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

1. – *Alexander Neidhardt 2020/09/28 14:37*

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) General user models

- The seamless auxiliary server supports three user models:
  - Basic User: A basic user sends selected values for dedicated items (meteorology) using SSH calls. He has access to a guest account on the ZABBIX web server to request graphs. Data can be archived in monthly files.
  - Standard User: A standard user inlcudes all features from a basic user and uses e-RemoteCtrl on the NASA Field System PC. All data from the NASA Field System and additional station specific data can be injected according to tags in web pages using HTTP requests over an SSH tunnel. He has access to a system specific web page on the ZABBIX web server. ata can be archived in monthly files.
  - Premium User: A premium user includes all features from the standard user and uses an additional communication between ZABBIX components, so that other components (UPS via SNMP, servers, switches, etc.) can be monitored. Additionally, it is possible to optionally establish a command interface to actively control antennas.
- The different user models support different features and have different requirements (

  DOC

).

Features https://vlbisysmon.evlbi.wettzell.de/zabbix/

	Basic Site	Standard Site	Premium Site
	IVS Seamless Auxiliary Data (EVN Jumping JIVE)	IVS Seamless Auxiliary Data (EVN Jumping JIVE)	Control center
User Account			
Linux user account (rbash)	X	X	X
SSH-keyfile for user required	X	X	X
ZABBIX guest account available	X	X	X
ZABBIX standard user available		X	X
ZABBIX administration user available			X
Data injection			
Passive monitoring	X	X	X
Simple zabbix_sender via SSH available	X	X	X
Single SSH-calls used per data value	X	X	X
Monitoring of single FS values	X	X	X
Monitoring of single non-FS values	(X)	(X)	X
Use of Wettzell server timetag	X	X	X
Use of station UTC server time	X	X	X
Use of FS UTC time	(X)	X	X
e-RemoteCtrl software at station required		X	X
Permanent SSH-con. to Wettzell required		X	X
Monitoring of all relevant FS values		X	X
ZABBIX Proxy/Agentd at station required			X
Monitoring of additional data (UPS,Server)			X
Graphical ZABBIX Web Interface			
Station monitoring screen available	X	X	X
IVS Seamless Aux. Data screen available	X	X	X
FS graphs available	X	X	X
Local HTML status web pages available		X	X
World-wide HTML status web pages avail.		X	X
IVS quick-status web page available		X	X
IVS world map available		X	X
Alarm trigger for FS log errors available		X	X
Additional alarm triggers available			X
Additional screens available			X
Additional graphs available			X
Messaging system (email, phone) avail.			X
Data Archive			
Extraction of spec. data to daily text files	X	X	X
Forwarding of text files to IVS servers	X	X	X
Web access to download area	X	X	X
Conversion oft ext files to VGOSDB files	P	P	P
Extraction of additional data to daily text f.			X
Remote Control			
Remote access to control system			X

1) Accounts

- Each user has a restricted user account on the ZABBIX server. It uses a version of “rbash” with a specific, adapted shell which restricts access and program calls. The shell is “rbash.sh” (an extended version also allows “scp” in the list of commands and is called “rbash_scp.sh”, but it is only used for specific users):

```
#!/bin/bash

# Script found here:
https://stackoverflow.com/questions/402615/how-to-restrict-ssh-users-to
-a-predefined-set-of-commands-after-login
# and modified

commands=("./bin/zabbix_sender" "date")
timestamp(){ date +'%Y-%m-%s %H:%M:%S'; }
#log(){ echo -e "$(timestamp)\t$1\t$(whoami)\t$2" > /var/log/rbash.log;
}
trycmd()
{
    # Process allowed shell commands to exit shell
    if [[ "$ln" == "exit" ]] || [[ "$ln" == "q" ]] || [[ "$ln" ==
"quit" ]] || [[ "$ln" == "bye" ]]; then
        exit

    # Process allowed shell command "help" to show help information
    elif [[ "$ln" == "help" ]]; then
        echo "rbash.sh: Allowed commands:"
        echo "============================"
        echo "  Shell commands:"
        echo "  ---------------"
        echo "  help            => get this help"
        echo "  echo            => echo text"
        echo "  exit|quit|q|bye => exit shell"
        echo ""
        echo "  User defined commands:"
        echo "  ----------------------"
        echo "${commands[@]}" | tr ' ' '\n' | awk '{print "   " $0}'

    # Process allowed shell commands to echo strings or pipe
information to other programs
    elif [[ "$ln" =~ ^echo\ .*$ ]]; then
        if [[ "$ln" =~ ^.*\|.*$ ]]; then
                ECHO_PART="$(cut -d'|' -f1 <<< $ln)"
                COMMAND_PART="$(cut -d'|' -f2 <<< $ln)"
                COMMAND_ALLOWED=false
                for cmd in "${commands[@]}"; do
                        pattern="^\ *$cmd.*$"
                        if [[ "$COMMAND_PART" =~ $pattern ]]; then
```

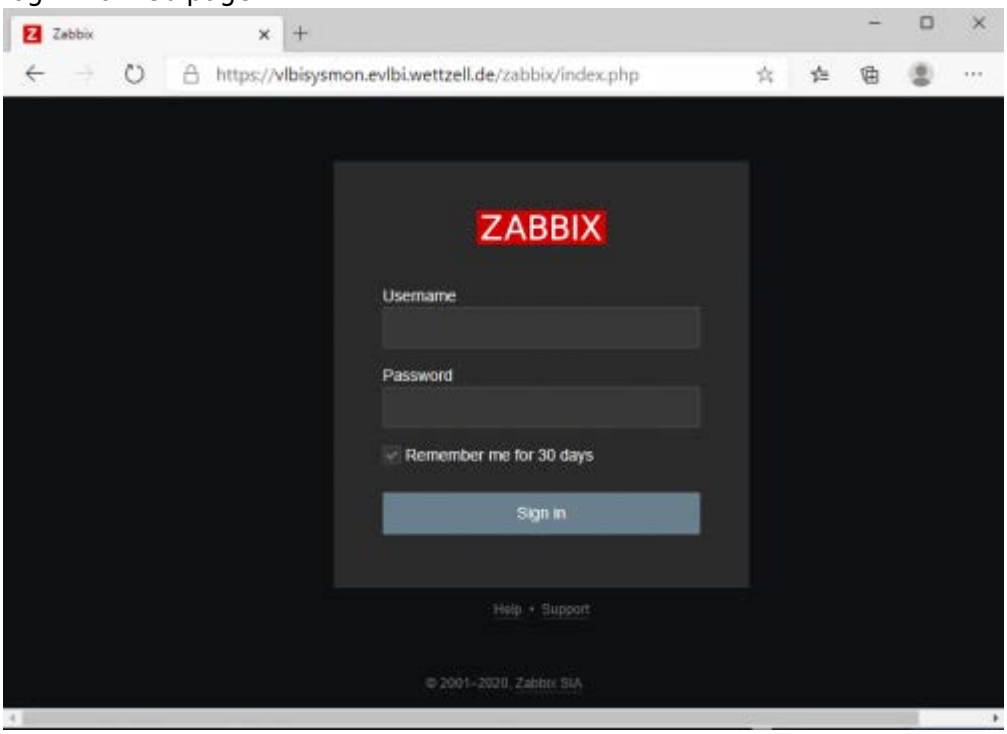
```
                                COMMAND_ALLOWED=true
                                break
                        fi
                done
                if [[ $COMMAND_ALLOWED = true ]] ; then
                        COMMAND="$ECHO_PART | $COMMAND_PART"
                        RESULT=`eval $COMMAND`
                        echo $RESULT
                else
                        echo -e "MIST rbash.sh DENIES $ln"
#                        log DENIED "$ln"
                fi
        else
                ln="${ln:5}"
                echo "$ln" # Beware, these double quotes are important
to prevent malicious injection
        fi
        # Log this command
#        log COMMAND "echo $ln"

    # Check entered command against list of user defined commands
    else
        COMMAND_ALLOWED=false
        for cmd in "${commands[@]}"; do
            pattern="^\ *$cmd.*$"
            if [[ "$ln" =~ $pattern ]]; then
                COMMAND_ALLOWED=true
                break
            fi
        done
        if [[ $COMMAND_ALLOWED = true ]]; then
            $ln
        else
            echo -e "rbash.sh DENIES $ln"
#            log DENIED "$ln"
        fi
    fi
}

# Optionally show a friendly welcome-message
if [[ "$ln" =~ ^scp ]]; then
    echo "$(timestamp) Welcome, $(whoami). Type 'help' for
information."
fi

# Optionally log the login
#log LOGIN "$@"

# Optionally log the logout
#trap "trap=\"\";exit" EXIT
```

```
# Process user defined commands, print prompt, and run shell in manual
user mode
if [[ "$1" == "-c" ]]; then
    shift
    #echo -e "> $@"
    ln=$@
    trycmd "$@"
else
    while echo -n "> " && read ln; do
        trycmd "$ln"
    done
fi
```

- Such account can be used with SSH login and a SSH key file. The key file must be requested from the server administrator (currently: alexander.neidhardt@tum.de)

- `ssh -i .ssh/<priv_key> <user>@141.74.6.20`
  `(where <priv_key is the private SSH key and <user> is the user name)`

- Additionally, Some user have ZABBIX credentials in form of a user name and a password to login via web page.



2) Data Injection

2.1) Basic User

- The simplest method to inject data to ZABBIX is to run “zabbix_sender” with host (= station name), key (= value/item name), and current value as SSH command.
- It is possible to use the ZABBIX server time for the injection, ignoring data transfer delays over

internet, e.g.:

```
  ssh -i ssh/<priv_key> <user>@141.74.6.20 './bin/zabbix_sender -z
141.74.6.20 -p 10052  -s WETTZELL_001_NASAFieldSystem -k
ERC.TEMPERATURE -o 12.0'
 (where <priv_key is the private SSH key and <user> is the user
name)
```

- A better method is, to use the local time (e.g. from NTP or the NASA Field System) and tag all injected values with this time, so that the real time at generation is saved, e.g.:

```
 UnixTime=$(date +%s); ssh -i ssh/<priv_key> <user>@141.74.6.20
"echo 'WETTZELL_001_NASAFieldSystem ERC.TEMPERATURE $UnixTime
\"7.0\"' | ./bin/zabbix_sender -z 141.74.6.20 -p 10052  -T -i -"
 (where <priv_key is the private SSH key and <user> is the user
name)
```

- Used items are:
  - **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**
- There are some standardized tags
  - **ERC.DOTMON** - Dotom Clock Offset (GPS-FMOUT, etc.)
  - **ERC.DOTMON2** - Dotom2 Clock Offset (GPS-FMOUT, etc.)
- There are no predefined station items. They can be defined individually, but require a separate agreement.
- Station items should have the following structure: <STATIONNAME>.<ITEMNAME>

2.2) Standard User

2.2.1) 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
```

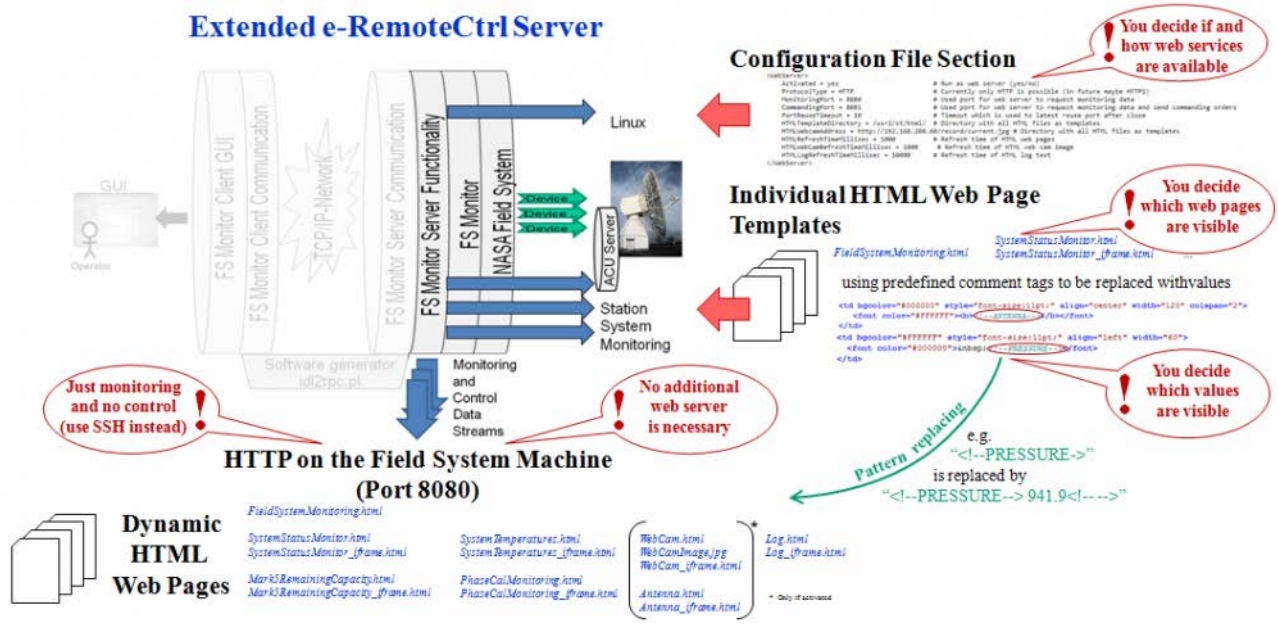
- You should get a new folder “eremotectrl_deliverable”
- Change into the new folder and build the code

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

- After building, you can continue with the configuration.

2.2.2) 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.



- 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`

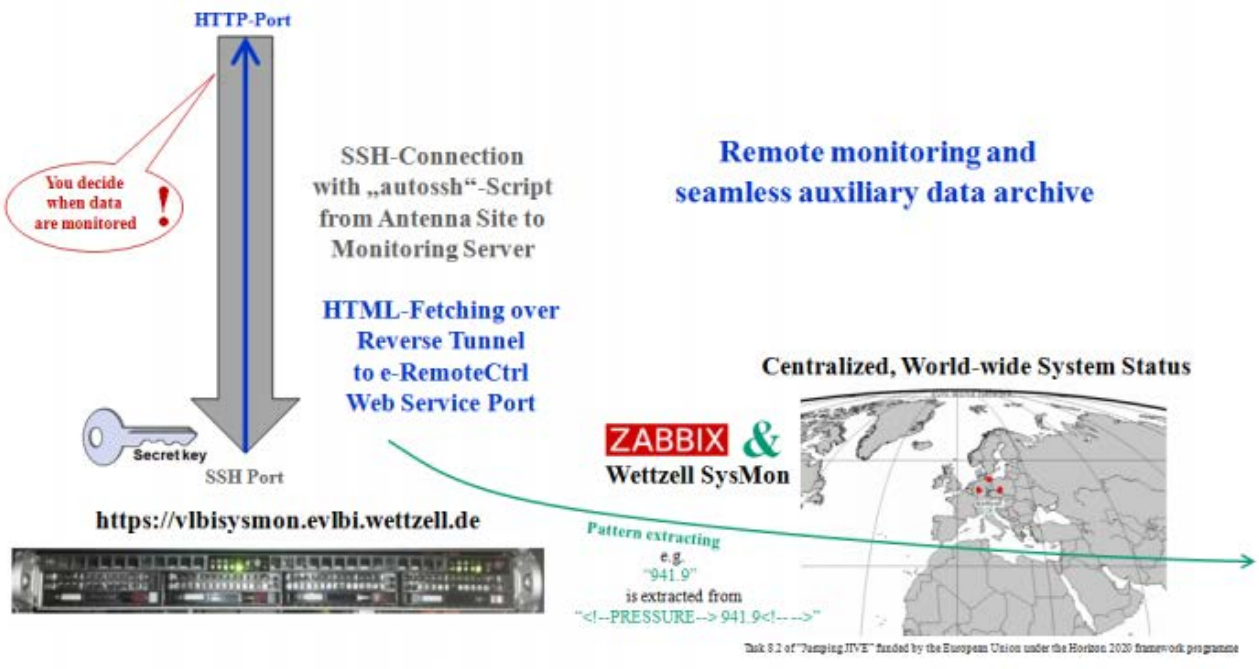
- If you open a web browser and connect to the IP address of the NASA Field System adding the port defined in the configuration file (e.g. http://fspc:8080) you should see the remote control web GUI. Attention: SHTTP is currently not supported.

- 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).

2.2.3) 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:

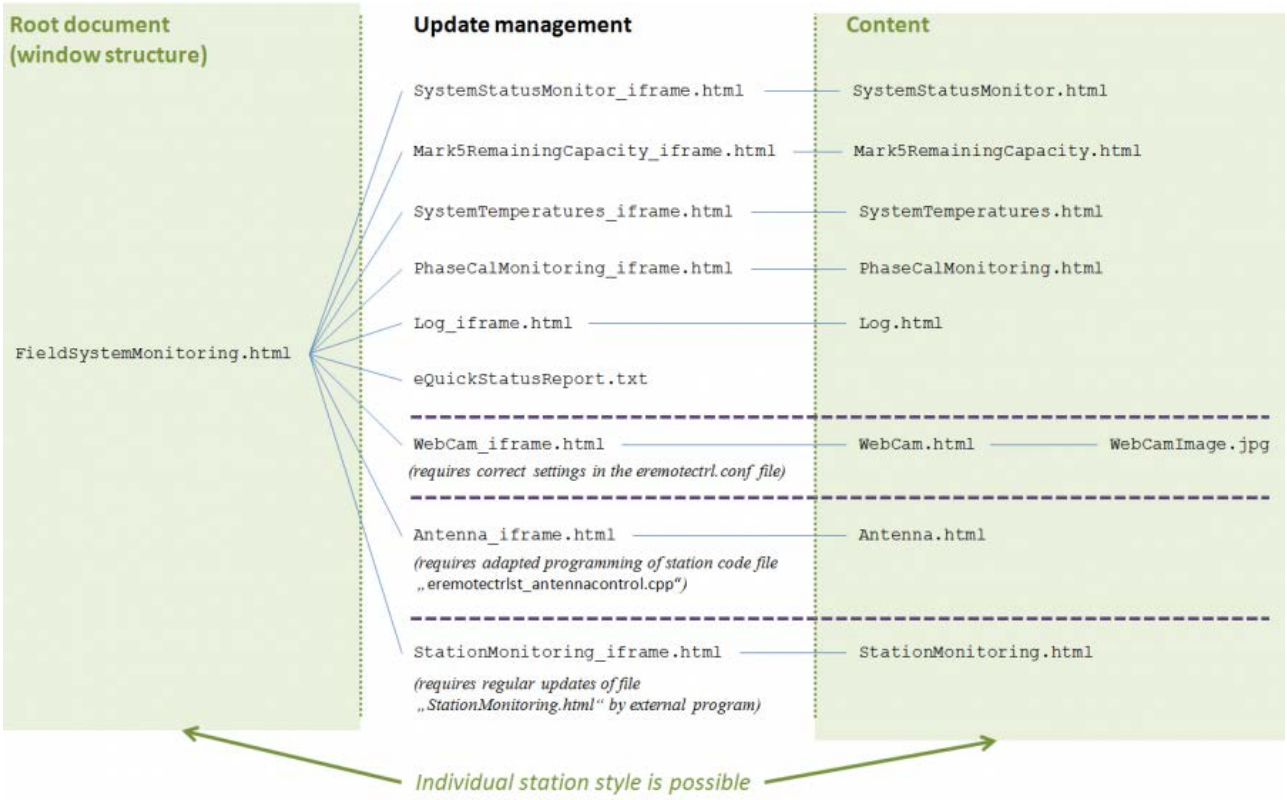


- The port number for HTTP are used as local endpoint of the reverse tunnel on the central monitoring server.
- Using the key file you can run the following script to start “autossh”
- Please take care that “autossh” is installed or install it if it is not yet available (as user “root”):
  - `apt-get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

2.2.4) Used data formats for the transfer

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



- 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 some standardized station tags.
  - **<!–ERC::DOTMON–>** - Dotmon Clock Offset (GPS minus FMOUT, etc.)
- 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.

2.2.5) 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="/raid/www/html/monitoring_archive/"
  WEBPAGES_PATH="${MONITORING_ARCHIVE_PATH}fs_web_pages/"
  VERBOSE_ON=1
  CREATE_IFRAME_PAGES=0
  WEB_PAGE_TEMPLATES=`/bin/echo $WEBPAGES_PATH/templates/`
  WEB_PAGE_TEMPLATES_SOURCE="/home/oper/Software/vlbisysmon/html/"

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

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

  HTMLFrameFileList=("index.html" \
  "SystemStatusMonitor_iframe.html" \
  "Mark5RemainingCapacity_iframe.html" \
  "SystemTemperatures_iframe.html" \
  "PhaseCalMonitoring_iframe.html" \
  ```

```
"WebCam_iframe.html" \
"Antenna_iframe.html" \
"Log_iframe.html" \
"StationMonitoring_iframe.html" )

HTMLFileList=("SystemStatusMonitor.html" \
"Mark5RemainingCapacity.html" \
"SystemTemperatures.html" \
"PhaseCalMonitoring.html" \
"WebCam.html" \
"Antenna.html" \
"Log.html" \
"StationMonitoring.html" \
"eQuickStatusReport.txt" )

# Special lists
HTMLFileListWETTZ13S=("StationSpectrum.html" \
"StationSpectrum0BandA.html" \
"StationSpectrum1BandB.html" \
"StationSpectrum2BandC.html" \
"StationSpectrum3BandD.html" \
"StationSpectrum0BandA.png" \
"StationSpectrum1BandB.png" \
"StationSpectrum2BandC.png" \
"StationSpectrum3BandD.png" \
"StationLatestScan.html" )
HTMLFileListWETTZELL=("StationQualityMonitoring.html" \
"StationSpectrumMonitoring.html" \
"StationLatestScan.html" \
"StationSpectrum.html" \
"StationSpectrum.png" )

while [[ 1 -eq 1 ]] ; do

    StartTime=`/bin/date +%s.%N`
    StartTimeSeconds=`/bin/date +%s`
    WebPageDirectory=$WEBPAGES_PATH$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 $WEBPAGES_PATH
        if [ "$CREATE_IFRAME_PAGES" -eq "1" ]; then
            FetchFrameFiles=2
        else
            FetchFrameFiles=1
        fi
    fi
    if [ ! -d "$WEB_PAGE_TEMPLATES" ]; then
        /bin/mkdir $WEB_PAGE_TEMPLATES
        /bin/cp ${WEB_PAGE_TEMPLATES_SOURCE}/* ${WEB_PAGE_TEMPLATES}/.
    fi
    if [ ! -d $WEBPAGES_PATH${AntennaName} ]; then
        /bin/mkdir ${WEBPAGES_PATH}${AntennaName}
        /bin/chmod 777 ${WEBPAGES_PATH}${AntennaName}
        if [ "$CREATE_IFRAME_PAGES" -eq "1" ]; then
            FetchFrameFiles=2
        else
            FetchFrameFiles=1
        fi
    fi

    if [ ! -f ${WebPageDirectory}/TimeOfLastCompleteUpdate.txt ]; then
        if [ "$CREATE_IFRAME_PAGES" -eq "1" ]; then
            FetchFrameFiles=2
        else
            FetchFrameFiles=1
        fi
    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 - $TimeOfLastCompleteUpdate"
| bc` -gt 600 ] ; then
            if [ "$CREATE_IFRAME_PAGES" -eq "1" ]; then
                FetchFrameFiles=2
            else
                FetchFrameFiles=1
            fi
        fi
    fi

    # = Copy IFRAME pages from local template directory
==================
    if [ "$FetchFrameFiles" -eq "2" ]; then
        /bin/cp ${WEB_PAGE_TEMPLATES}/* ${WebPageDirectory}/.
    fi
```

```
    # = Fetch all frame files
==========================================
    for file in "${HTMLFrameFileList[@]}" ; do
        if [ "$VERBOSE_ON" -eq "1" ]; then
            echo "Fetch ${WebPageDirectory}/$file"
        fi
        if [ -e ${WebPageDirectory}/${file} ]; then
            FileSizeIsZero=`wc -c < ${WebPageDirectory}/${file}`
        else
            FileSizeIsZero=0
        fi
        if [ "$FileSizeIsZero" -eq "0" -o "$FetchFrameFiles" -eq "1" -o
! -f ${WebPageDirectory}/${file} ]; then
            if [ "$VERBOSE_ON" -eq "1" ]; then
                echo "/usr/bin/wget --proxy=off -T 5 -t 1 -q -O
${WebPageDirectory}/${file}.tmp
http://127.0.0.1:${LocalTunnelPort}/${file}"
                echo "/bin/mv ${WebPageDirectory}/${file}.tmp
${WebPageDirectory}/${file}"
            fi
            /usr/bin/wget --proxy=off -T 5 -t 1 -q -o /dev/null -O
${WebPageDirectory}/${file}.tmp
http://127.0.0.1:${LocalTunnelPort}/${file}
            /bin/mv ${WebPageDirectory}/${file}.tmp
${WebPageDirectory}/${file}
        else
            if [ "$VERBOSE_ON" -eq "1" ]; then
                echo "==> Nothing to do!"
            fi
        fi
    done

    # = Fetch all other HTML files
========================================
    for file in "${HTMLFileList[@]}" ; do
        if [ "$VERBOSE_ON" -eq "1" ]; then
            echo "Fetch ${WebPageDirectory}/$file"
        fi
        if [ "$VERBOSE_ON" -eq "1" ]; then
            echo "/usr/bin/wget --proxy=off -T 5 -t 1 -q -O
${WebPageDirectory}/${file}.tmp
http://127.0.0.1:${LocalTunnelPort}/${file}"
            echo "/bin/mv ${WebPageDirectory}/${file}.tmp
${WebPageDirectory}/${file}"
        fi
        /usr/bin/wget --proxy=off -T 5 -t 1 -q -o /dev/null -O
${WebPageDirectory}/${file}.tmp
http://127.0.0.1:${LocalTunnelPort}/${file}
        /bin/mv ${WebPageDirectory}/${file}.tmp
${WebPageDirectory}/${file}
```

```
    done

    # = Fetch webcam image
==============================================
    WebCamImageFile=`/bin/sed -n -r 's/.*(WebCamImage\..*)\"\s+.*/\1/p'
${WebPageDirectory}/WebCam.html`
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Fetch ${WebPageDirectory}/${WebCamImageFile}"
    fi
    /usr/bin/wget --proxy=off -T 5 -t 1 -q -o /dev/null -O
${WebPageDirectory}/${WebCamImageFile}.tmp
http://127.0.0.1:${LocalTunnelPort}/${WebCamImageFile}
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Reduce size to 400x400 pixel"
    fi
    /usr/bin/convert ${WebPageDirectory}/${WebCamImageFile}.tmp -resize
400x400 ${WebPageDirectory}/${WebCamImageFile}.tmp2
    /bin/mv ${WebPageDirectory}/${WebCamImageFile}.tmp2
${WebPageDirectory}/${WebCamImageFile}
    /bin/rm ${WebPageDirectory}/${WebCamImageFile}.tmp

    # = Fetch all special HTML files
====================================
    if [ "$AntennaName" = "wettz13s" ]; then
        for file in "${HTMLFileListWETTZ13S[@]}" ; do
            if [ "$VERBOSE_ON" -eq "1" ]; then
                echo "Fetch ${WebPageDirectory}/$file"
            fi
            if [ "$VERBOSE_ON" -eq "1" ]; then
                echo "/usr/bin/wget --proxy=off -T 5 -t 1 -q -O
${WebPageDirectory}/${file}.tmp
http://127.0.0.1:${LocalTunnelPort}/${file}"
                echo "/bin/mv ${WebPageDirectory}/${file}.tmp
${WebPageDirectory}/${file}"
            fi
            /usr/bin/wget --proxy=off -T 5 -t 1 -q -o /dev/null -O
${WebPageDirectory}/${file}.tmp
http://127.0.0.1:${LocalTunnelPort}/${file}
            /bin/mv ${WebPageDirectory}/${file}.tmp
${WebPageDirectory}/${file}
        done
    fi
    if [ "$AntennaName" = "wettzell" ]; then
        for file in "${HTMLFileListWETTZELL[@]}" ; do
            if [ "$VERBOSE_ON" -eq "1" ]; then
                echo "Fetch ${WebPageDirectory}/$file"
            fi
            if [ "$VERBOSE_ON" -eq "1" ]; then
                echo "/usr/bin/wget --proxy=off -T 5 -t 1 -q -O
${WebPageDirectory}/${file}.tmp
```

```
http://127.0.0.1:${LocalTunnelPort}/${file}"
                echo "/bin/mv ${WebPageDirectory}/${file}.tmp
${WebPageDirectory}/${file}"
            fi
            /usr/bin/wget --proxy=off -T 5 -t 1 -q -o /dev/null -O
${WebPageDirectory}/${file}.tmp
http://127.0.0.1:${LocalTunnelPort}/${file}
            /bin/mv ${WebPageDirectory}/${file}.tmp
${WebPageDirectory}/${file}
        done
    fi

    # = 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

    # = Copy IVS quick status report
====================================
    if [ "$VERBOSE_ON" -eq "1" ]; then
        echo "Copy ${WebPageDirectory}/eQuickStatusReport.txt to
${WebPageDirectory}/eQuickStatusReport.info"
    fi
    /bin/cp ${WebPageDirectory}/eQuickStatusReport.txt
${WebPageDirectory}/eQuickStatusReport.info

    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
```

2.2.6) 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/get_fsvalue_from_webpage.sh":

```
MONITORING_ARCHIVE_PATH="/raid/www/html/monitoring_archive/"
WEBPAGES_PATH="${MONITORING_ARCHIVE_PATH}fs_web_pages/"
VERBOSE_ON=1
HOMEDIR=$(dirname "$0")
declare -a WebFileNames=("SystemStatusMonitor.html"
"SystemTemperatures.html"
"Antenna.html"
"Mark5RemainingCapacity.html"
"PhaseCalMonitoring.html"
"Log.html"
"WebCam.html"
"StationMonitoring.html")

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`

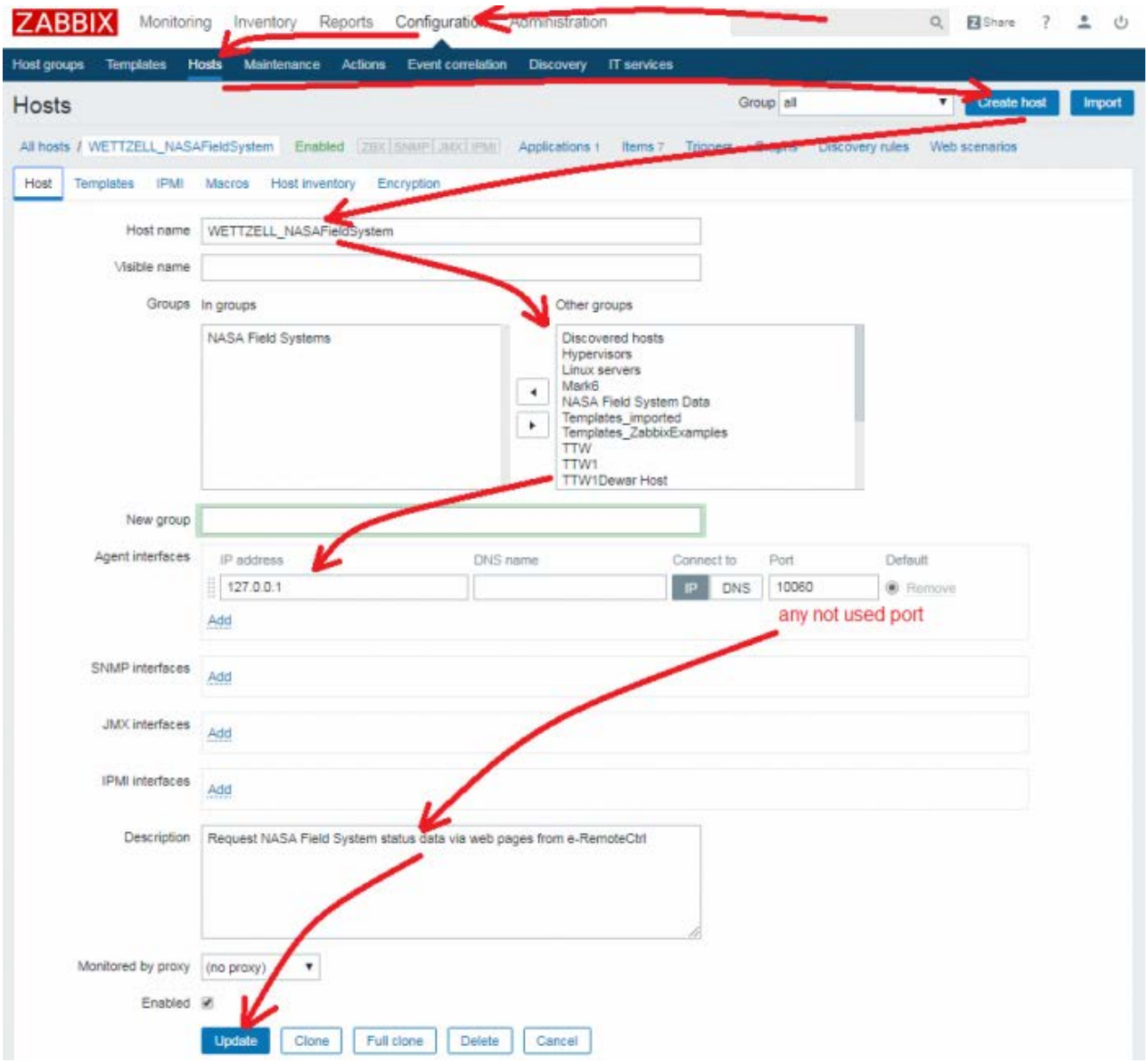
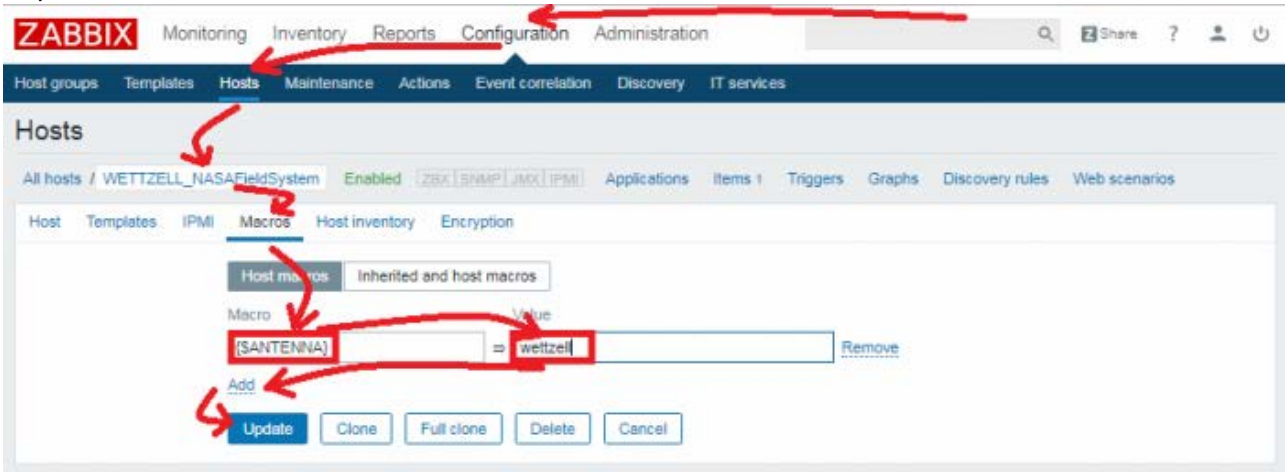
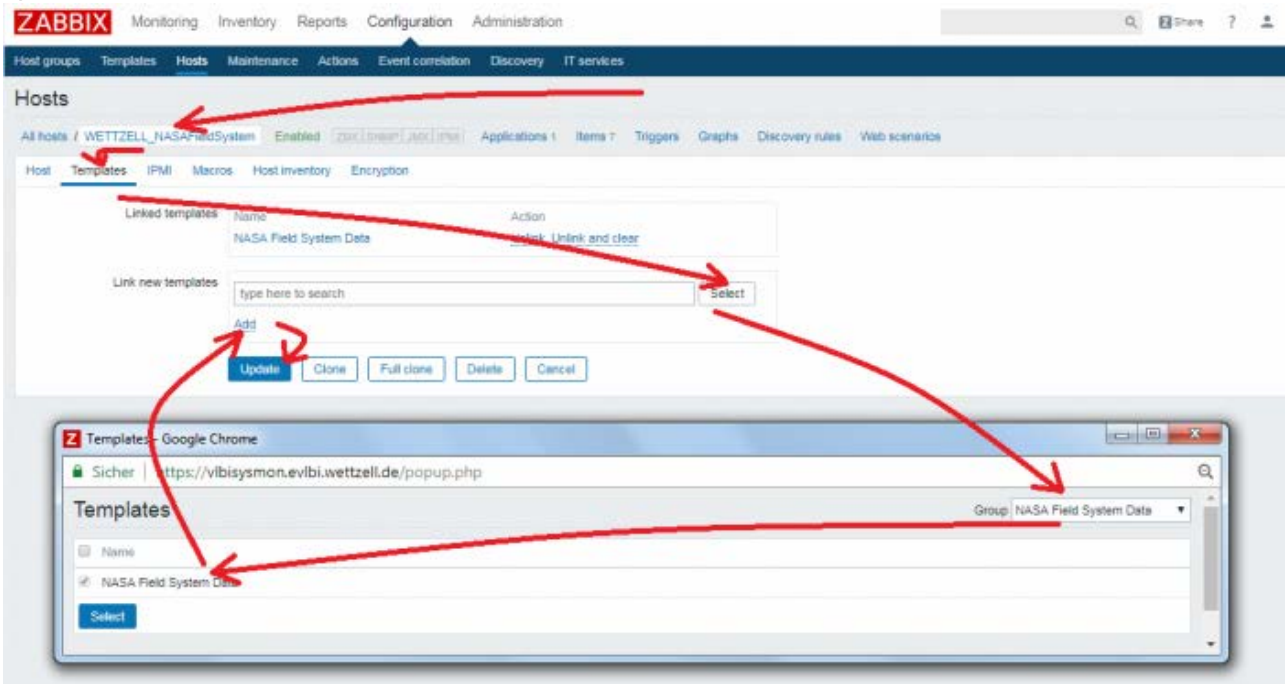
for FileName in "${WebFileNames[@]}"
do
    # Check if file exists
    if [[ ! -f ${WEBPAGES_PATH}${ANTENNA}/${FileName} ]]; then
        continue
    fi
    # Find value and use first match
    COMMAND="grep -o '<!--${PATTERN}-->'
${WEBPAGES_PATH}${ANTENNA}/${FileName}"
    TAGFOUND=`eval $COMMAND`
    if [ "$PATTERN" == "ERC::ERROR_MSG" ]; then
        COMMAND="sed 's/<br>/ \/\/ /g'
${WEBPAGES_PATH}${ANTENNA}/${FileName} | sed -e 's/ERROR//g' | sed -e
's/WARNING//g' | sed -e 's/<font color=\"#FF8040\">/[WARNING] /g' | sed
```

```
-e 's/<font color=\"#F92B06\">/[ERROR] /g' | sed -e 's/<font
color=\"#[a-zA-Z0-9]\">/[ERROR] /g' | sed -e 's/<\/font>//g' | grep -o
'<!--ERC::_MSG-->[^<]*<!-- -->' | sed -e 's/^<!--ERC::_MSG-
->\s*\(.*\)\s*<!--\s*-->/\1/'"
    else
        COMMAND="sed 's/<br>/ \/\/ /g'
${WEBPAGES_PATH}${ANTENNA}/${FileName} | grep -o '<!--${PATTERN}-
->[^<]*<!-- -->' | sed -e 's/^<!--${PATTERN}-->\s*\(.*\)\s*<!--\s*-
->/\1/'"
    fi
    VALUE=`eval $COMMAND`
    COMMAND="grep -o '<!--ERC::TIMECOVERED-->\s<!--\s[^<]*\s-->\s<!-- -
->' ${WEBPAGES_PATH}${ANTENNA}/${FileName} | 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 $TAGFOUND ]]; then
        break
    fi
done
if [[ -z $TAGFOUND ]]; then
    # Tag not found
    echo "1"
    exit 1
fi
if [[ -z $VALUE ]]; then
    # Tag is empty
    echo "0"
    exit 0
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

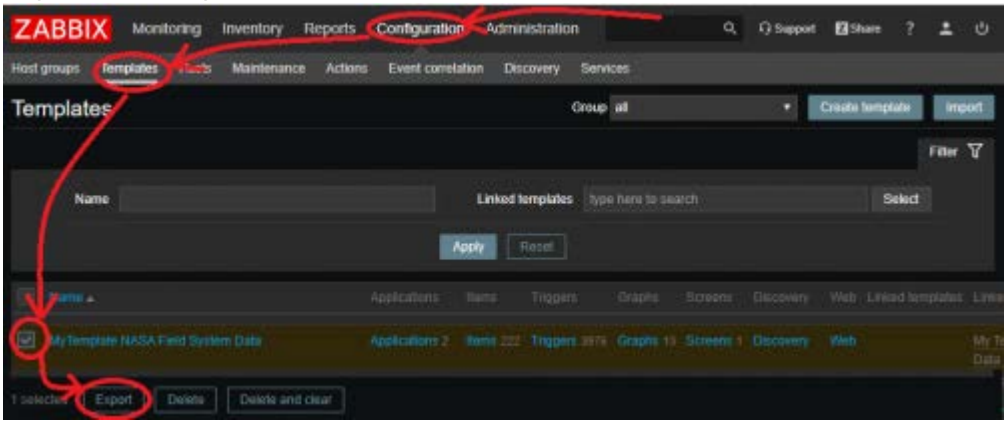
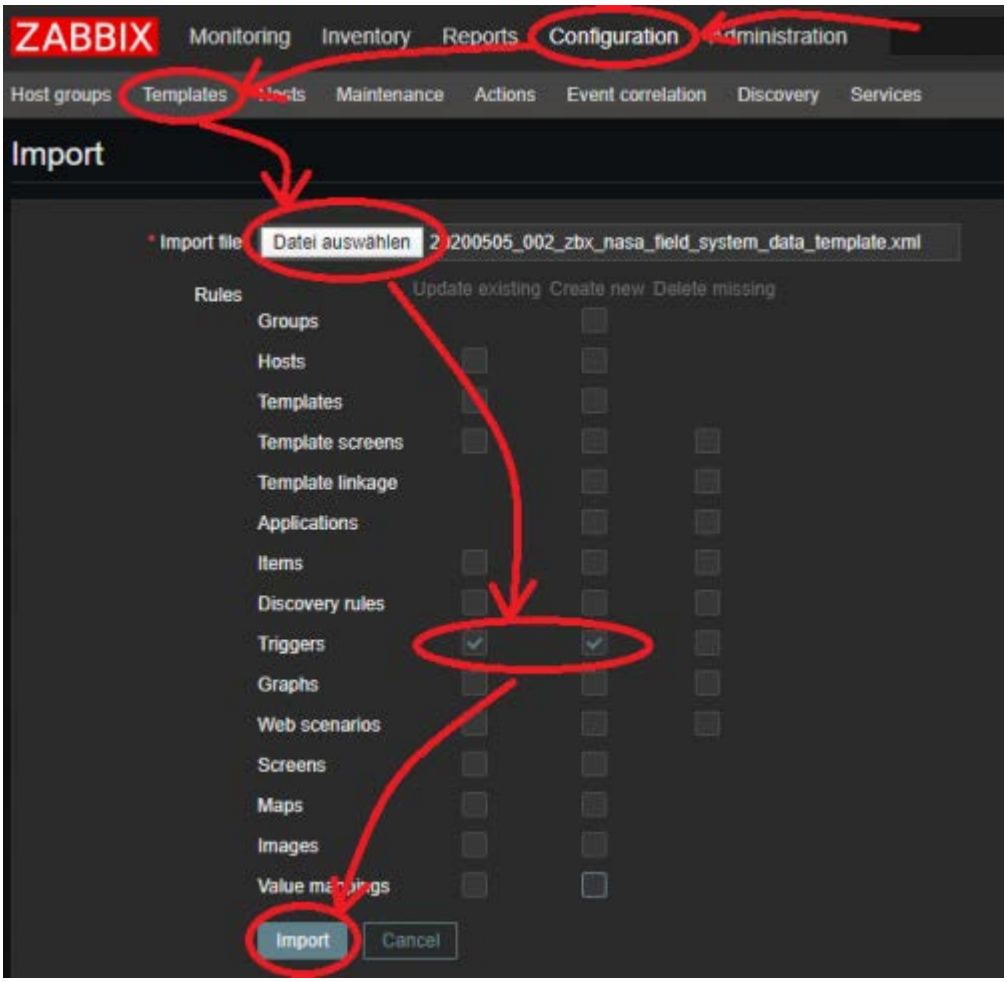
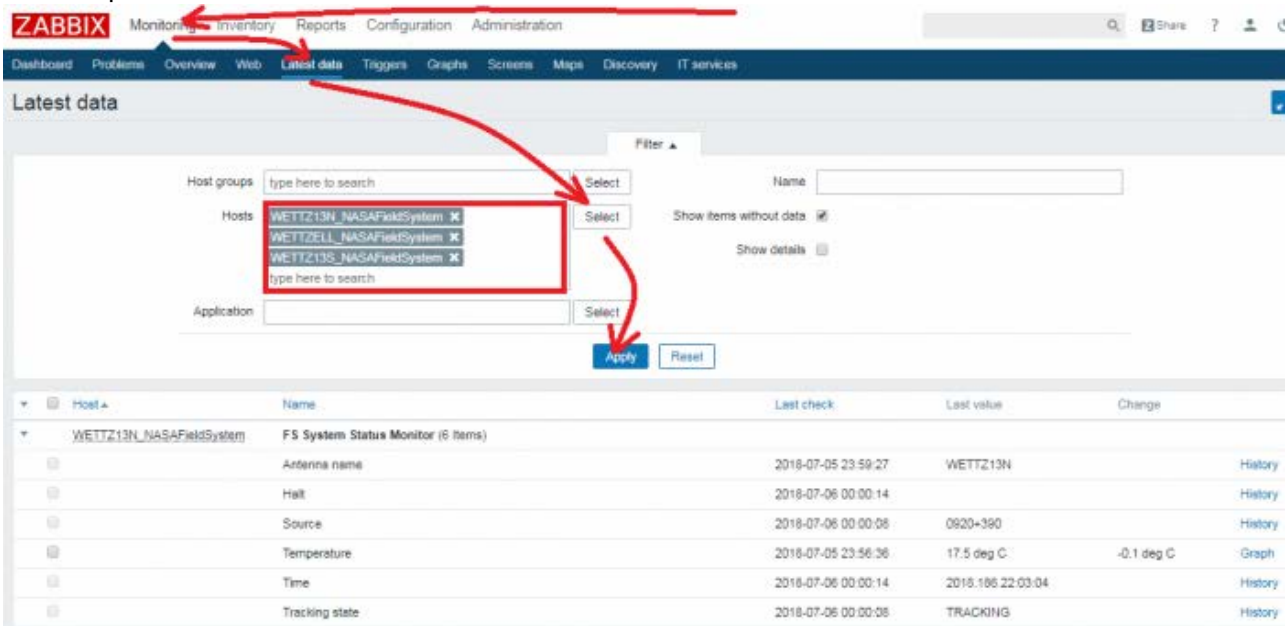
# Send value to ZABBIX
```

```
if [[ "$PATTERN" = "ERC::ACU_GENERAL_ERROR_MSG" || "$PATTERN" =
"ERC::ACU_AZIMUTH_ERROR_MSG" || "$PATTERN" =
"ERC::ACU_ELEVATION_ERROR_MSG" ]]; then
    # Sample from
https://www.tutorialkart.com/bash-shell-scripting/bash-split-string/
    delimiter="//"
    VALUESTR=$VALUE$delimiter
    VALUESTR=`echo $VALUESTR | sed 's/ *$//g'`
    array=();
    while [[ $VALUESTR ]]; do
        VALUE="${VALUESTR%%"$delimiter"*}";
        VALUE=`echo $VALUE | sed 's/^* //g'`
        if [[ ! -z $VALUE && "$VALUE" != "[Azimuth]" && "$VALUE" !=
"[General]" && "$VALUE" != "[Elevation]" ]]; then
            array+=( "${VALUE}" );
            if [[ "$PATTERN" = "ERC::ACU_GENERAL_ERROR_MSG" ]]; then
                VALUE=`echo "[General] $VALUE"`
            fi
            if [[ "$PATTERN" = "ERC::ACU_AZIMUTH_ERROR_MSG" ]]; then
                VALUE=`echo "[Azimuth] $VALUE"`
            fi
            if [[ "$PATTERN" = "ERC::ACU_ELEVATION_ERROR_MSG" ]]; then
                VALUE=`echo "[Elevation] $VALUE"`
            fi
            echo "${HOSTNAME} ERC.ACU_ERROR_MSG ${TIMESTAMP}
\"${VALUE}\"" |  ${HOMEDIR}/zabbix_sender -z 127.0.0.1 -p 10052 -T -i -
> /dev/null 2> /dev/null
        fi
        VALUESTR=${VALUESTR#*"$delimiter"};
        VALUESTR=`echo $VALUESTR | sed 's/ *$//g'`
        if [[ -z $VALUESTR ]]; then
            break
        fi
    done;
    #declare -p array
elif [[ "$PATTERN" = "ERC::ACU_GENERAL_STATUS_MSG" || "$PATTERN" =
"ERC::ACU_AZIMUTH_STATUS_MSG" || "$PATTERN" =
"ERC::ACU_ELEVATION_STATUS_MSG" ]]; then
    delimiter="//"
    VALUESTR=$VALUE$delimiter
    VALUESTR=`echo $VALUESTR | sed 's/ *$//g'`
    array=();
    while [[ $VALUESTR ]]; do
        VALUE="${VALUESTR%%"$delimiter"*}";
        VALUE=`echo $VALUE | sed 's/^* //g'`
        if [[ ! -z $VALUE && "$VALUE" != "[Azimuth]" && "$VALUE" !=
"[General]" && "$VALUE" != "[Elevation]" ]]; then
            array+=( "${VALUE}" );
            if [[ "$PATTERN" = "ERC::ACU_GENERAL_STATUS_MSG" ]]; then
                VALUE=`echo "[General] $VALUE"`
```

```
            fi
            if [[ "$PATTERN" = "ERC::ACU_AZIMUTH_STATUS_MSG" ]]; then
                VALUE=`echo "[Azimuth] $VALUE"`
            fi
            if [[ "$PATTERN" = "ERC::ACU_ELEVATION_STATUS_MSG" ]]; then
                VALUE=`echo "[Elevation] $VALUE"`
            fi
            echo "${HOSTNAME} ERC.ACU_STATUS_MSG ${TIMESTAMP}
\"${VALUE}\"" |  ${HOMEDIR}/zabbix_sender -z 127.0.0.1 -p 10052 -T -i -
> /dev/null 2> /dev/null
        fi
        VALUESTR=${VALUESTR#*"$delimiter"};
        VALUESTR=`echo $VALUESTR | sed 's/ *$//g'`
        if [[ -z $VALUESTR ]]; then
            break
        fi
    done;
    #declare -p array
elif [[ "$PATTERN" = "ERC::ERROR_MSG" ]]; then
    delimiter="//"
    VALUESTR=$VALUE$delimiter
    VALUESTR=`echo $VALUESTR | sed 's/ *$//g'`
    array=();
    PREVIOUS_TIMESTAMP=`/bin/cat
/tmp/${ANTENNA}_LATEST_ERROR_TIMESTAMP.txt 2> /dev/null`
    if [[ -z $PREVIOUS_TIMESTAMP ]]; then
        PREVIOUS_TIMESTAMP=0
    fi
    while [[ $VALUESTR ]]; do
        VALUE="${VALUESTR%%"$delimiter"*}";
        VALUE=`echo $VALUE | sed 's/^* //g'`
        array+=( "${VALUE}" );
        VALUESTR=${VALUESTR#*"$delimiter"};
        VALUESTR=`echo $VALUESTR | sed 's/ *$//g'`
        if [[ -z $VALUESTR ]]; then
            break
        fi
        TESTLOG=`echo $VALUE | grep -P
".*\d+\.\d+\.\d+:\d+:\d+\.\d+\?"`
        if [[ -z $TESTLOG ]]; then
            continue
        fi
        LOGTIMETAG=`echo $VALUE | sed -e 's/.*\]\s\+\(.\+\)?.*/\1/'`
        VALUE=`echo $VALUE | sed -e 's/\(.*\]\)\s\+.\+?\s*\(.*\)/\1
\2/'`
        #echo $LOGTIMETAG # 2020.120.19:37:35.87
        COMMAND="echo '${LOGTIMETAG}' | sed -e
's/^\([0-9]\+\)\.[0-9]\+\.[0-9]\+:[0-9]\+:[0-9]\+.*/\1/'"
        YEAR=`eval $COMMAND`
        #echo $YEAR
```

```
        COMMAND="echo '${LOGTIMETAG}' | sed -e
's/^[0-9]\+\.[0]*\([0-9]\+\)\.[0-9]\+:[0-9]\+:[0-9]\+.*/\1/'"
        DOY=`eval $COMMAND`
        DOY=`echo $((--DOY))`
        #echo $DOY
        COMMAND="echo '${LOGTIMETAG}' | sed -e
's/^[0-9]\+\.[0-9]\+\.\([0-9]\+\):[0-9]\+:[0-9]\+.*/\1/'"
        HOUR=`eval $COMMAND`
        #echo $HOUR
        COMMAND="echo '${LOGTIMETAG}' | sed -e
's/^[0-9]\+\.[0-9]\+\.[0-9]\+:\([0-9]\+\):[0-9]\+.*/\1/'"
        MINUTE=`eval $COMMAND`
        #echo $MINUTE
        COMMAND="echo '${LOGTIMETAG}' | 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 $SECOND ]]; then
            LOGTIMESTAMP=$(date +%s)
        else
            LOGTIMESTAMP=$(date -d "${DOY} days ${YEAR}-01-01
${HOUR}:${MINUTE}:${SECOND}" +"%s")
        fi
        #echo $LOGTIMESTAMP
        if [ $LOGTIMESTAMP -gt $PREVIOUS_TIMESTAMP ]; then
            echo "${HOSTNAME} ${KEY} ${LOGTIMESTAMP} \"${VALUE}\"" |
${HOMEDIR}/zabbix_sender -z 127.0.0.1 -p 10052 -T -i - > /dev/null 2>
/dev/null
            echo "${LOGTIMESTAMP}" >
/tmp/${ANTENNA}_LATEST_ERROR_TIMESTAMP.txt
            PREVIOUS_TIMESTAMP=$LOGTIMESTAMP
        fi
    done;
    #declare -p array
else
    echo "${HOSTNAME} ${KEY} ${TIMESTAMP} \"${VALUE}\"" |
${HOMEDIR}/zabbix_sender -z 127.0.0.1 -p 10052 -T -i - > /dev/null 2>
/dev/null
fi

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”).



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



- Now everything is prepared to activate the predefined template "*MyTemplate NASA Field System Data*" to read all available data from the web pages of a NASA Field System. The current version is this: 20200505_002_zbx_nasa_field_system_data_template.xml (an extended template "*My Template NASA Field System Data with Vertex ACU*" with specific add-ons for the Vertex Antenna Control JUnit is here:
  20200502_zbx_nasa_field_system_data_with_vertex_ace_template.xml
  ; it adds error codes and additional triggers and includes the regular "*MyTemplate NASA Field*

System Data")



- 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.

- It also creates triggers for each NASA Field System error defined in "*/usr2/fs/control/fserr.ctl* ". If new errors are defined in new releases of the Field System, the following script "*convert_fserrors.pl*" can be used to create the trigger section of the XML-file for the template. Start it with "convert_fserrors.pl <PATH_TO_CONTROL_FILE>".

```
#!/usr/bin/perl

use strict;
use Switch;

if (($#ARGV+1) == 0) {
    print STDERR "No error message control file from NASA Field System
as argument\n";
    exit 1;
}
my $FilePath = $ARGV[0];

print "    <triggers>\n";
if (!(open(FHIN, '<', $FilePath))) {
    return 1;
}
if (tell(FHIN) == -1) {
    return 1;
}
```

```
my $State = 0;
my $ErrorCodeRegExp = "";
my $ErrorCode = "";
my $ErrorText = "";
my $line;
my $LineNumber = 0;
my $NumberOfTriggers = 0;
my %FoundErrors;
foreach $line (<FHIN>) {
    #print $line;
    $LineNumber = $LineNumber + 1;
    $line = $line =~ s/>/&gt;/r;
    $line = $line =~ s/</&lt;/r;
    $line = $line =~ s/&/&amp;/r;
    switch($State) {
        if ($line =~ /^\s*""?\s*$/) {
            $State = 0;
            next;
        }
        case 0  {
            if ($line =~ /^\w\w\s+-?\d+/) {
                my $ErrorLetterCodeLetter1;
                my $ErrorLetterCodeLetter2;
                my $ErrorNumberCode;
                $State = 1;
($ErrorLetterCodeLetter1,$ErrorLetterCodeLetter2,$ErrorNumberCode) =
($line =~ /^(\w)(\w)\s+(-?\d+)/);
                $ErrorCodeRegExp =
"[".(lc($ErrorLetterCodeLetter1)).(uc($ErrorLetterCodeLetter1))."]"."["
.(lc($ErrorLetterCodeLetter2)).(uc($ErrorLetterCodeLetter2))."]"."\\s+"
.$ErrorNumberCode;
                $ErrorCode =
$ErrorLetterCodeLetter1.$ErrorLetterCodeLetter2." ".$ErrorNumberCode;
            }
        }
        case 1  {
            chomp ($line);
            $ErrorText = $ErrorCode.": ".$line." (VLBI
Beobachtungsfehler)";
            if(exists($FoundErrors{$ErrorText})) {
                print STDERR "Error already found: again in line
".$LineNumber."\n";
                print STDERR $ErrorText."\n";
                exit 1;
            }
            $FoundErrors{$ErrorText} = $ErrorCodeRegExp;
            print "        <trigger>\n";
            print "            <expression>({MyTemplate NASA Field
System
Data:ERC.ERROR_MSG.regexp(&quot;".$ErrorCodeRegExp."&quot;,180)}=1) and
```

```
({MyTemplate NASA Field System
Data:ERC.ERROR_MSG.regexp(&quot;\[WARNING\]&quot;,180)}=1) and
({MyTemplate NASA Field System
Data:ERC.SCHEDULE.regexp(&quot;none&quot;,180)}=0)</expression>\n";
            print "            <recovery_mode>0</recovery_mode>\n";
            print "            <recovery_expression/>\n";
            print "            <name>NASAFS INFO
".$ErrorText."</name>\n";
            print "
<correlation_mode>0</correlation_mode>\n";
            print "            <correlation_tag/>\n";
            print "            <url/>\n";
            print "            <status>1</status>\n";
            print "            <priority>1</priority>\n";
            print "            <description/>\n";
            print "            <type>0</type>\n";
            print "            <manual_close>0</manual_close>\n";
            print "            <dependencies/>\n";
            print "            <tags/>\n";
            print "        </trigger>\n";
            print "        <trigger>\n";
            print "            <expression>({MyTemplate NASA Field
System
Data:ERC.ERROR_MSG.regexp(&quot;".$ErrorCodeRegExp."&quot;,180)}=1) and
({MyTemplate NASA Field System
Data:ERC.ERROR_MSG.regexp(&quot;\[ERROR\]&quot;,180)}=1) and
({MyTemplate NASA Field System
Data:ERC.SCHEDULE.regexp(&quot;none&quot;,180)}=0)</expression>\n";
            print "            <recovery_mode>0</recovery_mode>\n";
            print "            <recovery_expression/>\n";
            print "            <name>NASAFS WARNING
".$ErrorText."</name>\n";
            print "
<correlation_mode>0</correlation_mode>\n";
            print "            <correlation_tag/>\n";
            print "            <url/>\n";
            print "            <status>1</status>\n";
            print "            <priority>2</priority>\n";
            print "            <description/>\n";
            print "            <type>0</type>\n";
            print "            <manual_close>0</manual_close>\n";
            print "            <dependencies/>\n";
            print "            <tags/>\n";
            print "        </trigger>\n";
            #print "MIST ".$ErrorCodeRegExp." ".$ErrorText."\n";
            $NumberOfTriggers = $NumberOfTriggers + 1;
        }
        else {
            print "[ERROR] Wrong state\n";
            goto FINISH;
```

```
        }
    }
}

FINISH:
close (FHIN);
print "    </triggers>\n";
print STDERR "============================\n";
print STDERR "Number of triggers: ".$NumberOfTriggers."\n";
print STDERR "============================\n";
```

- Export the template
- 

- Replace the section “<triggers> … </triggers>” with the newly generated version and import it again.
- 

- 

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

- The current test antennas are
  - 

2.2.7) 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"> </font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="center"
width="50"> <font color="#000000"> </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-
-> </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-
-> </font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="160"> <font color="#000000"><!--ERC::TIME_MODULEA-
-> </font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="160"> <font color="#000000"><!--ERC::REMEININGGB_MODULEA-
-> </font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="100"> <font color="#000000"><!--
ERC::REMEININGPERCENT_MODULEA--> </font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="left"
width="160"> <font color="#000000"><!--ERC::CHECKTIME_MODULEA-
-> </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-
-> </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-
-> </font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="160"> <font color="#000000"><!--ERC::TIME_MODULEB-
-> </font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="160"> <font color="#000000"><!--ERC::REMEININGGB_MODULEB-
-> </font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="right"
width="100"> <font color="#000000"><!--
ERC::REMEININGPERCENT_MODULEB--> </font></td>
  <td bgcolor="#FFFFFF" style="font-size:11pt;" align="left"
width="160"> <font color="#000000"><!--ERC::CHECKTIME_MODULEB-
-> </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 "  "




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;

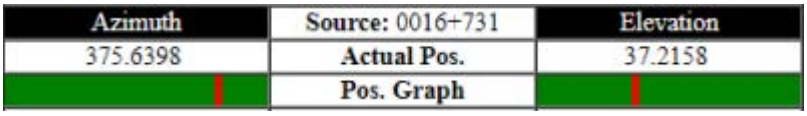
    /* printf ("usSRCSTACUGetStatus\n"); */

    /// 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:

- 

3) Data Archive

- All data are stored for at least 3 to 6 month in the database. They must be extracted for a long-term archive. This can be done with a Python script. The original software is:
  zabbixapi.tar.gz
  . The latest code is
  ZabbixAPI.py (2020-10-06)
  with the configuration file
  config.zip
  (the code repository is here: http://xsamba.wtz/svn/vlbi/trunk/scripts/ZabbixAPI/).
- It is automatically used to produce daily data files. Only data which are registered to be extracted are used to produce such files.
- Manually, it can be used to extract data for specific items. It supports the following features:

```
python3 ZabbixAPI.py --help
usage: ZabbixAPI.py [-h] [-ASC] [-C CONFFILEPATH] [-DESC] [-f] [-F
FILENAMEOUT] [-H HOST] [-I ITEM] [-K KEY] [-L]
                    [-LIM LIMIT] [-TS STARTDATE] [-TE STOPDATE] [-VL
VERBOSITYLEVEL]

Fetch data from Zabbix monitoring service of the IVS Seamless Auciliary
Data Archive

optional arguments:
  -h, --help          show this help message and exit
  -ASC                Order values ascending (together with -K/-I and -
L)
  -C CONFFILEPATH     Use specified Configuration file
  -DESC               Order values descending (together with -K/-I and
-L)
  -f                  Print output to file with standard filename
  -F FILENAMEOUT      Print output to file with individual filename
  -H HOST             Select host with hostname or host ID
  -I ITEM             Select item with ID
  -K KEY              Select item with key
  -L                  Show a list of hosts, items (together with -H),
or values (-K/-I)
  -LIM LIMIT          Limit the number of requested records
  -TS STARTDATE       Start time of records ["YYYY-MM-DD HH:MM:SS"]
  -TE STOPDATE        Stop time of records ["YYYY-MM-DD HH:MM:SS"]
  -VL VERBOSITYLEVEL  Verbosity level [0=off, 1=basic, 2=detailled]
```

- Sample calls are:

```
1) Get help
python[.exe] ZabbixAPI.py
```

```
or
python[.exe] ZabbixAPI.py --help

2) List available hosts
python[.exe] ZabbixAPI.py -L

3) Validate host ID or hostname
python[.exe] ZabbixAPI.py -H 10192
(where 10192 is the host ID)
or
python[.exe] ZabbixAPI.py -H OHIGGINS_000_NASAFieldSystem
(where OHIGGINS_000_NASAFieldSystem is the hostname)

4) Get list of available items of a specific host
python[.exe] ZabbixAPI.py -H 10192 -L
(where 10192 is the host ID)
or
python[.exe] ZabbixAPI.py -H OHIGGINS_000_NASAFieldSystem -L
(where OHIGGINS_000_NASAFieldSystem is the hostname)

5) Validate item ID of a specific hostname
python[.exe] ZabbixAPI.py -H 10192 -I 29759
(where 10192 is the host ID and 29759 is the item ID)

6) Validate item key of a specific hostname
python[.exe] ZabbixAPI.py -H 10192 -K ERC.TIME
(where 10192 is the host ID and ERC.TIME is the item key)

7) Get list of values for a specific item ID or key of a specific host
python[.exe] ZabbixAPI.py  -H  WETTZELL_000_NASAFieldSystem -I 26733 -L
(where WETTZELL_000_NASAFieldSystem is the hostname and 26733 is the
item ID)
or
python[.exe] ZabbixAPI.py  -H 10133 -K ERC.TEMPERATURE -L
(where 10133 is the hostname and ERC.TEMPERATURE is the item key)

8) Get ascending list of values for a specific item ID or key of a
specific host
python[.exe] ZabbixAPI.py  -H 10133 -K ERC.TEMPERATURE -L -ASC
(where 10133 is the hostname and ERC.TEMPERATURE is the item key)

9) Get descending list of values for a specific item ID or key of a
specific host
python[.exe] ZabbixAPI.py  -H 10133 -K ERC.TEMPERATURE -L -DESC
(where 10133 is the hostname and ERC.TEMPERATURE is the item key)

10) Get list with limited number of records for a specific item ID or
key of a specific host
python[.exe] ZabbixAPI.py  -H 10133 -K ERC.TEMPERATURE -L -LIM 15
(where 10133 is the hostname, ERC.TEMPERATURE is the item key, and 15
```

```
is the max number of records)

11) Get list of values for a specific item ID or key of a specific host
for one specific day
python[.exe] ZabbixAPI.py  -H 10133 -K ERC.TEMPERATURE -L -TS
"2020-09-30"
(where 2020-09-30 is the specific calendar day)

12) Get ist of values for a specific item ID or key of a specific host
for time interval
python[.exe] ZabbixAPI.py  -H 10133 -K ERC.TEMPERATURE -L -TS
"2020-09-30 00:00:00" -TE "2020-09-30 01:30:00"
(where 2020-09-30 00:00:00 is the start time and 2020-09-30 01:30:00 is
the stop time)

13) Use a specific configuration file
python[.exe] ZabbixAPI.py -H 10192 -L -C myconfig.ini
(where 10192 is the host ID and myconfig.ini is the configuration file
name with path)

14) Write output to file with standard name "output.txt"
python[.exe] ZabbixAPI.py  -H 10133 -K ERC.TEMPERATURE -L -TS
"2020-09-30" -f

15) Write output to file with individual, user-defined name
python[.exe] ZabbixAPI.py  -H 10133 -K ERC.TEMPERATURE -L -TS
"2020-09-30" -F Meteo_Temp.txt
(where Meteo_Temp.txt is a sample filename)

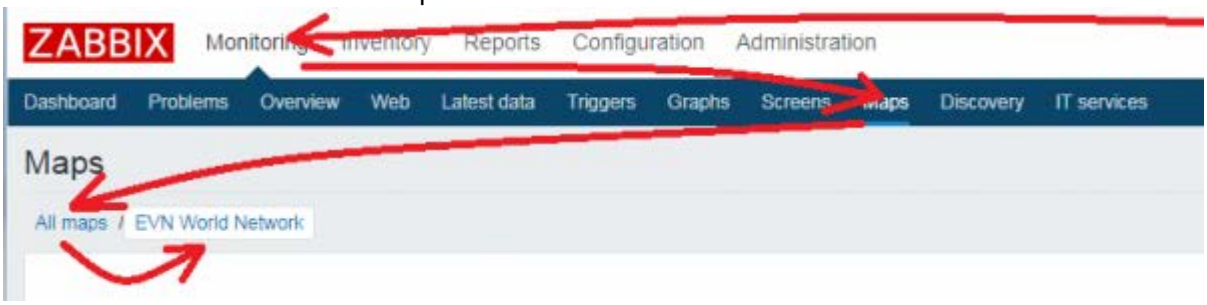
16) Switch verbosity level on
python[.exe] ZabbixAPI.py -H 10192 -VL 2
(where 10192 is the host ID and 2 is the verbosity level; higher number
means more details)
```

- A sample call to read the ten newest values is here:

- ```
  python.exe ZabbixAPI.py -H 10191 -I 29466 -L
  List of values of current item 29466 in current host: 10191

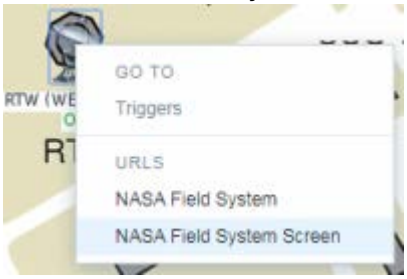
  #Date                  Unixtime    Value
  #-------------------------------------------------
   2020-10-06 12:17:01   1601986621  10.4000
   2020-10-06 12:16:01   1601986561  10.3000
   2020-10-06 12:15:01   1601986501  10.3000
   2020-10-06 12:14:02   1601986442  10.3000
   2020-10-06 12:13:02   1601986382  10.3000
   2020-10-06 12:12:02   1601986322  10.1000
   2020-10-06 12:11:02   1601986262  10.1000
   2020-10-06 12:10:01   1601986201  10.0000
   2020-10-06 12:09:01   1601986141  10.0000
  ```
  ```

```
2020-10-06 12:08:01   1601986081  9.9000
```

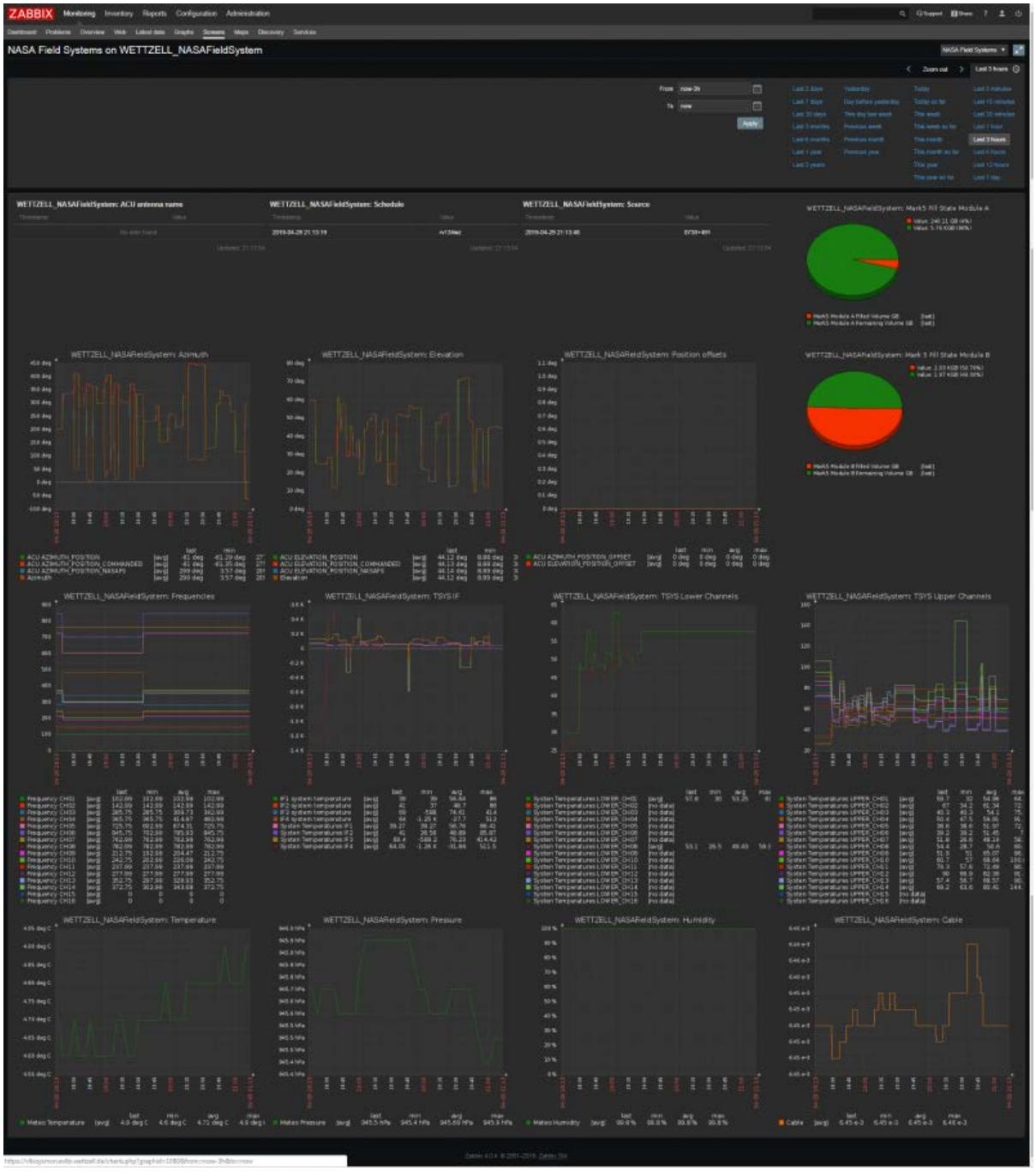
4) Accessing data and web pages by antenna staff

4.1) ZABBIX web page

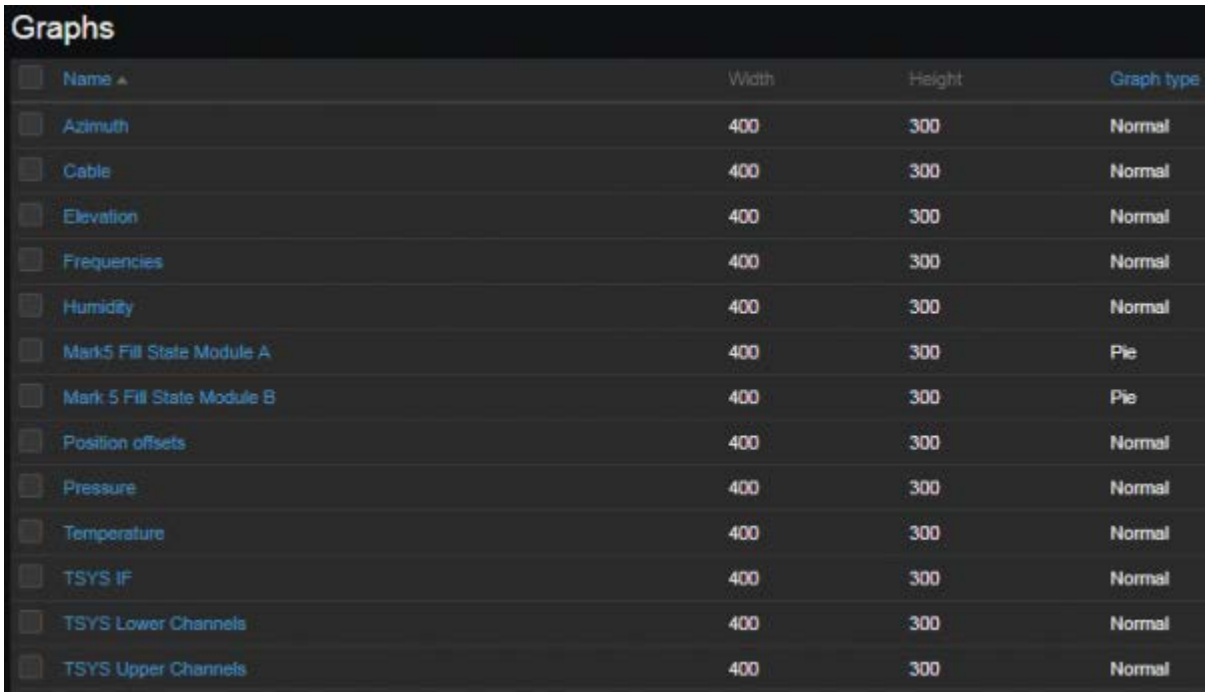
- After a positive feedback, you are now able to access the data pages of the monitoring system.
- You received a username and password. It can be used to login to the central monitoring page and to your specific copy of the NASA Field System pages. Just connect to https://vlbisysmon.evlbi.wettzell.de.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

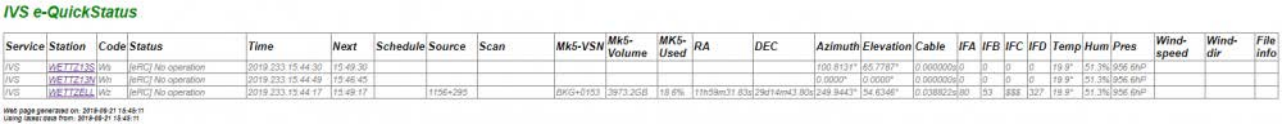


- Collected are 110 values: see 20180816valuesavailable_latestdata.pdf
  plus 110 bool values showing if the request of each value succeeded
- The following graphs are predefined



4.2) IVS/(EVN) Quick Status report web page

- Having all data available at one central server, a network quick status information can be generated, showing the status of all telescopes in one table under https://vlbisysmon.evlbi.wettzell.de/monitoring_archive/fs_web_pages/:
- A sample looks like this:



- e-RemoteCtrl directly supports the sending of status files "*eQuickStatusReport.txt*":

```
<eQuickStatusInfo>
   Service = IVS
   Stationname = WETTZELL
   StationIVSCode = Wz
   Schedule =
   Status = [eRC] No operation
   DateTime = 2019.233.15:48:21
   TimeNext = 15:49:17
   Source = 1156+295
   Scan =
   Mark5VSN = BKG+0153
   Mark5Volume = 3973.2
   Mark5Used = 18.6
   RightAscension = 11h59m31.83s
   Declination = 29d14m43.80s
   Azimuth = 251.0241
   Elevation =  54.0070
   CableDelay = 0.038822
   SystemTemperatureIFA = 80
```

```
    SystemTemperatureIFB = 53
    SystemTemperatureIFC = 123
    SystemTemperatureIFD = 327
    MeteorologyTemperature = 19.9
    MeteorologyHumidity =   50.5
    MeteorologyPressure =   956.6
    MeteorologyWindSpeed = 6.12
    MeteorologyWindDirection = 66
</eQuickStatusInfo>
```

- The quick status files from the different telescopes are renamed to have the extension "*.info*" on the central data server and are interpreted by the program "quickstatus" to prepare the table of the status web page. The source code is located at "*/home/oper/Software/vlbisysmon/utils/quickstatus/* ". The executable is in "*/home/oper/Software/vlbisysmon/bin/* " with a configuration file in "*/home/oper/Software/vlbisysmon/control/* ". The source code is under version control at the Wettzell observatory: http://xsamba.wtz/svn/vlbi/trunk/code/quickstatus/ (just internally available)
- The program continuously runs and is started with a cronjob script installed with "crontab -e"

```
 *    *    *    *    *
/home/oper/Software/vlbisysmon/bin/create_quickstatus.sh
```

- The start script is the following:

```
SERVICE='quickstatus'
PROGPATH='/home/oper/Software/vlbisysmon/bin/'

SERVICETEST="$SERVICE"

#/bin/sleep 1
if /bin/ps ax | /bin/grep -v grep | /bin/grep -v ".sh" | /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}
/home/oper/Software/vlbisysmon/control/equickstatus.conf &
fi

exit 0
```

4.3) Field System Monitoring web page

- Beside a local acces to the NASA Field System Monitoring web page, it can also be accessed via

Internet on the following location: https://vlbisysmon.evlbi.wettzell.de/monitoring_archive/fs_web_pages/<ANTENNA>/, where <ANTENNA> is the site name

- The output is the following page but with just updates of one second

4.4) Data archive web page

- Archived files can be downloaded without user credentials from https://vlbisysmon.evlbi.wettzell.de/monitoring_archive/<ANTENNA>/<YEAR>/<MONTH>/<ITEM>/<DAY>_<ANTENNA>_<ITEM>.txt
- A file content looks like this:

vlbisysmon.evlbi.wettzell.de/monitoring_archive/WETTZELL/2020/09/MeteoTemperature/20200919_WETTZELL_MeteoTemperature.txt

```
1600473702      2020-09-19 00:01:42     11.1000
1600474002      2020-09-19 00:06:42     10.7000
1600474300      2020-09-19 00:11:40     10.0000
1600474601      2020-09-19 00:16:41     9.7000
1600474900      2020-09-19 00:21:40     9.6000
1600475201      2020-09-19 00:26:41     9.3000
1600475502      2020-09-19 00:31:42     9.3000
1600475800      2020-09-19 00:36:40     9.1000
1600476101      2020-09-19 00:41:41     9.4000
1600476400      2020-09-19 00:46:40     9.6000
1600476702      2020-09-19 00:51:42     9.7000
1600477002      2020-09-19 00:56:42     9.8000
1600477300      2020-09-19 01:01:40     9.9000
1600477601      2020-09-19 01:06:41     9.9000
1600477900      2020-09-19 01:11:40     10.0000
1600478202      2020-09-19 01:16:42     10.0000
1600478502      2020-09-19 01:21:42     9.8000
1600478801      2020-09-19 01:26:41     9.6000
1600479101      2020-09-19 01:31:41     9.5000
1600479400      2020-09-19 01:36:40     9.6000
1600479701      2020-09-19 01:41:41     9.8000
1600480000      2020-09-19 01:46:40     9.7000
1600480301      2020-09-19 01:51:41     9.3000
1600480600      2020-09-19 01:56:40     9.2000
1600480902      2020-09-19 02:01:42     8.6000
1600481202      2020-09-19 02:06:42     8.7000
1600481501      2020-09-19 02:11:41     8.6000
1600481800      2020-09-19 02:16:40     8.8000
1600482102      2020-09-19 02:21:42     9.0000
1600482401      2020-09-19 02:26:41     8.6000
1600482701      2020-09-19 02:31:41     8.6000
```

5) 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.)

IVS Seamless Auxiliary Data Archive

Quick reference / manual

Info: A sample script can be requested from Stuart Weston (stuart.weston@aut.ac.nz).

You need a separate keyfile for the SSH connection which you have to request by email (alexander.neidhardt@tum.de). If you want to use your own, please send your public key out to alexander.neidhardt@tum.de.

What you really need as a basic user are the following steps (basic user):

---

1) Data sending

You can do the data sending with the following sample call for temperature:
UnixTime=$(date +%s); ssh -i hobart hobart@141.74.6.20 "echo 'HOBART_000_NASAFieldSystem ERC.TEMPERATURE $UnixTime \"0.0\"' | ./bin/zabbix_sender -z 141.74.6.20 -p 10052 -T -i -"
Explanation:
Unixtime .... => creates your local time, so that the data transfer does not influence the data injection. Your local time should be in UTC. If not please change the date command.
ssh ... => is the connection with your key file "hobart" and the user "hobart"
echo ... = is the table of injected data for the ZABBIX host "HOBART_000_NASAFieldSystem"
./bin/zabbix_sender ... => is the injection call
You just have to change "\0.0\" to the real value, e.g. "\12.5\" for 12.5 degree Celsius, each time you start the command.

The required key files for your met data are:
ERC.TEMPERATURE => Temperature => unit deg C
ERC.HUMIDITY => Humidity => unit %
ERC.PRESSURE => Pressure => unit hPa
ERC.WINDSPEED => Windspeed => unit km/h
ERC.WINDDIRECTION => Winddirection => unit degree

And for clock offset (GPS-FMOUT):
ERC.DOTMON => Dotmon Clock Offsets (GPSminusFMOUT) => unit usec

Attention: Input data should always be in the form with max 4 places after the floating point, e.g. for clock offset 3.8 (the unit is 10e-6 sec). If it does not fit, you have to shift the value according to the units or I have to change the unit or a preprocessing individually, which should be avoided. Maximum update rate is 1 second. We defined 1 minute updates for seamless auxiliary data.

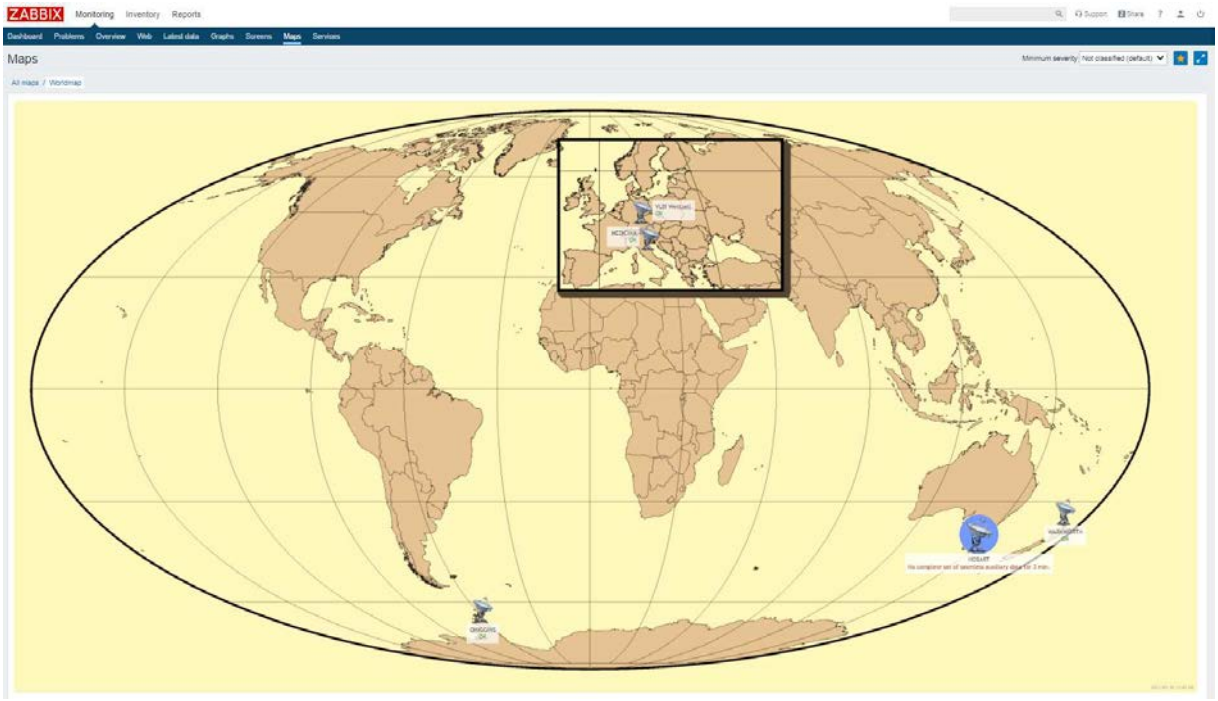
You should see something like this, if everything works:
2020-09-1601302693 14:18:13 Welcome, hobart. Type 'help' for information.
info from server: "processed: 1; failed: 0; total: 1; seconds spent: 0.000048" sent: 1; skipped: 0; total: 1

---

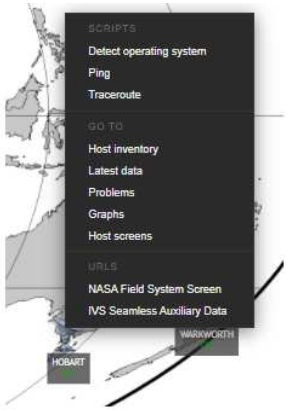
2) Data view

Open https://vlbisysmon.evlbi.wettzell.de/zabbix/
User: IVS Guest
Pwd: IVS Guest

You should see a world map. If not, please select Monitoring => Maps => Worldmap.



Search your position and click on it. You should see:



Select IVS Seamless Auxiliary Data for all available graphs or select "Graphs" for pages with selectable, individual graphs. The use should be quite intuitive.

3) Data extraction

There is currently a Python-script "ZabbixAPI.py", which is attached. It requires the config.ini in the same folder. The use is described when calling "python3 ZabbixAPI.py --help" (or "python.exe ..." under Windows)

Please let me (alexander.neidhardt@tum.de) know, if you have some problems.

4) Full support as standard user with e-RemoteCtrl

If you want full support of a standard user (as you have experience with e-RemoteCtrl), please also let me know and we prepare the latest version of e-RemoteCtrl, but the basic user setup should be enough for elementary data in the auxiliary seamless data archive for now.