSpecifics.AntennaControl.ServerPort",
    strServerPort))
        {
            return SRCSTACU_NOK;
        }
        uiServerPort = (unsigned int) atoi(strServerPort.c_str());
        if (uiServerPort < 1 ||
                uiServerPort > 65535)
        {
            return SRCSTACU_NOK;
    ```

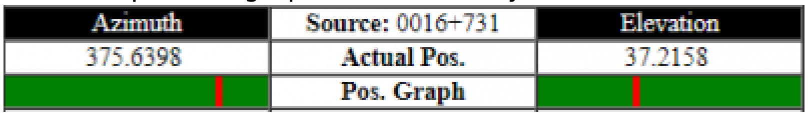
```
    }

    printf ("usSRCSTACUInit\n");
    return SRCSTACU_OK;
}

unsigned short usSRCSTACUClose (void * & pvACUInstance)
{
    printf ("usSRCSTACUClose\n");
    return SRCSTACU_NOK;
}

unsigned short usSRCSTACUGetStatus (void * & pvACUInstance,
                                    unsigned short & usError,
                                    std::string & strAntennaName,
                                    std::string & strACUTime,
                                    std::string &
strCurrentSource,
                                    double &
dActualPositionAzimuthDegree,
                                    double &
dActualPositionElevationDegree,
                                    double &
dCommandedPositionAzimuthDegree,
                                    double &
dCommandedPositionElevationDegree,
                                    double &
dPositionOffsetAzimuthDegree,
                                    double &
dPositionOffsetElevationDegree,
                                    std::string & strAzimuthMode,
                                    std::string &
strElevationMode,
                                    std::list<std::string> &
CStatusList,
                                    std::list<std::string> &
CErrorList)
{
    printf ("usSRCSTACUGetStatus\n");
    return SRCSTACU_NOK;
}

unsigned short usSRCSTACUMoveToPosition (void * & pvACUInstance,
        double  dPositionFirstAxis,
        double  dPositionSecondAxis,
        std::string strMountSystemIdentifier)
{
    printf ("usSRCSTACUMoveToPosition\n");
    return SRCSTACU_NOK;
}
```

```
unsigned short usSRCSTACUCommandOrder (void * & pvACUInstance,
                                       std::string
strOrderIdentifier)
{
    printf ("usSRCSTACUCommandOrder\n");
    return SRCSTACU_NOK;
}

unsigned short usSRCSTACUGetSupportedOrders (void * &
pvACUInstance,
        std::list<std::string> & COrderList)
{
    printf ("usSRCSTACUGetSupportedOrders\n");
    return SRCSTACU_NOK;
}

unsigned short usSRCSTACUGetSupportedMountSystems (void * &
pvACUInstance,
        std::list<std::string> & CMountSystemList)
{
    printf ("usSRCSTACUGetSupportedMountSystems\n");
    return SRCSTACU_NOK;
}
```

- “usSRCSTACUInit” is called during startup. It is used to initialize the connection to the antenna monitoring. The example shows the reading of configuration file sections with the stadardized method from the e-RemoteCtrl server.
  - Parameter: const CSimpleStructuredConf & CStationConfiguration → Station part of the configuration file
  - Parameter: void * & pvACUInstance ← Pointer to the antenna control instance (must be converted to specific type)
  - Return value: unsigned short ← Errorcode (EREMOTECTRL_ACU_OK, EREMOTECTRL_ACU_NOK)
- “usSRCSTACUClose” is called if the server is shut down to disconnect from the antenna monitoring.
  - Parameter: void * & pvACUInstance ↔ Pointer to the antenna control instance (must be converted to specific type)
  - Return value: unsigned short ← Errorcode (EREMOTECTRL_ACU_OK, EREMOTECTRL_ACU_NOK)
- “usSRCSTACUGetStatus” returns defined antenna status parameter
  - Parameter: void * & pvACUInstance ↔ Pointer to the antenna control instance (must be converted to specific type)
  - Parameter: unsigned short & usError ← Current error number
  - Parameter: std::string & strAntennaName ← Antenna name
  - Parameter: std::string & strACUTime ← Time of the antenna control unit
  - Parameter: std::string & strCurrentSource ← Current source
  - Parameter: double & dActualPositionAzimuthDegree ← Actual position in azimuth
  - Parameter: double & dActualPositionElevationDegree ← Actual position in elevation
  - Parameter: double & dCommandedPositionAzimuthDegree ← Commanded position in azimuth

- Parameter: double & dCommandedPositionElevationDegree ← Commanded position in elevation
- Parameter: double & dPositionOffsetAzimuthDegree ← Position offset in azimuth
- Parameter: double & dPositionOffsetElevationDegree ← Position offset in elevation
- Parameter: std::string & strAzimuthMode ← Mode of the azimuth control
- Parameter: std::string & strElevationMode ← Mode of the elevation control
- Parameter: std::list<std::string> & CStatusList ← List with status information
- Parameter: std::list<std::string> & CErrorList ← List with error information
- Return value: unsigned short ← Errorcode (EREMOTECTRL_ACU_OK, EREMOTECTRL_ACU_NOK)
- Sample:

```
 unsigned short usSRCSTACUGetStatus (void * & pvACUInstance,
                                    unsigned short & usError,
                                    std::string & strAntennaName,
                                    std::string & strACUTime,
                                    std::string &
strCurrentSource,
                                    double &
dActualPositionAzimuthDegree,
                                    double &
dActualPositionElevationDegree,
                                    double &
dCommandedPositionAzimuthDegree,
                                    double &
dCommandedPositionElevationDegree,
                                    double &
dPositionOffsetAzimuthDegree,
                                    double &
dPositionOffsetElevationDegree,
                                    std::string & strAzimuthMode,
                                    std::string &
strElevationMode,
                                    std::list<std::string> &
CStatusList,
                                    std::list<std::string> &
CErrorList)
{
    bool bError = false;
    acu_ttw_client * pCACUInstance = NULL;
    std::string * pstrStatusList_val = NULL;
    unsigned int _pstrStatusList_len = 0;
    std::string * pstrErrorList_val = NULL;
    unsigned int _pstrErrorList_len = 0;
    std::string strEmptyString;

    /* printf ("usSRCSTACUGetStatus\n"); */

    /// Init return values
    usError = 0;
    strAntennaName = strEmptyString.c_str();
    strACUTime = strEmptyString.c_str();
    strCurrentSource = strEmptyString.c_str();
    dActualPositionAzimuthDegree = 0.0;
    dActualPositionElevationDegree = 0.0;
    dCommandedPositionAzimuthDegree = 0.0;
    dCommandedPositionElevationDegree = 0.0;
    dPositionOffsetAzimuthDegree = 0.0;
    dPositionOffsetElevationDegree = 0.0;
    strAzimuthMode = strEmptyString.c_str();
    strElevationMode = strEmptyString.c_str();
    CStatusList.clear();
    CErrorList.clear();

    if (pvACUInstance == NULL)
    {
        bError = true;
        goto Cleanup;
    }
    pCACUInstance = (acu_ttw_client *) pvACUInstance;

    /// Get status
    try
    {
        if (pCACUInstance->usGetStatus (usError,
                                        strAntennaName,
                                        strACUTime,
                                        strCurrentSource,
dActualPositionAzimuthDegree,
dActualPositionElevationDegree,
dCommandedPositionAzimuthDegree,
dCommandedPositionElevationDegree,
dPositionOffsetAzimuthDegree,
dPositionOffsetElevationDegree,
                                        strAzimuthMode,
                                        strElevationMode,
                                        pstrStatusList_val,
_pstrStatusList_len,
                                        pstrErrorList_val,
_pstrErrorList_len ))
        {
            bError = true;
            goto Cleanup;
        }
    }
    catch (...)
    {
        bError = true;
        goto Cleanup;
    }
    /// Convert arrays into lists
    try
    {
        if (pstrStatusList_val != NULL)
        {
            for (unsigned int uiIndex = 0; uiIndex <
_pstrStatusList_len; ++uiIndex)
            {
CStatusList.push_back(pstrStatusList_val[uiIndex]);
            }
            delete [] pstrStatusList_val;
            pstrStatusList_val = NULL;
        }
        if (pstrErrorList_val != NULL)
        {
            for (unsigned int uiIndex = 0; uiIndex <
_pstrErrorList_len; ++uiIndex)
            {
                CErrorList.push_back(pstrErrorList_val[uiIndex]);
            }
            delete [] pstrErrorList_val;
            pstrErrorList_val = NULL;
        }
    }
    catch (...)
    {
        bError = true;
        goto Cleanup;
    }

Cleanup:
    /// Cleanup
    if (bError)
    {
        usError = 0;
        strAntennaName = strEmptyString.c_str();
        strACUTime = strEmptyString.c_str();
        strCurrentSource = strEmptyString.c_str();
        dActualPositionAzimuthDegree = 0.0;
        dActualPositionElevationDegree = 0.0;
        dCommandedPositionAzimuthDegree = 0.0;
        dCommandedPositionElevationDegree = 0.0;
        dPositionOffsetAzimuthDegree = 0.0;
        dPositionOffsetElevationDegree = 0.0;
        strAzimuthMode = strEmptyString.c_str();
        strElevationMode = strEmptyString.c_str();
        CStatusList.clear();
        CErrorList.clear();
        if (pstrStatusList_val != NULL)
        {
            delete [] pstrStatusList_val;
            pstrStatusList_val = NULL;
        }
        if (pstrErrorList_val != NULL)
        {
            delete [] pstrErrorList_val;
            pstrErrorList_val = NULL;
        }
        return SRCSTACU_NOK;
    }

    return SRCSTACU_OK;
}
```

- “usSRCSTACUMoveToPosition”, “usSRCSTACUGetSupportedOrders”, and “usSRCSTACUGetSupportedMountSystems” were used for commanding of the antenna and are not used for the monitoring part.
- Using the e-RemoteCtrl-defined station code, it is possible to use the internal calculations etc. or fill the position graphs automatically:
- 

Azimuth	Source: 0016+731	Elevation
375.6398	Actual Pos.	37.2158
🟩	Pos. Graph	🟩

8) Remote commands (on a separate SSH connection and just for antenna staff)

- **The monitoring software is not designed and used for remote commanding of the antenna.**
- Using the technique of reverse tunneling it is not possible to use ports of the NASA Field System PC from external which are not opened by the SSH tunnel, started on the NASA Field System PC by the staff(see script: autossh_run.sh).
- Therefore, local staff can only use separate SSH connections to the NASA Field System PC to run the NASA Field System program “oprin” which enables to enter commands. The staff is responsible for these separate connections.
- (The classic e-RemoteCtrl client with wxWidgets-2.8 is still possible using the defined security levels. But it is not supported anymore.)