geneko

GWR

User Application Examples

on Geneko Cellular Routers

Document version: 1.2
Date: September 2016

Abstract

This document describes user application examples delivered on Geneko Cellular Routers.

Document Control

Document:	**Version**	**1.2**
	File	GWR User Application Examples 2016 Sep Rev C.pdf
	Status	Valid

Creation:	**Role**	**Name**
	Author	Tanja Savic
	Review	C

Approval:	**Role**	**Name**
	Team Leader	Darko Kojic

Trademark

Content

List of Figures

1. Introduction

Geneko Cellular Routers (GWR-HS, GWR362, GWR462, GWRI-362, GWRI-462 and GWG models) support use of user applications, thereby extending standard router functionality. Supported languages for user applications are:

- Shell scripts (sh, bash, dash)
- LUA
- C/C++
- Perl
- Python (on GWR-HS, GWR362, GWR462, GWRI-362 and GWRI-462 models)

User applications are stored in a permanent location and can be started on system startup. They are executing in a chroot environment. A chroot environment is an operating system call that will change the root location temporarily to a new folder. Chroot runs a command or an interactive shell from another directory and treats that directory as root directory.

1.1 Purpose

The purpose of this document is to show how to use existing user application examples on Geneko Cellular Routers.

Intended audience is customer technical engineers and management interested in detailed progress information.

2. User application environment

Use Putty, Secure CRT, etc on Windows, or Putty, Minicom, GTKterm or your favorite Linux terminal on Linux for connection over serial RS-232 port or SSH over LAN port.

After logging in as admin the user should call gwr_chroot command to activate user application environment.

For example: Use SSH to enter in global configuration mode.
ssh 192.168.1.1 //ssh to br0 at TCP PORT 22

Login as: **admin**
admin@192.168.1.1's password:**admin**
admin@geneko>**gwr_chroot**

There is possibility to user write his/her own examples. In directory **/home/admin** user can make new directories, write scripts, applications to extend router functionality. In folder startup are stored scripts, that need to be started during the startup of the router.

geneko:~# **cd /home/admin**
geneko:~# **ls**
startup/

All exiting examples are stored in directory **/home/examples**.

geneko:~# **cd /home/examples/**
geneko:~#**ls**

```
geneko:/home/examples# ls
ftp_connect.sh*                          router_network_configuration_example.sh*
get_modem_info.sh*                       send_e-mail.sh*
gpio_functionality.sh*                   serial_read_example.sh*
gpio_send_sms.sh*                        sms_send.sh*
read_sms_perform_action.sh*              tcp_server_example.sh*
remove_sim_card_pin.sh*                  tcp_server_example.tcpsvd.sh*
report_ppp_status.sh*
```

Figure 1 – GWG Gateway display of examples

3. Examples

3.1 Connection and access of FTP server

In this example is described configuration of FTP connection.

geneko:/home/examples# **vi ftp_connect**

```
#!/bin/bash

#(c) 2015 GENEKO

#GENEKOSH Example: Connect and access of FTP server

#
# Server port where this script listens
#
LISTEN_PORT=3222

IP_address=0.0.0.0

#Establish connection
tcpsvd -vE $IP_address $LISTEN_PORT ftpd -w

~
```

Figure 2 – ftp_connect script

geneko:/home/examples# **cp /home/examples/ftp_connect /home/admin/startup/**
geneko:/home/examples# **reboot**

Use FileZilla client for connection and access of FTP server.

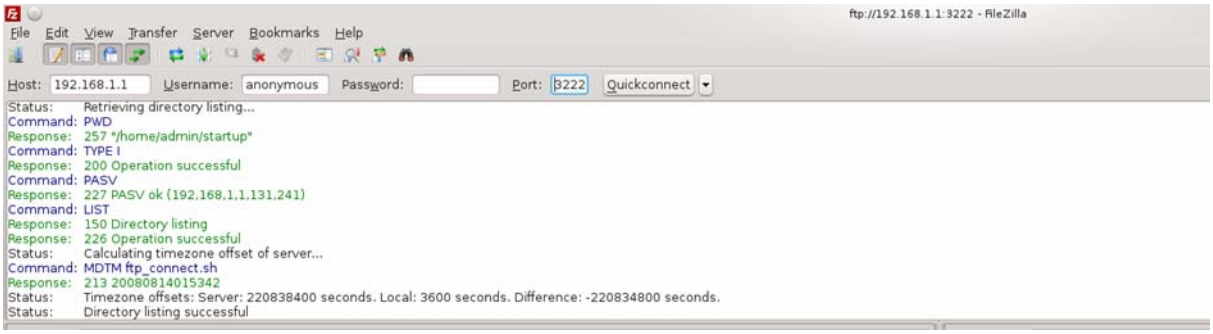

Figure 3 – FileZilla client

3.2 Get modem info

In this example is described basic modem information, network operator, supported AT commands.

geneko:/home/examples#**./get_modem_info**

```
Thank you for using GENEKOSH scripting

Cinterion
PHS8-E
REVISION                03.001

SIM card:                READY
Network operator:        "MTS"
Connected to network:    YES

Supported AT commands:

AT+CLAC       AT&C          AT&D          AT&F          AT&S          AT&V          AT&W
ATE           ATI           ATL           ATM           ATQ           ATV           ATX
ATZ           ATT           ATP           AT\Q          ATD           ATA           ATH
AT0           ATS0          ATS3          ATS4          ATS5          ATS6          ATS7
ATS8          ATS10         AT+IPR        AT+CMUT       AT+CMUX       AT+GMI        AT+GMM
AT+GMR        AT+GCAP       AT+GSN        AT+CMEE       AT+WS46       AT+CFUN       AT+CBST
AT+CRLP       AT+CSSN       AT+CREG       AT+CGREG      AT+GCAP       AT+CSCS       AT+CSTA
AT+CR         AT+CEER       AT+CRC        AT+CGDCONT    AT+CGEQREQ    AT+CGEQMIN    AT+CGQREQ
AT+CGQMIN     AT+CGPADDR    AT+CGDATA     AT+CGSMS      AT+CSMS       AT+CMGF       AT+CSCA
AT+CSMP       AT+CSDH       AT+CSCB       AT+ES         AT+ESA        AT+CSQ        AT+CPAS
AT+CPIN       AT+CGATT      AT+CGACT      AT+CGCMOD     AT+CPBS       AT+CPBR       AT+CPBF
AT+CPBW       AT+CSVM       AT+CPMS       AT+CNMI       AT+CMGL       AT+CMGR       AT+CMGS
AT+CMSS       AT+CMGW       AT+CMGD       AT+CMGC       AT+CNMA       AT+CMMS       AT+CHUP
AT+CCFC       AT+CCUG       AT+COPS       AT+CLCK       AT+CPWD       AT+CUSD       AT+CAOC
AT+CACM       AT+CAMM       AT+CPUC       AT+CCWA       AT+CHLD       AT+CIMI       AT+CGMI
AT+CGMM       AT+CGMR       AT+CGSN       AT+CNUM       AT+CSIM       AT+CRSM       AT+CCLK
AT+CLCC       AT+COPN       AT+CPOL       AT+CPLS       AT+CLIP       AT+COLP       AT+CLIR
AT+VTS        AT^SGAUTH     AT
```

Figure 4 – Result after execution get_modem_info

3.3 Read data from serial port

In this example is described reading data from serial port and reply back to sender.

geneko:/home/examples#**./serial_read_example**

```
geneko:/home/examples# ./serial_read_example

Thank you for using GWR-HS scripting.

ATTENTION: Please make shure that CLI is disabled on serial port.

Please connect cable to serial port, and set parameters:
   PORT:   /dev/ttyS1
   SPEED:  115200
   PARAMS: cs8 -parenb -parodd -cstopb

Please type anything and press <ENTER>

test
```

Figure 5 – Result after execution serial_read_example

geneko:/home/examples#**vi serial_read_example**

```
#!/bin/bash
#
# (c) 2014 GENEKO
#
# Example: Read data from serial port and reply back to sender
#

#
# Please configure serial parameters if needed
#
SERIAL_PORT='/dev/ttyS1'        # Default: /dev/serial          Available: ttyS1 - external serial, ttyS0 - internal debug port
SERIAL_SPEED='115200'           # Default: '115200'             Available: 300 - 115200
SERIAL_BITS='cs8'               # Default: 'cs8'                Available: 'cs5', 'cs6', 'cs7', 'cs8'
SERIAL_PARITY='-parenb -parodd' # Default: '-parenb -parodd'    Available: 'parenb', 'parodd', '-parenb -parodd'
SERIAL_STOP_BITS='-cstopb'      # Default: '-cstopb'            Available: '-cstopb' (1 stop bit), 'cstopb' (2 stop bits)

#
# Required global variables
#
SERIAL_DEBUG="NO"
CRLF="\\010\\013"

#
# Write data to serial port
#
serial_write() {
   if [ "${SERIAL_DEBUG}" = "YES" ]; then
      echo -e "Write \"${1}\" to serial port ${SERIAL_PORT}"
   fi
   echo -e "$1" >"${SERIAL_PORT}"
   }

#
# Read from serial port
#
read_serial() {
   read -r input <"${SERIAL_PORT}"
   serial_output=`echo ${input} | tr -d '\r\n'`
   echo "${serial_output}"
   }

#
# Print data on screen and $SERIAL_PORT
#
write_both() {
   serial_write "${1}"
   echo -e "${1}"
```

Figure 6 – Display serial_read_example

3.4 Read received SMS messages

In this example is described the possibility of a modem GWG Gateway to receive SMS messages.

geneko:/home/examples#**./read_sms_perform_action**

```
Modem mode:     DATA

Switching to text mode.
AT+CMGF=1
OK
Modem mode:     TEXT

-----------------------------------------------------------------------


-----------------------------------------------------------------------
0
REC READ
+381640140716

16/02/17
12:15:09+04
Test
-----------------------------------------------------------------------
1
REC READ
+381640140716

16/02/17
12:44:16+04
Gwg
-----------------------------------------------------------------------

Command completed successfuly.
```

Figure 7 – Result after execution read_sms_perform_action

geneko:/home/examples#**vi read_sms_perform**

```
#!/bin/bash
#
# (c) 2014 GENEKO
#
# GENEKOSH Example: Check for received SMS messages
#

ATCMD="/usr/local/bin/send_at_command"
GREP="/bin/grep"
AWK="/bin/awk"

getSmsServiceParam() {
  lne=`/bin/cat /params/configuration/params/j_sms_settings.params | JSON.sh -l | egrep '\["'${1}'"\]*' | awk '{ print $2 }'`
  if [ "$lne" == "" ]; then
     echo "<empty/not entered>"
  else
     echo "$lne"
  fi
}

sim1_enable_service=$(getSmsServiceParam "sim1_enable_control")
sim2_enable_service=$(getSmsServiceParam "sim2_enable_control")

echo
if [ "$sim1_enable_service" == "true" ]; then
   echo "Please disable Remote Controll for SIM1 through WEB interface or CLI."
   echo
   exit
fi
if [ "$sim2_enable_service" == "true" ]; then
   echo "Please disable Remote Controll for SIM2 through WEB interface or CLI."
   echo
   exit
fi

modem_mode=`${ATCMD} 'AT+CMGF?' | ${GREP} -v "+CGMF" | ${AWK} -F\: '{ print $2 }' | ${AWK} '{ print $1 }'`
if [ "${modem_mode}" == "" ]; then
   exit 1
fi
if [ $modem_mode -eq 0 ]; then
   echo -e "Modem mode:\tDATA"
   echo
   echo "Do you realy wan't to disconnect data call?"
   read -r shure
   echo
   if [ "$shure" == "y" ] || [ "$shure" == "Y" ] || [ "$shure" == "d" ] || [ "$shure" == "D" ]; then
      ${ATCMD} "AT+CMGF=1"
```

```
   else
      echo "Thank you for using GENEKOSH examples."
      exit
   fi
fi

modem_mode=`send_at_command 'AT+CMGF?' | ${GREP} -v "+CGMF" | ${AWK} -F\: '{ print $2 }' | ${AWK} '{ print $1 }'`
if [ $modem_mode -eq 0 ]; then
   echo "Unable to switch modem to TEXT mode."
   echo
   exit 1
else
   echo -e "Modem mode:\tTEXT"
   echo
   echo "----------------------------------------------------------------------"
   echo
fi

msgs_found=0
while read line
do
   if [ "$line" == "OK" ]; then
      if [ $msgs_found -eq 0 ]; then
         echo
         echo "Message store is empty."
         # echo
         # echo "If SMS service is enabled on router, please disable Remote Control?"
         echo
      else
         echo ${sms_message}
      fi
      echo "----------------------------------------------------------------------"
      echo
      echo "Command completed successfuly."
      break;
   fi
   new_sms=`echo "${line}" | awk '{ if (substr($0,0,5)=="+CMGL") { print substr($0, 7, length($0)-7) } }'`
   if [ "${new_sms}" == "" ]; then
      if [ "$sms_message" == "" ]; then
         sms_message="$line"
      else
         sms_message="${sms_message}\r\n$line"
      fi
   else
      # Execute command stored in ARRAY sms_data[]
      echo ${sms_message}
      echo "----------------------------------------------------------------------"
      echo "----------------------------------------------------------------------"
      msgs_found=1
      IFS=',' read -a sms_data <<< "$new_sms"
      for i in {0..5}
      do
         sms_data=`echo "${sms_data[${i}]}" | sed 's/^[\"^t]*//;s/[\"^]*$//'`
         echo ${sms_data}
      done
      sms_message=""
   fi
done <<< "`${ATCMD} 'AT+CMGL=\"ALL\"' | ${GREP} -v 'AT+CMGL=\"ALL\"'`"

exit
```

Figure 8 – Display read_sms_perform #1

3.5 Remove SIM card PIN

In this example is described information about SIM card PIN and PUK code and possibility to unlock SIM card.

geneko:/home/examples#**./remove_sim_card_pin**

```
geneko:/home/examples# ./remove_sim_card_pin

Thank you for using GENEKOSH scripting

Checking existence of PIN code...
SIM STATUS: Not waiting for PIN (no PIN or PIN already entered).

Your SIM card is currently unlocked!
If you really need to disable SIM lock on this card, please use
   ./remove_sim_card_pin --force
```

Figure 9 – Result after execution remove_sim_card_pin

geneko:/home/examples#**vi remove_sim_card_pin**

```
#!/bin/bash
#
# (c) 2014 GENEKO
#
# GENEKOSH Example: Remove lock from SIM card
#

#
# Start script
#
echo
echo "Thank you for using GENEKOSH scripting"
echo

echo "Checking existence of PIN code..."
helper=`/usr/local/bin/send_at_command "AT+CPIN?" | awk '{ print $2 }' | tr -d '\n\r'`
if [ "${helper}" == "READY" ]; then
   echo "SIM STATUS: Not waiting for PIN (no PIN or PIN already entered)."
else
   if [ "${helper}" == "SIM PIN" ]; then
      echo "SIM STATUS: Waiting for SIM PIN code."
   else
      if [ "${helper}" == "SIM PUK" ]; then
         echo "SIM STATUS: Waiting for SIM PUK code."
      else
         echo "SIM STATUS: Unknown SIM status \"${helper}\""
      fi
   fi
fi
echo
if [ "${helper}" == "SIM PIN" ] || [ "$1" == "-force" ] || [ "$1" == "--force" ]; then
   echo "Your SIM card is locked. Please enter pin to unlock:"
   read -r sim_pin
   if [ "$sim_pin" == "" ]; then
      echo "SIM PIN is not entered. Aborting!"
      exit
   else
      echo
      echo "Entered SIM PIN: '${sim_pin}'."
      echo
      echo "Do You really want to unlock card with this PIN code?"
      echo "If wrong PIN is entered three times, your SIM card will be blocked."
      echo
      read -r shure
      if [ "$shure" == "y" ] || [ "$shure" == "Y" ] || [ "$shure" == "d" ] || [ "$shure" == "D" ]; then
         echo -e "unlocking SIM card ...\tsend_at_command AT+CPIN=${sim_pin}"
         helper=`/usr/local/bin/send_at_command AT+CPIN=\"${sim_pin}\" | tail -1 | grep OK`
- remove_sim_card_pin 1/71 1%
         helper=`/usr/local/bin/send_at_command AT+CLCK=\"SC\",0,\"${sim_pin}\" | tail -1 | grep OK`
         if [ "$helper" == "OK" ]; then
            echo "   SIM card successfully unlocked!"
         else
            echo "   SIM card was already unlocked."
         fi
         echo -e "removing SIM lock ...\tsend_at_command AT+CLCK=\"SC\",0,\"${sim_pin}\""
         helper=`/usr/local/bin/send_at_command AT+CLCK=\"SC\",0,\"${sim_pin}\" | tail -1 | grep OK`
         if [ "$helper" == "OK" ]; then
            echo "   SIM unlock complete!"
         else
            echo "   Unable to unlock SIM card."
         fi
      else
         echo "Program aborted by user request."
         exit
      fi
   fi
else
   echo "Your SIM card is currently unlocked!"
   echo "If you really need to disable SIM lock on this card, please use"
   echo "   $0 --force"
fi
```

Figure 10 – Display remove_sim_card_pin

3.6 Report ppp_0 status

In this example is described basic information about ppp_0 status, mobile provider, APN, username, password and etc..

geneko:/home/examples#**./report_ppp_status**

```
geneko:/home/examples# ./report_ppp_status

(Status: Established)

Mobile provider:  mts
Username:         <empty/not entered>
Password:         <empty/not entered>
APN:              genekogwr

IP address:       172.27.234.23
```

Figure 11 – Result after execution report_ppp_status

geneko:/home/examples#**vi report_ppp_status**

```
#!/bin/bash
#
# (c) 2014 GENEKO

#
# GENEKOSH Example: Check for /dev/ppp_0 interface and report status
#

WAN_SETTINGS="/params/configuration/params/j_mobile_settings.params"

getNetworkParam() {
  lne=`/bin/cat ${WAN_SETTINGS} | JSON.sh -l | egrep '\["'${1}'",'${2}','"'${3}'"\]*' | awk '{ print $2 }' | sed 's/^[\"^t]*//;s/[\"^]*$//'`
  if [ "$lne" == "" ]; then
     echo "<empty/not entered>"
  else
     echo "$lne"
  fi
}

TMP_WAN_IFACE=`ifconfig ppp_0 2> /dev/null | wc -l`
if [ $TMP_WAN_IFACE -eq 0 ]; then
   echo "(Status: Not Established)"
else
   TMP_WAN_ADDRESS=`ifconfig ppp_0 | grep "inet addr" | cut -d ":" -f 2 | awk '{printf $1}'`

   echo
   echo "(Status: Established)"
   echo
   helper=$(getNetworkParam "auth" "0" "provider_name")
   echo -e "Mobile provider:  ${helper}"
   helper=$(getNetworkParam "auth" "0" "username")
   echo -e "Username:\t  ${helper}"
   helper=$(getNetworkParam "auth" "0" "password")
   echo -e "Password:\t  ${helper}"
   helper=$(getNetworkParam "auth" "0" "apn")
   echo -e "APN:\t\t  ${helper}"

   echo
   if [ "$TMP_WAN_ADDRESS" = "" ]; then
      echo -e "IP address:\t  Waiting for IP address..."
   else
      echo -e "IP address:\t  "${TMP_WAN_ADDRESS}
   fi
fi
```

Figure 12 – Display report_ppp_status

3.7 Router Network configuration

In this example is described making router network configuration. There is the possibility of exporting the new configuration via web interface.

geneko:/home/examples#**./router_network_configuration_example**

```
geneko:/home/examples# ./router_network_configuration_example
[firmware_versions]
firmware_versions.params="1.0"
firmware_versions.serial="201403031341"
firmware_versions.version="1.0.0"
firmware_versions.model="geneko"
[network_settings]
network_settings.ipaddress="10.0.10.97"
network_settings.netmask="255.255.255.0"
[dhcpserver_settings]
dhcpserver_settings.dhcpsettings.netmask="255.255.255.0"
dhcpserver_settings.dhcpsettings.enable=true
dhcpserver_settings.dhcpsettings.network="10.0.10.0"
dhcpserver_settings.addressranges_ui[1].startipaddress="10.0.10.240"
dhcpserver_settings.addressranges_ui[1].stopipaddress="10.0.10.250"
dhcpserver_settings.addressranges_ui[1].enable=true
dhcpserver_settings.addressranges[1].startipaddress="10.0.10.240"
dhcpserver_settings.addressranges[1].stopipaddress="10.0.10.250"
dhcpserver_settings.addressranges[1].enable=true

Applying new configuration...
Configuration deployed: OK
Configuration upload is successfull. Please reboot the device to take effect.
geneko:/home/examples# reboot
```

Figure 13 – Result after execution router_network_configuration_example

geneko:/home/examples#**vi router_network_configuration_example**

```
#!/bin/bash
cd ~
CONF_FILE="router.conf.$$"

ROUTER_IP="10.0.10.97"
SUBNET_MASK="255.255.255.0"
DHCP_START_IP="240"
DHCP_END_IP="250"

#
# Get network from router IP address
#
IFS=. read _ip1 _ip2 _ip3 _ip4 <<< "${ROUTER_IP}"
NETWORK="${_ip1}.${_ip2}.${_ip3}"

#
# Prepare configuration file
#
echo "[firmware_versions]"> ${CONF_FILE}
echo "firmware_versions.params=\"1.0\"">> ${CONF_FILE}
echo "firmware_versions.serial=\"201403031341\"">> ${CONF_FILE}
echo "firmware_versions.version=\"1.0.0\"">> ${CONF_FILE}
echo "firmware_versions.model=\"geneko\"">> ${CONF_FILE}
echo "[network_settings]">> ${CONF_FILE}
echo "network_settings.ipaddress=\"${ROUTER_IP}\"">> ${CONF_FILE}
echo "network_settings.netmask=\"${SUBNET_MASK}\"">> ${CONF_FILE}
echo "[dhcpserver_settings]">> ${CONF_FILE}
echo "dhcpserver_settings.dhcpsettings.netmask=\"${SUBNET_MASK}\"">> ${CONF_FILE}
echo "dhcpserver_settings.dhcpsettings.enable=true">> ${CONF_FILE}
echo "dhcpserver_settings.dhcpsettings.network=\"${NETWORK}.0\"">> ${CONF_FILE}
echo "dhcpserver_settings.addressranges_ui[1].startipaddress=\"${NETWORK}.${DHCP_START_IP}\"">> ${CONF_FILE}
echo "dhcpserver_settings.addressranges_ui[1].stopipaddress=\"${NETWORK}.${DHCP_END_IP}\""> ${CONF_FILE}
echo "dhcpserver_settings.addressranges_ui[1].enable=true">> ${CONF_FILE}
echo "dhcpserver_settings.addressranges[1].startipaddress=\"${NETWORK}.${DHCP_START_IP}\""> ${CONF_FILE}
echo "dhcpserver_settings.addressranges[1].stopipaddress=\"${NETWORK}.${DHCP_END_IP}\""> ${CONF_FILE}
echo "dhcpserver_settings.addressranges[1].enable=true">> ${CONF_FILE}

#
# Show configuration
#
/bin/cat "${CONF_FILE}"

echo

#
# Import configuration
#
#
configuration_import -l "${CONF_FILE}"

#
# Remove vonfiguration file
#
/bin/rm "${CONF_FILE}"
```

Figure 14 – Display router_network_configuration_example #1

3.8 Connect and access of TCP server

In this example is described reading data from TCP socket and return reply.
geneko:/home/examples#**./tcp_server_example**

```
geneko:/home/examples# ./tcp_server_example
EXAMPLE SERVER: To test server, please connect to 192.168.1.1 on port 7777
```

Figure 15 – Result after execution tcp_server_example

Use PuTTY for connection on port 7777, connection type is Telnet.

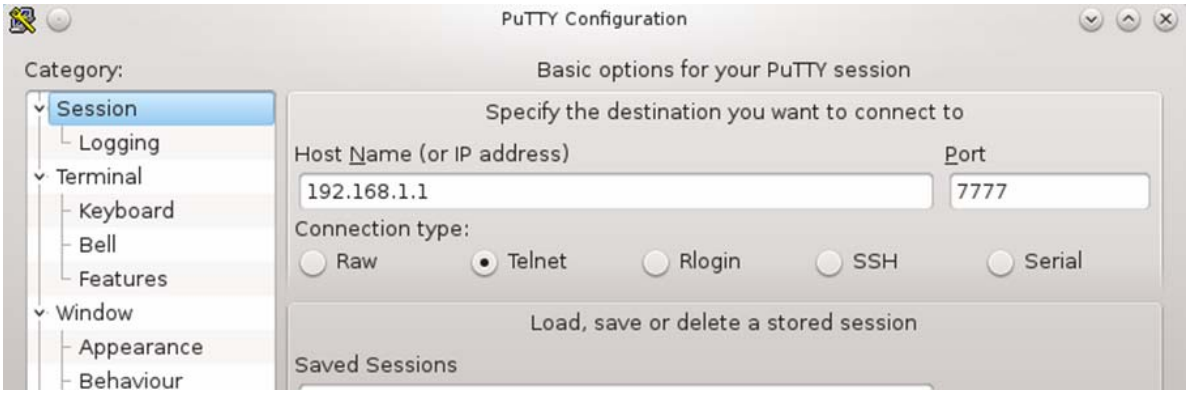

Figure 16 – PuTTY configuration

```
Welcome to GSH server

br0       Link encap:Ethernet  HWaddr 00:1E:5C:00:79:9A
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::21e:5cff:fe00:799a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1325 errors:0 dropped:0 overruns:0 frame:0
          TX packets:633 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:84742 (82.7 KiB)  TX bytes:79283 (77.4 KiB)

eth0      Link encap:Ethernet  HWaddr 00:1E:5C:00:79:9A
          inet6 addr: fe80::21e:5cff:fe00:799a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1325 errors:0 dropped:0 overruns:0 frame:0
          TX packets:641 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:88158 (86.0 KiB)  TX bytes:79931 (78.0 KiB)
          Interrupt:41 Base address:0xc000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:220 errors:0 dropped:0 overruns:0 frame:0
          TX packets:220 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:23073 (22.5 KiB)  TX bytes:23073 (22.5 KiB)

ppp_0     Link encap:Ethernet  HWaddr A2:4B:F4:2D:AF:06
          inet addr:172.27.234.23  Bcast:172.27.234.31  Mask:255.255.255.240
          inet6 addr: fe80::a04b:f4ff:fe2d:af06/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:974 (974.0 B)  TX bytes:1994 (1.9 KiB)


Please type something terminated with <ENTER>
                                        test

You wrote: test

Thank you for testing GSH functionalities.
```

Figure 17 – PuTTY display

3.9 SMS Send

In this example is described sending sms when the strength of a signal is lower than recommended.

geneko:/home/examples#**./sms_send**
Enter the phone number in the form +31625670634:
+381640140555 <ENTER>
The signal strength is in the recommended range. Signal strength is 19.

If the signal is lower than recommended, message will be sent to the registered number.

```
#!/bin/bash

#(c) 2016 GENEKO

#GENEKOSH Example: SMS_send when the signal is lower then recommended.

#SMS will be sent to this number
echo "Enter the phone number in the form +31625670634:"
read number
#If signal is lower than this value, message will be sent
LOWER_LIMIT_SIGNAL=7

#Call the service with the AT command argument, at first
CSQ=$(send_at_command "AT+CSQ" )
#Received signal strength indication
RSSI=$(echo $CSQ | cut -d:  -f 2 | cut -d, -f 1)

if [ $RSSI -lt $LOWER_LIMIT_SIGNAL ] ; then
        sms_send -n "$number" -m "The signal strength is lower than recommended. Signal strength is $RSSI"
else
        echo "The signal strength is in the recommended range.Signal strength is $RSSI"
fi

~
```

Figure 18 – Display sms_send

3.10 E-mail send

In this example is described sending e-mail.

geneko:/home/examples**#$ ./smtp.netcat.test mx.example.com 25 from@example.com to@example.com**

```
geneko:~# ./mail.sh webmail.geneko.rs 25  mlukac@geneko.rs tsavic@geneko.rs
 Enter the subject of e-mail.
GWG
 Enter the content of e-mail.
GWG sends e-mail.
220 ******************************
250-mail.geneko.co.rs
250-PIPELINING
250-SIZE 52428800
250-VRFY
250-ETRN
250-XXXXXXXA
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
250 2.1.0 Ok
250 2.1.5 Ok
354 End data with <CR><LF>.<CR><LF>
250 2.0.0 Ok: queued as E9EC51BE459F
221 2.0.0 Bye
geneko:~#
```

Figure 19 – Result after execution script send_e-mail

geneko:/home/examples**#vi send_e-mail**

```
#!/bin/bash
# script to send mail with netcat.
# expects the following arguments:
# 1. recepient mail server
# 2. port (typically 25)
# 3. mail from (e.g. from@example.com)
# 4. mail to (e.g. to@example.com)
#call script: $ ./smtp.netcat.test mx.example.com 25 from@example.com to@example.com

#input parameters, subject and content of e-mail
echo " Enter the subject of e-mail."
read subject
echo " Enter the content of e-mail."
read content

# for e-mail_input function
from=$3
to=$4

# error handling
function err_exit { echo -e 1>&2; exit 1; }

# check if proper arguments are supplied
if [ $# -ne 4 ]; then
  echo -e "\n Usage error!"
  echo " This script requires four arguments:"
  echo " 1. recepient mail server"
  echo " 2. port (typically 25)"
  echo " 3. mail from (e.g. from@example.com)"
  echo " 4. mail to (e.g. to@example.com)"
  exit 1
fi

# create message
function mail_input {
  echo "ehlo $(hostname -f)"
  sleep 8
  echo "MAIL FROM: <$from>"
  sleep 8
  echo "RCPT TO: <$to>"
  sleep 8
  echo "DATA"
  sleep 8
  echo "From: <$from>"
  sleep 8
  echo "To: <$to>"
  sleep 8
  echo "Subject: $subject"
  echo "$content"
  echo "."
  echo "quit"
}

# send
mail_input | nc $1 $2 || err_exit
```

Figure 20 – Display send_e_mail

3.11 General Purpose Input/Output (GPIO)

This chapter describes the General Purpose Input/Output (GPIO) functionality on the GWG Gateway.
On router's board are 5 GPIO generic pins which represent:

1. +5VDC with 500mA resettable PTC fuse
2. IO1
3. IO2
4. IO3
5. GND

IO1, IO2, IO3 are 3 user selectable input or output.
Input value is readable (high=1, low=0).
Output value is writable (high=1, low=0).

3.11.1 gpio_functionality

In this example is described GPIO functionality supported by **service gpio**.

admin@geneko>service gpio

service gpio -i	read value of an input GPIO
service gpio -o	write value to an output GPIO (gpio_id must be specified)
service gpio -g <gpio_id>	GPIO to read/write. If this option is not specified it will return combined value of all GPIO inputs. If the ID is wrong, the program will return EXIT_WRONG_ID value
service gpio -v <value>	value to be written to an output GPIO. Valid values are 0 and 1. If the value is wrong, the program will return EXIT_WRONG_VALUE value
-h	this help

geneko:/home/examples#**./gpio_functionality.sh**

```
geneko:/home/examples# ./gpio_functionality.sh
**Read value of an input GPIO.Enter the number of pin (e.g 1 or 2 or 3 ):
2
Result is 1. Value 1 representing a high voltage.
**Read combined value of all GPIO inputs.
All pins return value 1, and representing high voltage.
**Write value to an output GPIO
Set value of pin to an output GPIO
Enter the number of pin (e.g 1 or 2 or 3 ):
2
Enter value to be written to an output GPIO. Valid values are 0 and 1.
1
**Read combined value of all GPIO inputs.
Pin 2 returns value 0 and representing low voltage.
```

Figure 21 - Result after execution gpio_functionality.sh

geneko:/home/examples#**vi gpio_functionality.sh**

```
#!/bin/bash

##########################################################################################

# GPIO example describes the General Purpose Input/Output (GPIO) functionality

##########################################################################################

#Read value of an input GPIO
input () {
echo "**Read value of an input GPIO.Enter the number of pin (e.g 1 or 2 or 3 ): "
read pin

service_1=`service gpio -g $pin -i`
if [ "$service_1" = "0" ]; then
  echo "Result is "$service_1"." "Value 0 representing a low voltage. "
else
  echo "Result is "$service_1"." "Value 1 representing a high voltage."
fi
}

#Read combined value of all GPIO inputs
combined_input() {
echo "**Read combined value of all GPIO inputs."
service_2=`service gpio`
if [ "$service_2" = "7" ]; then
  echo "All pins return value 1, and representing high voltage."
elif [ "$service_2" = "6" ]; then
  echo  "Pin 1 returns value 0 and representing low voltage. "
elif [ "$service_2" = "5" ]; then
  echo  "Pin 2 returns value 0 and representing low voltage. "
elif [ "$service_2" = "4" ]; then
  echo "Pin 1 and pin 2 return value 0, and representing low voltage."
elif [ "$service_2" = "3" ]; then
  echo "Pin 3 returns value 0 and representing low voltage."
elif [ "$service_2" = "2" ]; then
  echo  "Pin 1 and pin 3 return value 0 and representing low voltage."
elif [ "$service_2" = "1" ]; then
  echo  "Pin 2 and pin 3 return value 0 and representing low voltage."
fi
}

#Write value to an output GPIO
output() {
echo "**Write value to an output GPIO"
echo "Set value of pin to an output GPIO"
echo "Enter the number of pin (e.g 1 or 2 or 3 ):"
read g
echo "Enter value to be written to an output GPIO. Valid values are 0 and 1."
read o
service_3=`service gpio -g $g -o $g -v $o`
combined_input
}

##############################################################################################

input
combined_input
output
```

Figure 22 - gpio_functionality

3.11.2 gpio_send_sms

In this example is described, when gpio pin change its state, GWG Gateway read combined value of all GPIO inputs. If the state of GPIO pins is changed, GWG Gateway sends sms.

geneko:/home/examples#**./gpio_send_sms.sh**

```
Enter the phone number in the form +31625670634:
+381640140716
** Check GPIO pin state
Pin 2 returns value 0 and representing low voltage.
**Write value to an output GPIO
Set value of pin to an output GPIO
Enter the number of pin (e.g 1 or 2 or 3 ):
1
Enter value to be written to an output GPIO. Valid values are 0 and 1.
1
Read combined value of all GPIO inputs. If the state of GPIO pins is changed, send sms.
```

Figure 23 - Result after execution gpio_send_sms.sh

geneko:/home/examples# **vi gpio_send_sms.sh**

```
#!/bin/bash

##########################################################################################

# GPIO example: When gpio pin changes its state to Low or High, send sms

##########################################################################################

#Write value to an output GPIO
output() {
echo "**Write value to an output GPIO"
echo "Set value of pin to an output GPIO"
echo "Enter the number of pin (e.g 1 or 2 or 3 ):"
read g
echo "Enter value to be written to an output GPIO. Valid values are 0 and 1."
read o
service_3=`service gpio -g $g -o $g -v $o`
}
#Remember the state of gpio pin before the change
check_gpio_state() {
echo "** Check GPIO pin state"
service_1=`service gpio`
if [ "$service_1" = "7" ]; then
  echo "All pins return value 1, and representing high voltage."
elif [ "$service_1" = "6" ]; then
  echo  "Pin 1 returns value 0 and representing low voltage. "
elif [ "$service_1" = "5" ]; then
  echo  "Pin 2 returns value 0 and representing low voltage. "
elif [ "$service_1" = "4" ]; then
  echo "Pin 1 and pin 2 return value 0, and representing low voltage."
elif [ "$service_1" = "3" ]; then
  echo "Pin 3 returns value 0 and representing low voltage."
elif [ "$service_1" = "2" ]; then
  echo  "Pin 1 and pin 3 return value 0 and representing low voltage."
elif [ "$service_1" = "1" ]; then
  echo  "Pin 2 and pin 3 return value 0 and representing low voltage."
fi

}
```

```
send_sms() {
echo "Read combined value of all GPIO inputs. If the state of GPIO pins is changed, send sms."
service_2=`service gpio`
if [ "$service_2" = "7" ]; then
  sms_send -n "$number" -m "GPIO pin changed its state. All pins return value 1, and representing high voltage."
elif [ "$service_2" = "6" ]; then
  sms_send -n "$number" -m "GPIO pin changed its state. Pin 1 returns value 0 and representing low voltage. "
elif [ "$service_2" = "5" ]; then
  sms_send -n "$number" -m "GPIO pin changed its state. Pin 2 returns value 0 and representing low voltage. "
elif [ "$service_2" = "4" ]; then
  sms_send -n "$number" -m "GPIO pin changed its state. Pin 1 and pin 2 return value 0, and representing low voltage."
elif [ "$service_2" = "3" ]; then
  sms_send -n "$number" -m "GPIO pin changed its state. Pin 3 returns value 0 and representing low voltage."
elif [ "$service_2" = "2" ]; then
  sms_send -n "$number" -m "GPIO pin changed its state. Pin 1 and pin 3 return value 0 and representing low voltage."
elif [ "$service_2" = "1" ]; then
  sms_send -n "$number" -m "GPIO pin changed its state. Pin 2 and pin 3 return value 0 and representing low voltage."
fi
}

##########################################################################################

#SMS will be sent to this number
echo "Enter the phone number in the form +31625670634:"
read number
check_gpio_state
output
if [ "$service_1" != "$service_3" ]; then
send_sms
fi
```

Figure 24 - gpio_send_sms

GENEKO

Bul. Despota Stefana 59a
11000 Belgrade ▪ Serbia

Phone: +381 11 3340-591, 3340-178
Fax: +381 11 3224-437

e-mail: gwrsupport@geneko.rs
www.geneko.rs