ERA-GLONASS Conformance and Performance Testing
Application Note

Products:

- R&S®CMW500
- R&S®CMW-KA094
- R&S®CMW-KA095
- R&S®SMBV100A
- R&S®SMBV-K360

Emergency Road Assistance (ERA-GLONASS) is a service provided in the Russian Federation with the goal of reducing response times for accidents or other emergencies on the roadways. This application note briefly describes the technology behind ERA-GLONASS and presents conformance tests for ERA-GLONASS using the R&S®CMW500 RF tester and the R&S®SMBV100A vector signal generator. A Test software for ERA-GLONASS makes it quick and easy to perform these tests with the GSM or WCDMA wireless communications standard. It also shows a test solution for GNSS performance tests for ERA-GLONASS using the R&S®SMBV100A vector signal generator and the option R&S®SMBV-K360 together with CMWrun.

Note:

The eCall Conformance and Performance Testing is described in Application Note 1MA241.

Visit our homepage for the most recent version of this application note (www.rohde-schwarz.com/appnote/1MA251).

This document is complemented by software. The software may be updated even if the version of the document remains unchanged.

ERA_GLONASS — 1MA251_3e (01/2018)

Bernhard Schulz

Application Note

Contents

1 Introduction

ERA-GLONASS is a service provided in the Russian Federation with the goal of reducing response times for accidents or other emergencies on the roadways. The Russian government has enacted a law requiring integrated ERA modules on all new automobile models. ERA is harmonized with the European eCall system. It uses the same principals and protocols, but provides further features like a redundant channel (SMS) for MSD transfer and is designed for services like, e.g., fleet management, a toll road system or digital tachographs.

Although ERA-GLONASS can not prevent accidents, when accidents do occur, a call is placed automatically (e.g. when the airbag deploys) to the emergency number 112. Essential information (including the vehicle's current location) is transmitted in a standard data format. The in-vehicle system (IVS) collects the data and transmits it via a GSM or WCDMA voice call to the public safety answering point (PSAP).

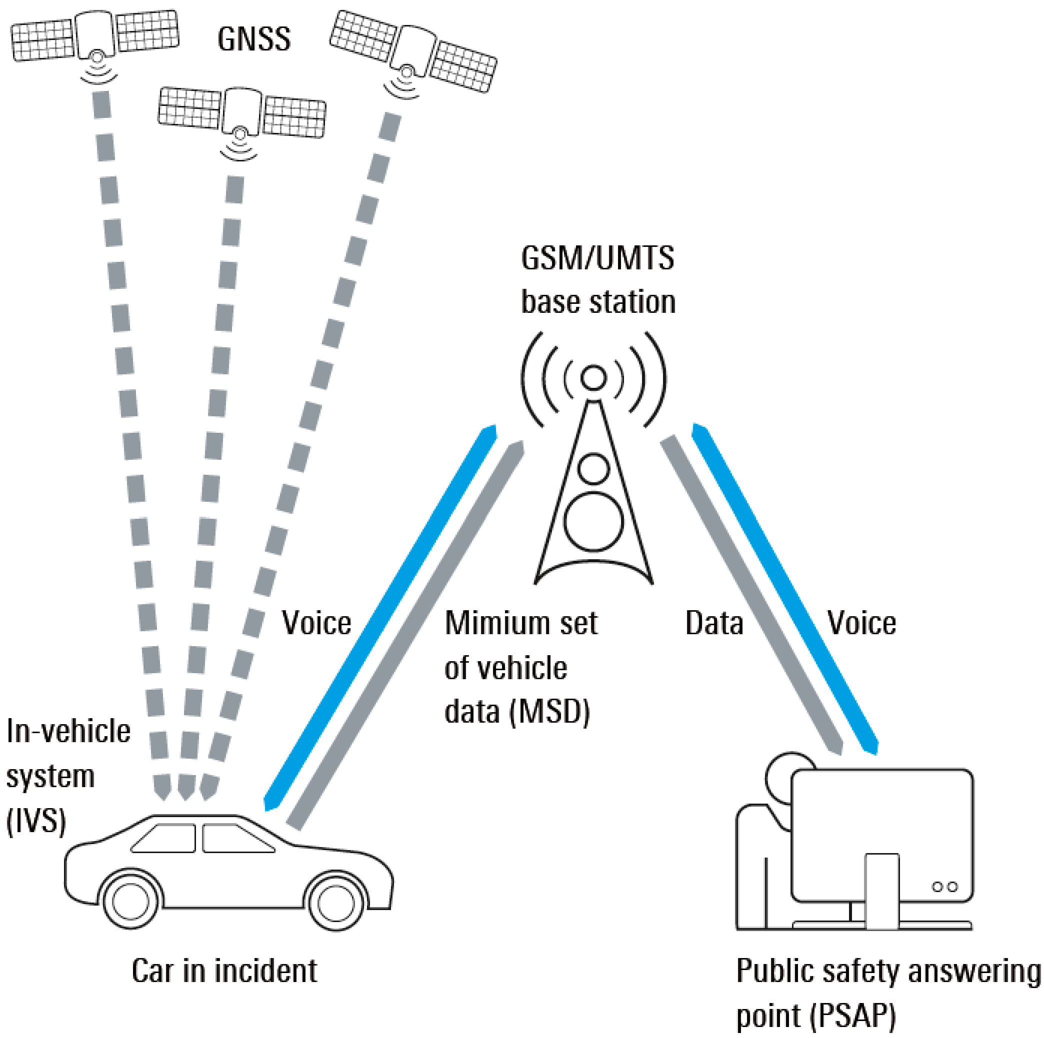


Figure 1-1: ERA-GLONASS system: In an emergency situation, the car makes a cellular (GSM or WCDMA) emergency call to emergency services. The car automatically transmits essential data, including its location. After that, an emergency services operator can speak directly with the vehicle's occupants, e.g. to request additional information.

Performance and Conformance Tests

The application note describes the principles behind ERA-GLONASS and explains both, tests of the GNSS performance and tests of the IVS conformance.

The GNSS performance tests are for evaluation of the GNSS receiver of the IVS with tests for parameters like GNSS receiver sensitivity or the position accuracy.

The IVS conformance tests are for evaluation of the whole IVS, such as the correct transmission of the whole data (MSD) via mobile networks (GSM or WCDMA) including a valid vehicle position.

IVS Conformance Tests

This application note describes the conformance testing of the IVS using the test solution offered by Rohde & Schwarz.

The provided PC software and the test solution make it easier for the user to concentrate on the actual test tasks:

- Simulation of the PSAP and control of the ERA-GLONASS emergency calls via GSM or WCDMA
- Measurement of times and decoding of the minimum set of data (MSD)
- Display of the exchanged protocol messages
- An in-depth understanding of the specifications for wireless communications and GNSS is not necessary
- Deep familiarity with the operation of the test instrument is likewise unnecessary
- All relevant GSM or WCDMA cellular network parameters can be modified
- This allows reproducible measurement results
- A true emergency call using the emergency number 112 is possible

GNSS Performance Tests

This application note describes the GNSS performance testing of the IVS using the test solution offered by Rohde & Schwarz.

The provided PC software (CMWrun) and the test solution make it easier for the user to concentrate on the actual test tasks:

- The test suite controls the test equipment via SCPI and the IVS via vendor-specific commands.
- The test cases are performed fully automatic, without user interaction.
- An in-depth understanding of the specifications for GNSS is not necessary
- Deep familiarity with the operation of the test instrument is likewise unnecessary
- All relevant GNSS parameters can be modified
- This allows reproducible measurement results

The following abbreviations are used in this Application Note for Rohde & Schwarz test equipment:

- The R&S®CMW500 radio communication tester is referred to as the CMW.
- The R&S®SMBV100A vector signal generator is referred to as the SMBV.

2 What is ERA-GLONASS ?

2.1 ERA-GLONASS: System and Concepts

ERA-GLONASS is a service provided in the Russian Federation with the goal of reducing response times for accidents or other emergencies on the roadways. The Russian government has enacted a law requiring integrated ERA modules on all new automobile models. ERA is harmonized with the European eCall system. It uses the same principals and protocols, but provides further features like a redundant channel (SMS) or additional data services.

Table 2-1: Features and differences ERA/eCall

Feature	**eCall**	**ERA-GLONASS**
Radio Access Networks	GSM(2G) / UMTS(3G)(optional)	GSM(2G) / UMTS(3G)
GNSS	GPS	GLONASS (mandatory), GPS
In-Band Modem	yes	yes (same to pan-European eCall)
secondary (redundant) channel	no	SMS
MSD	MSD	MSD (like the MSD in eCall, but optional additional fields)
additional (packet) data channel	no	yes for location-based services like • fleet management • insurance telematics • toll collection • recovery of stolen vehicles • digital tachygraphy

Although ERA-GLONASS can not prevent accidents, when accidents do occur, a call is placed automatically (e.g. when the airbag deploys) to the emergency number 112. Essential information (including the vehicle's current location) is transmitted in a standard data format. In addition to the automatic call, ERA can also be used to place a call manually (e.g. in another type of emergency).

See Chapter 1, "Introduction", on page 3 for an overview.

An in-vehicle system (IVS) is integrated as a key element into every automobile (Figure 2-1). The primary components of the IVS are a GNSS receiver (GLONASS, typically in combination with GPS) to determine the current position and a cellular module (GSM and/or WCDMA) to permit transmission of the minimum set of data (MSD) via a cellular network to a public safety answering point (PSAP). The data is transmitted over a voice channel on the cellular network because voice channels coverage is better, e.g in rural areas. Pure data channels are not available in every base station, depending on the configuration. However, because voice channels are used, the data must be specifically adapted to the speech coder. This is done by means of a specialized modulation in a so called in-band modem.

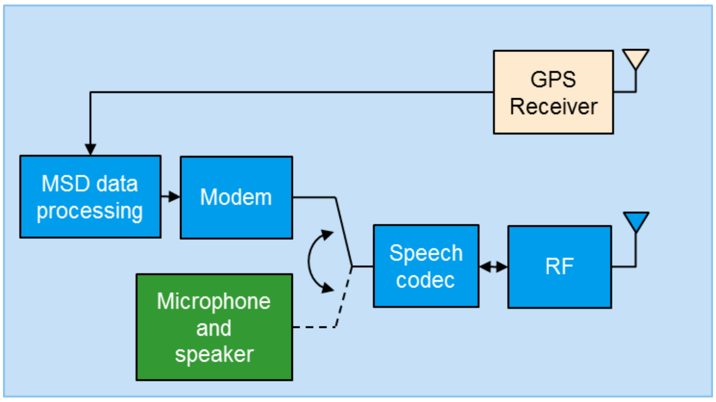


Figure 2-1: Key components of ERA-GLONASS: The IVS prepares essential data and transmits it via the voice channel over the air interface.

The public safety answering point (PSAP, e.g. police, fire, EMT) answers the call (and receives the data) and initiates the appropriate response (e.g. dispatching an ambulance). The voice connection can then be used to request additional information.

The IVS transmits the data to the PSAP as standardized MSD.

Table 2-2: MSD data fields (version 2). The most critical data is the vehicle's location (blocks 7...11)

Block	Name	Necessity	Description
1	Format version	Mandatory	MSD format version; set to 2 for the current format EN 15722:2015
2	Message identifier	Mandatory	Session specific counter; starts at 1 and is incremented with every retransmission
3	Control	Mandatory	Conveys the following information: Automatic or manual activation test call (TRUE/FALSE) Test call (TRUE/FALSE) Position can be trusted (TRUE/FALSE) Vehicle class
4	Vehicle ID	Mandatory	VIN (vehicle identification number) according to ISO3779
5	Propulsion type (energy storage)	Mandatory	Gasoline, diesel, hydrogen, electric, etc. (TRUE/FALSE)
6	Timestamp	Mandatory	Timestamp of incident event
7	Vehicle location	Mandatory	Last known vehicle position. Latitude and longitude (ISO 6709) in milliarcseconds
8	Vehicle direction	Mandatory	Deviation from the direction to the magnetic north pole in 2 degrees steps
9	Recent vehicle location n-1	Optional	Change in latitude and longitude compared to the last MSD transmission
10	Recent vehicle location n-2	Optional	Change in latitude and longitude compared to the last but one MSD transmission

11	No. of passengers	Optional	Number of occupants in the vehicle according to available information.
12	Optional additional data	Optional	Optional information for the emergency rescue service (103 bytes, ASN.1 encoded); may also point to an address, where this information is located

In-Band modem: primary MSD transmission

The MSD consists of 140 bytes and 28 CRC bits, making it a total of 1148 bits in length. After the coding for the forward error correction, the MSD is 1380 bits in length. An MSD consists of a prefixed synchronization frame and three data parts. Between the individual parts, the signal is muted (Figure 2-2).

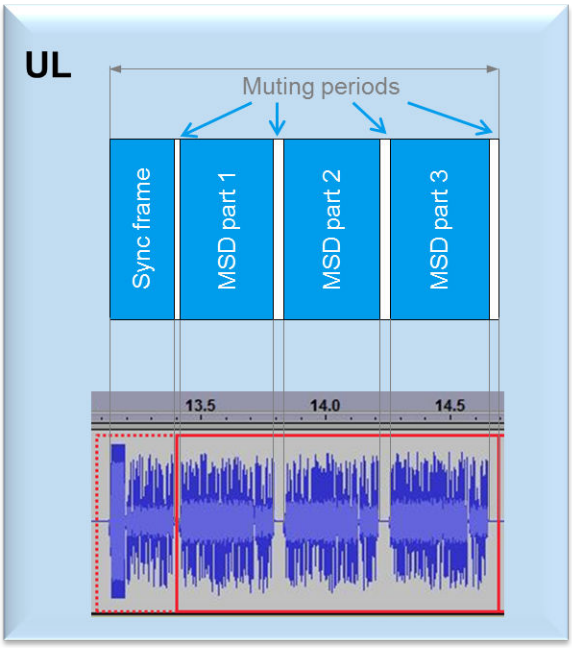


Figure 2-2: The MSD consists of a sync frame and three individual MSD parts (uplink data parts). In between the parts, the signal is muted. An excerpt from an example audio file (wav format) is shown at the bottom of the figure.

The MSD data is bipolar pulse position modulated (BPPM). A basic pulse ($p_{UL}(n)$) is offset either in the positive or negative direction (Figure 2-3). In total, three bits are packed into a symbol, and eight different symbols are available. There are two modes, fast and robust. In robust mode, the symbols are transmitted at double the spacing compared to fast mode. This makes it easier for the receiver to recognize the symbols in situations where the channel conditions are poor.

Table 2-3 lists the parameters for the individual symbols. Figure 2-4 provides an example of the modulation over time.

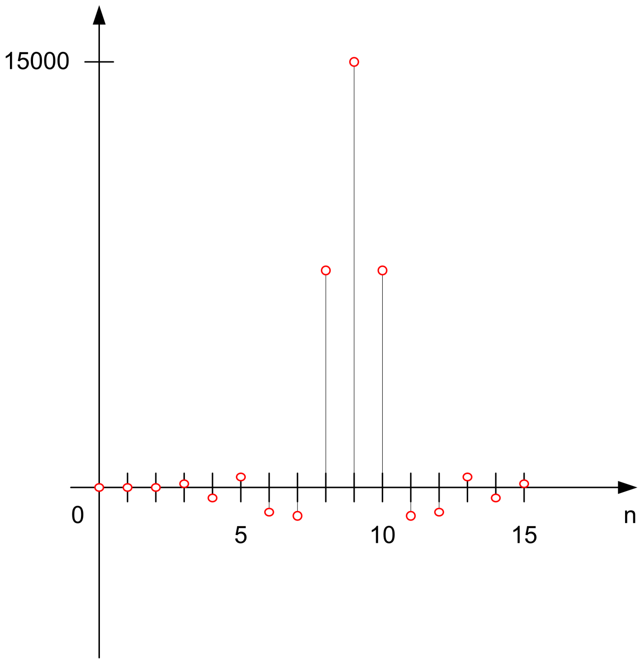


Figure 2-3: The basic BPPM pulse (pUL (n)) (shown here in fast mode as an example).

Table 2-3: Symbol modulation mapping (according to [2]) $w_{UL}(n) = q\ (p(n) \rightarrow k)$

Symbol		Uplink waveform (fast mode: n = 0,1,...,15)		Uplink waveform (robust mode: n = 0,1,...,31))	
d	b	sign q	cyclic shift k	sign q	cyclic shift k
0	000	1	0	1	0
1	001	1	4	1	8
2	010	1	8	1	16
3	011	1	12	1	24
4	100	–1	12	–1	24
5	101	–1	8	–1	16
6	110	–1	4	–1	8
7	111	–1	0	–1	0

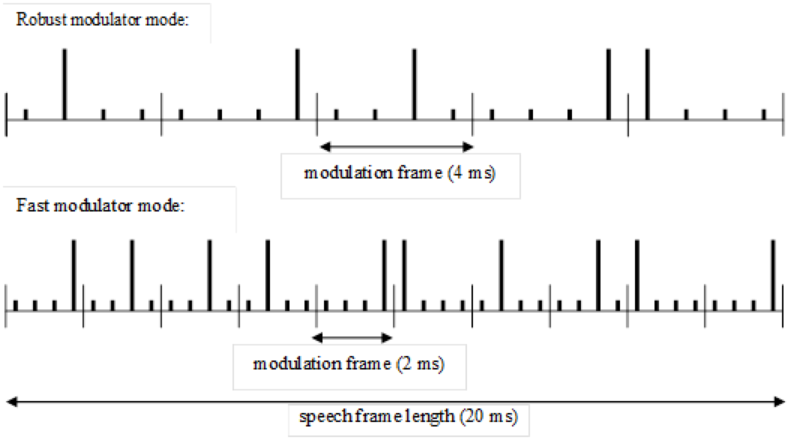


Figure 2-4: Overview of the pulses in one frame. Robust mode is shown at the top and fast mode at the bottom [2].

secondary channel: SMS

As a backup method, in ERA-GLONASS SMS can be used to transmit the MSD. This is the secondary possibility in order to increase the probability of the MSD reception. The SMS transmission can be requested by the PSAP, if the voice connection has been established successfully but the correct MSD reception fails. It also can be used by the IVS if the voice channel cannot be established successfully.

additional data channel

As a basic service, data channels can be used for additional location-based services (see Table 2-1). Here the packet data channel possibilities already provided by GSM like (E)GPRS or WCDMA like HSPA(+) are used.

2.2 Sequences

The following figure uses a typical scenario to illustrate communications over the time.

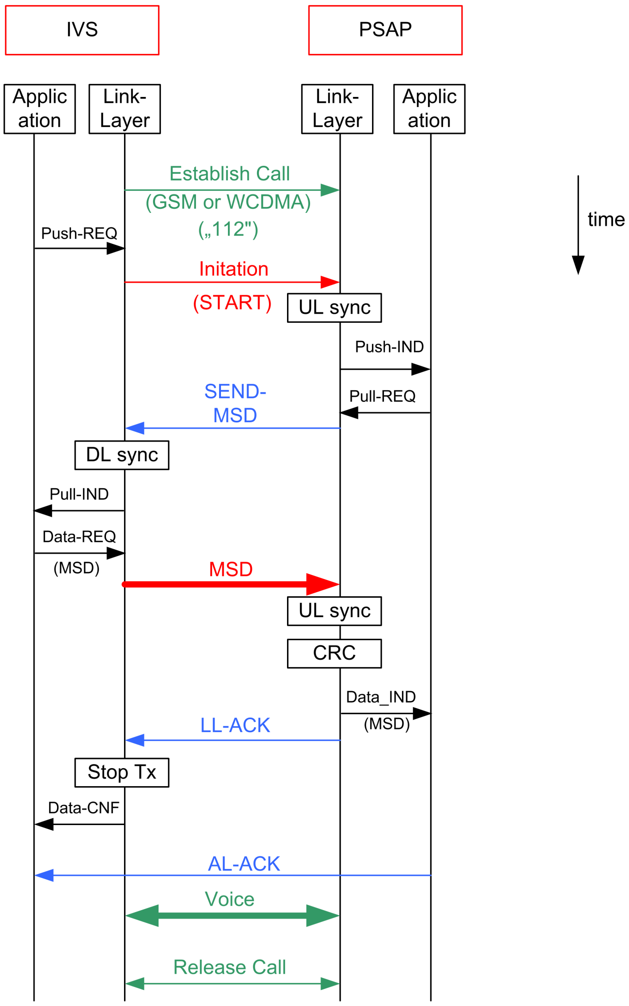


Figure 2-5: Typical ERA-GLONASS sequence. Transmission of the MSD is an iterative process. If the PSAP does not understand the MSD, it transmits NACK and the IVS resends the MSD with a different redundancy version (RV0....7). If the MSD is received successfully, the PSAP sends an ACK.

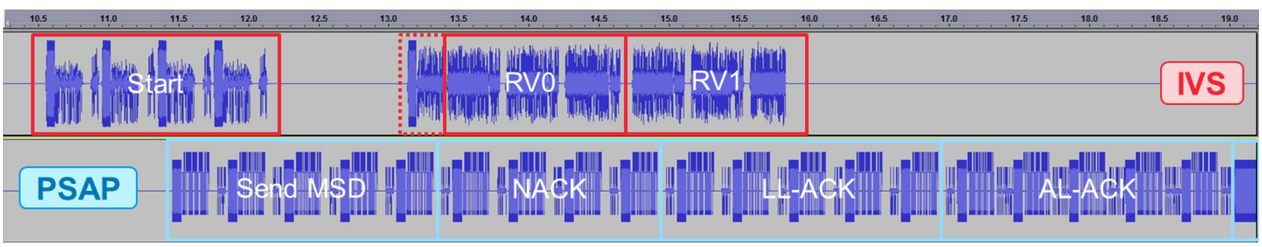


Figure 2-6: The call sequence as seen in the audio file (IVS at the top, PSAP at the bottom).

1. When an emergency occurs, the IVS automatically sets up a connection (emergency call) to the PSAP over GSM or WCDMA ("Establish Call"). START messages are then sent continuously (maximum of 5 times, "Initiation").
2. As soon as the PSAP receives the Call and has recognized the START signal, it transmits the SEND-MSD command to the IVS.

3. Once the IVS has successfully decoded the SEND-MSD ('Downlink Start') command, it transmits one SYNC frame followed by the actual MSD with redundancy version RV0 ("MSD").
4. (Only in Figure 2-6): In the example, the PSAP does not understand MSD RV0 and therefore responds with a NACK ("MSD").
5. (Only in Figure 2-6): The IVS recognizes the NACK and resends the MSD with RV1 to the PSAP ("MSD").
6. The PSAP successfully decodes the MSD RV1 (CRC check), and transmits LL-ACK as feedback. The IVS stops transmitting the MSD.
7. The PSAP transmits AL-ACK, which indicates that the eCall is successfully completed and the data in the MSD is understood. The connection then switches back to voice. A PSAP operator can now speak with the vehicle occupants, for example. Finally, the GSM or WCDMA connection is cleared down ("Release Call")

2.3 Standards

ERA-GLONASS is defined by a number of different standards by the Russian Federal Agency on Technical Regulating and Metrology (GOST) (Please note that this list is not complete):

General

- **GOST 33465-2015**: Global Navigation Satellite System. EMERGENCY RESPONSE SYSTEM IN CASE OF ACCIDENTS. The data exchange protocol of the device/system of calling emergency services with the infrastructure of emergency response system in case of accidents.
- **GOST 33464-2015**: Global Navigation Satellite System. EMERGENCY RESPONSE SYSTEM IN CASE OF ACCIDENTS. The device/system of calling emergency services. General technical requirements.

Compliance Testing

- **GOST 33467-2015**: Global Navigation Satellite System. EMERGENCY RESPONSE SYSTEM IN CASE OF ACCIDENTS. Methods of functional testing of the device/system of calling emergency services and data transfer protocols.
- **GOST 33468-2015**: Global Navigation Satellite System. EMERGENCY RESPONSE SYSTEM IN CASE OF ACCIDENTS. Test methods of the device/system of calling emergency services for compliance with the requirements for loudspeaker communication quality in a vehicle cabin.
- **GOST 33470-2015**: Global Navigation Satellite System. EMERGENCY RESPONSE SYSTEM IN CASE OF ACCIDENTS. Test methods of wireless communication modules in the device/system of calling emergency services.
- **GOST 33471-2015**: Global Navigation Satellite System. EMERGENCY RESPONSE SYSTEM IN CASE OF ACCIDENTS. Test methods for navigation module of the device/system of calling emergency services.

Please note that the numbering of the standards differ in Russia and the Eurasian Customs Union.

Table 2-4: Different Names of Standards

Eurasian Customs Union	Russia
General	
GOST 33465-2015	GOST R 54619-2011
GOST 33464-2015	GOST R 54620-2011
	GOST R 54721-2011
	GOST R 55524-2013
Compliance Testing	
GOST 33467-2015	GOST R 55530-2013
GOST 33468-2015	GOST R 55531-2013
GOST 33470-2015	GOST R 55533-2013
GOST 33471-2015	GOST R 55534-2013:

ERA-GLONASS is harmonized with eCall, thus standards used by eCall also apply Figure 2-7 shows these standards.

Figure 2-7: Overview of the most important standards.

The solution from Rohde & Schwarz presented here fulfills all requirements from these standards. It ensures that the end-to-end conformance test complies with GOST 33465-2015 [1] and the GNSS performance tests with GOST 33471-2015 [6], as discussed in the next chapters.

3 Conformance Testing of IVS

3.1 Why test in the lab?

Testing of ERA-GLONASS devices is necessary at various levels:

- Module level (board)
  - R&D, design verification
  - Production
  - Maintenance
- Device level (IVS)
  - R&D, design verification
  - Coexistence
  - Conformance testing
  - Acceptance testing
  - Production
  - Repair / maintenance
- System level (car)
  - Radiated performance
  - Vehicle body specific
  - Coexistence
  - Desensitization
  - Outdoor, indoor

The solution presented here is intended for tests at the module level and at the device level. This solution permits all basic parameters of the GSM or WCDMA network settings, such as channel or level, to be managed directly. As a result, measurements are fully reproducible. A true emergency call using the emergency number 112 in shielded environment is additionally possible. The effects of uncontrolled settings, such as those made by network operators, is avoided.

3.2 Test Setup

The test solution presented here covers the conformance test for ERA-GLONASS in line with GOST 33465-2015 [1]. Figure 3-1 shows the simplified test setup without an external sound card, Figure 3-2with external sound card.

Select the wanted test setup in the PSAP settings (see Figure 3-6)

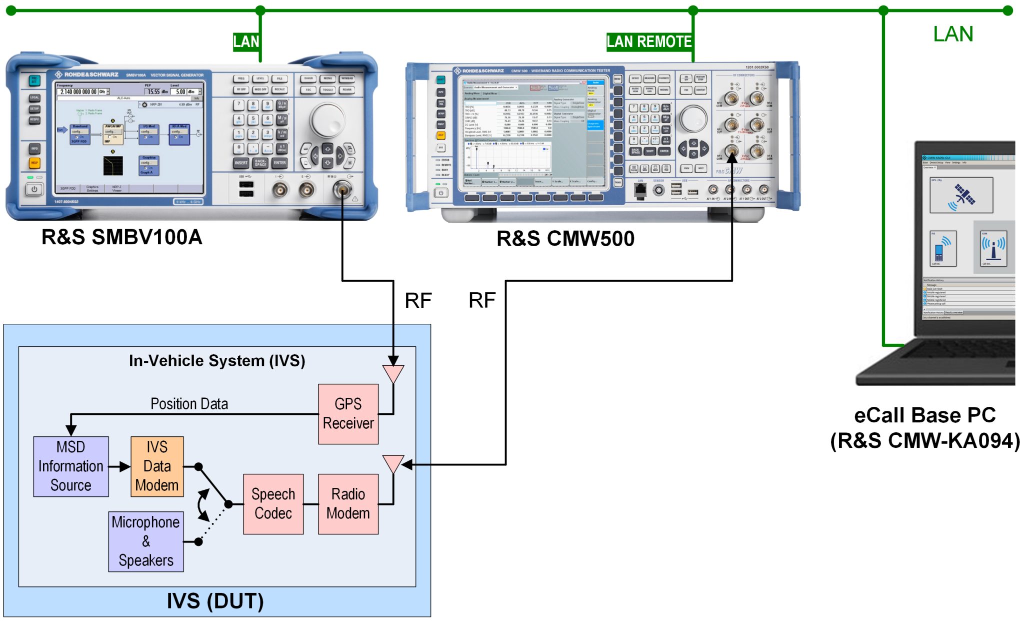


Figure 3-1: Test setup for the IVS conformance test.

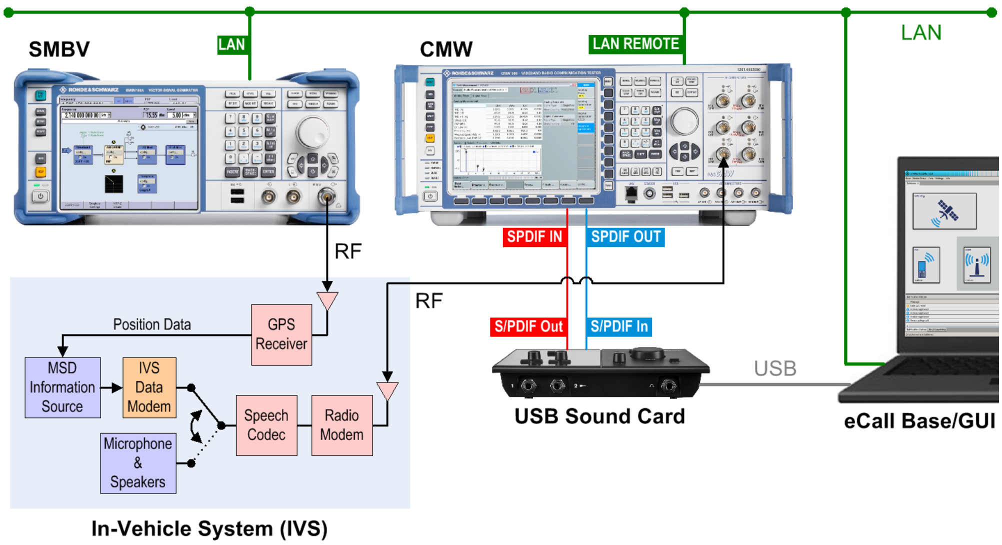
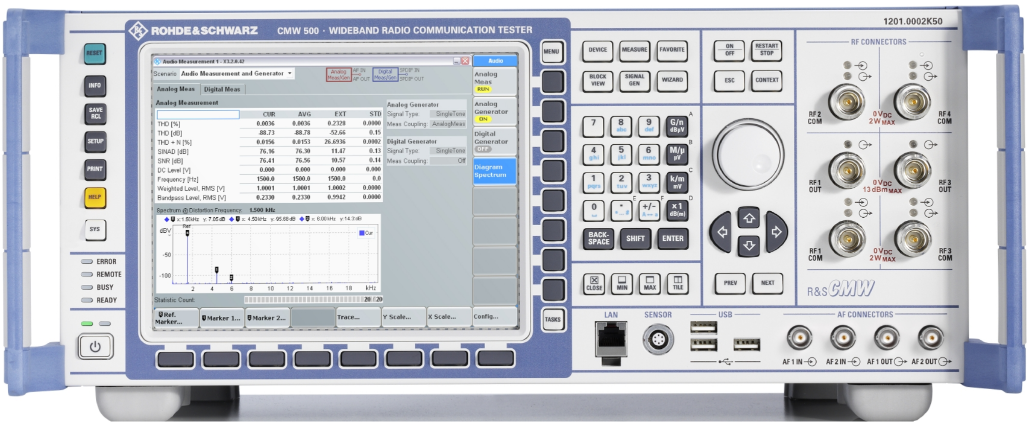


Figure 3-2: Test setup for the IVS conformance test with external sound card.

The CMW simulates the cellular network and provides a GSM (2G) or WCDMA (3G) cell. Via the RF connection to the IVS the signaling (e.g. call setup) over GSM or WCDMA as well as the voice connection takes place. The SMBV supplies simulated GNSS signals, such as GLONASS and or GPS, that are used by the IVS for positioning. The CMW-KA095 (which needs the CMW-KA094) software installed on an external PC simulates the PSAP, sets necessary parameters for ERA-GLONASS on the CMW, and fully remote controls the CMW. It also remote controls the SMBV. It is not necessary here to delve into the operation of the CMW or the SMBV. The audio signal transported via the GSM or WCDMA connection is either handled inside the CMW or routed via a digital SPDIF interface to/from an external sound card. The ERA-GLONASS protocol and the MSD transmission are also handled via the voice connection between the CMW and the IVS. The end-to-end conformance test runs between the PSAP simulator installed on the external PC and the IVS under test.

3.2.1 CMW Radio Communication Tester



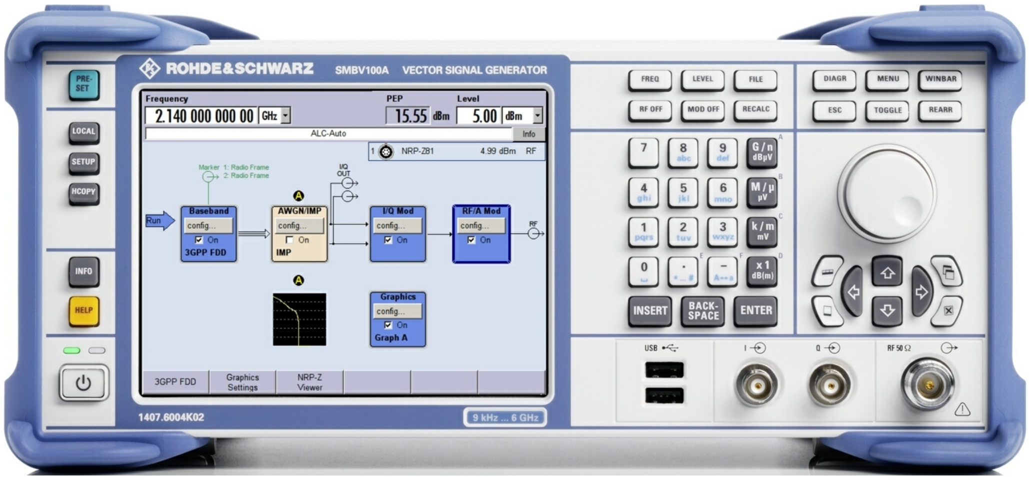
The CMW is the all-in-one test solution for radiocommunications applications such as mobile radio or wireless connectivity. It supports all essential standards, including:

- 2G
  - GSM, EGPRS, EGPRS2, EGDE Evolution and VAMOS
- 3G
  - W-CDMA with HSDPA, HSUPA and HSPA+
  - TD-SCDMA
  - CDMA2000 and 1xEV-DO Rev A/B
- 4G
  - LTE (FDD and TDD), LTE-A incl. MIMO
- Wireless Connectivity
  - Bluetooth
  - WLAN
  - WiMAX

The CMW tests all OSI layers, ranging from the physical layer to end-to-end tests, including both RF tests and protocol tests.

The additional packet data channels can be tested with the CMW standalone without the ERA-GLONASS software KA095.

3.2.2 SMBV Vector Signal Generator



In its role as signal generator, the SMBV supports various wireless communications standards as well as other radio standards. It serves as a specialist for GNSS signals and generates the GNSS signals for ERA-GLONASS.

A brief overview of the GNSS characteristics:

- GPS, Glonass, Galileo, BeiDou, QZSS
- Up to 24 satellites can be simulated
- Automatic satellite handover for unlimited simulation time is supported
- The sky view section displays the current position and state (active or inactive) of the satellites.
- Multipath scenarios such as observed in dense cities (LOS + echoes)
- Moving scenarios simulate the motion of a receiver along a user-defined trajectory such as observed in a moving car (waypoint formats supported such as KML, NMEA files)
- The map view section shows the current position of the receiver, which means that the receiver trajectory can be observed

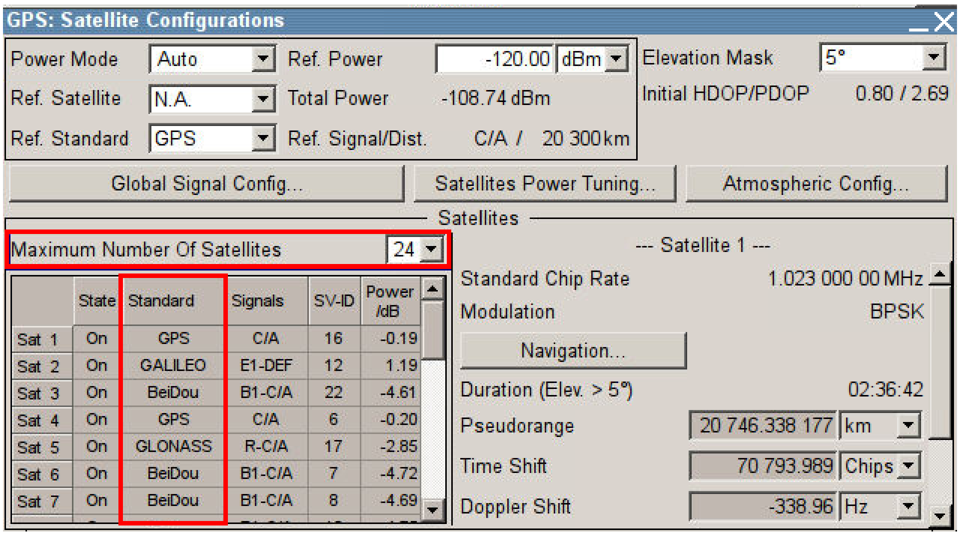


Figure 3-3: Example for a satellite configuration in the SMBV.

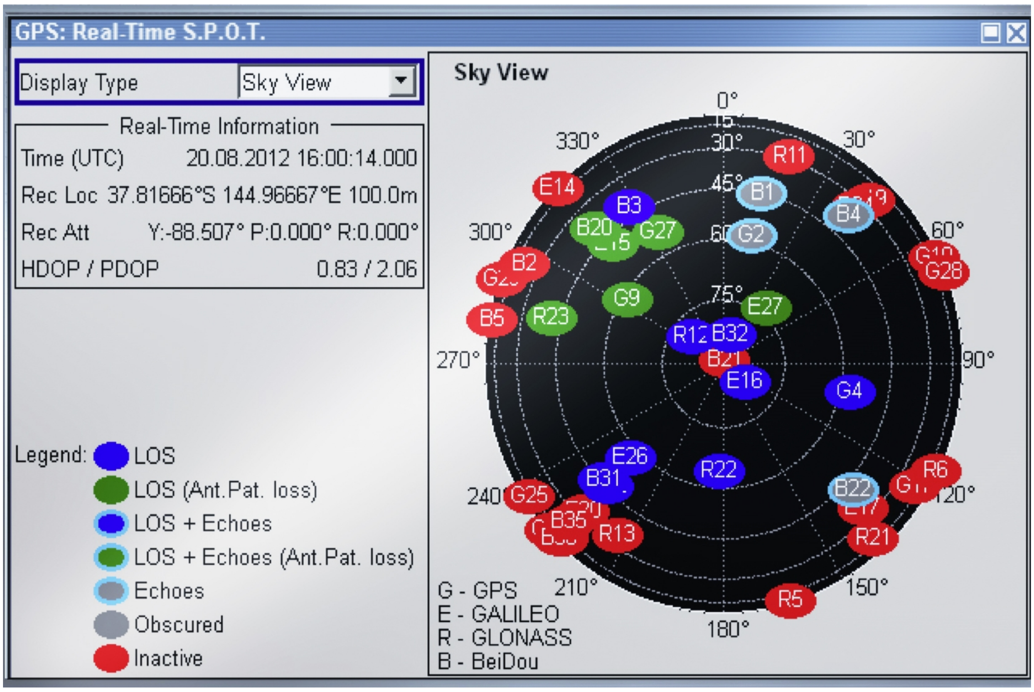


Figure 3-4: The Sky View shows the distribution of the satellites above the local horizon

Needed GNSS options

Following options of the SMBV are needed for eCall / ERA-GLONASS:

Table 3-1: GNSS Scenarios

Scenario	Option SMBV	Configuration	Satellites
GPS - City	K44	static, selectable city	fixed satellite constellation
GLONASS - City	K94	static, selectable city	fixed satellite constellation
GPS - Atlanta	K65	static, Atlanta location	fixed satellite constellation
GPS - Melbourne	K65	static, Melbourne location	fixed satellite constellation
GPS - Melbourne moving	K65	moving, Melbourne location	fixed satellite constellation
GPS - Atlanta Individual Power	K65	static, Atlanta location	variable satellite constellation
GPS - Melbourne Individual Power	K65	static, Melbourne location	variable satellite constellation
GPS - Melbourne moving, Individual Power	K65	moving, Melbourne location	variable satellite constellation
GPS - Santa Cruz Individual Power	K65	static, Santa Cruz location	variable satellite constellation
GPS - Santa Cruz moving, Individual Power	K65	moving, Santa Cruz location	variable satellite constellation
GPS - Atlanta Individual Power (3GPP TS51.010.1 V7.7.0)	K65	static, Atlanta location	variable satellite constellation
GPS - Tokyo Individual Power	K65	static, Tokyo location	variable satellite constellation
GPS - Barcelona moving	K65	moving, Barcelona location	fixed satellite constellation

The GPS City and GLONASS City scenarios both support following cities:

- New York
- Sydney
- Munich
- Moscow
- Tokyo
- Seoul

3.2.3 KA095 PSAP Simulator Software (ERA-GLONASS Test Software)

The CMW-KA095 software runs on an external PC and serves as the PSAP simulator. It also remotely controls (e.g. via LAN) the CMW and the SMBV. The SW runs the end-to-end conformance test. The MSD is decoded and the measurement results are displayed:

- PSAP simulation for ERA-GLONASS over GSM or WCDMA
- MSD transmission time
- Time since call establishment
- Time since start trigger (from PSAP)
- MSD decoding according to CEN EN 15722:2015 and GOST R 54620-2011 A1 for every redundancy version and uplink data part
- Optional recording of un-decoded signal from IVS
- Details on PUSH and SYNC indications
  - Timing
  - Count
- SMS protocol with more than 200 SMS commands based on GOST R 54619-2011
- Optional: Fixed position GPS/GLONASS simulation / GPS moving scenario

3.3 The First ERA-GLONASS Measurement: Basic Operation

The following sections describe the general process for setting up an ERA-GLONASS. The procedures guide you through the test system preparation and an ERA-GLONASS with MSD transfer from the IVS to the simulated PSAP.

Use the test setup shown in Figure 3-2.

The IVS to be tested is connected via an RF cable to a CMW that simulates a GSM or WCDMA cell. Via this connection, the IVS establishes a GSM emergency call and transfers an MSD to the CMW. The GNSS signal is provided via an RF cable to the IVS.

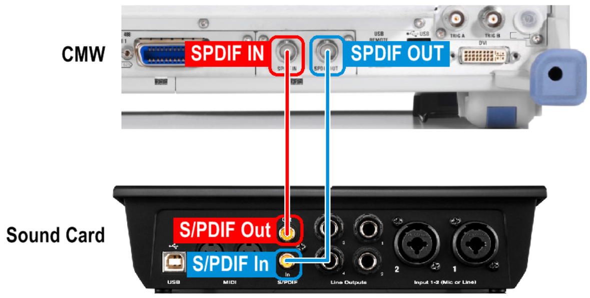
The audio signal transported via the GSM or WCDMA connection is routed via a digital SPDIF interface to/from an external sound card.

The ERA-GLONASS application base and GUI are installed on a PC. The application simulates a PSAP, controls the CMW via the LAN and accesses the sound card via a USB connection.

3.3.1 Preparing the Test Setup

The following steps prepare the test setup and start the ERA-GLONASS application.

1. Prepare the test setup shown in :
   a) Connect the PC to the LAN.
   b) Connect the "LAN REMOTE" connector on the rear panel of the CMW to the LAN.
   c) Connect the "LAN" connector on the rear panel of the SMBV to the LAN.
   d) Connect the GSM RF connector of the IVS to "RF 1 COM" on the front panel of the CMW.
   e) Connect the GPS RF connector of the IVS to "RF" on the front panel of the SMBV.
   f) (optional for use of external soundcard (CMW-Z94): Connect the rear panel of the sound card to the rear panel of the CMW:



   The cables are delivered together with the sound card (CMW-Z94).
   g) Connect the sound card via a USB cable to the PC.

2. Set up the PC. The required steps are described in detail in the CMW-KA095 User Manual, section "Preparing the Test System for Use".

   The following list gives an overview:
   - Install the ERA-GLONASS application software.
   - Install the .NET Framework 4.0.
   - Install a VISA library with development support for .NET Framework 4.0.
   - Install the sound card driver and configure the sound card.
   - Configure the sound device settings of the PC.

- Make the license CMW-KA095 (and for the CMW-KA094) available, for example by attaching a smart card with the license.

3.3.2 SIM Card (UICC)

The Subscriber Identity Modul (SIM-Card) allows mobile devices (UE) identification and authentication in a mobile network. Typically the network providers hand out SIM-Cards to the endusers to allow access to the provider's mobile network. When switched on, the UE searches for basestaions with certain codes (Mobile Network Code - MNC). With the registration the UE transmits the IMSI (International Mobile Subscriber Identity), which is stored on the SIM card. Additionally, certain codes and algorithm saved on the SIM-Card allow ciphering of signaling data and user data. E.g. in WCDMA, a certain authentication code is needed to register to the network and to setup a connection successfully.

Rohde & Schwarz provides special SIM-Cards for testing UE's with the CMW. With the use of these SIM-cards, the UE registers to the network provided by the CMW. Thus the UE does not register to 'real' mobile networks.

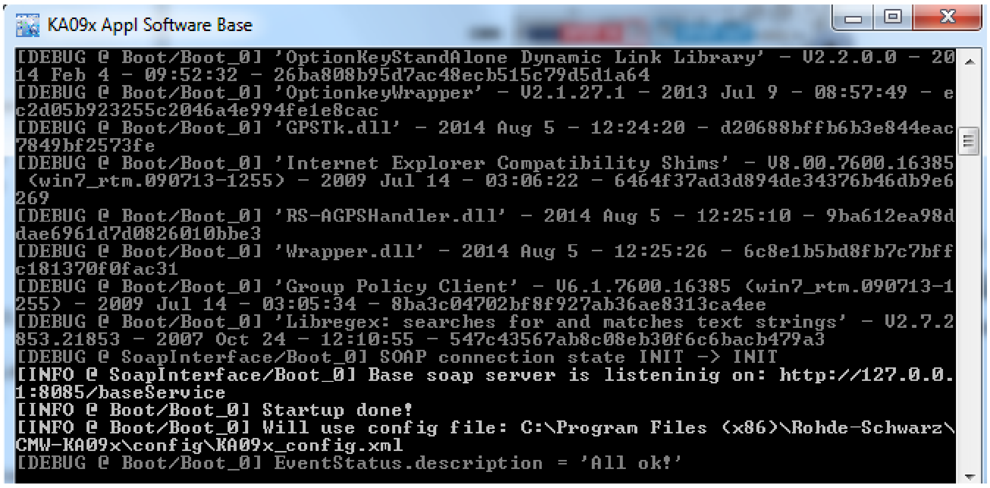
3.3.3 Starting and Configuring the ERA-GLONASS Test Software (KA095)

The following steps start the eCall test software and adapt the settings to the test setup.

The procedure explains the required steps, starting from the default settings. If you have already modified settings, you may reset the settings to their default values by deleting or renaming the following configuration file before starting the application: `%APPDATA%\Rohde-Schwarz\CMW-KA09x_GUI\<version>\user.config`. The eCall test software consists of two parts:

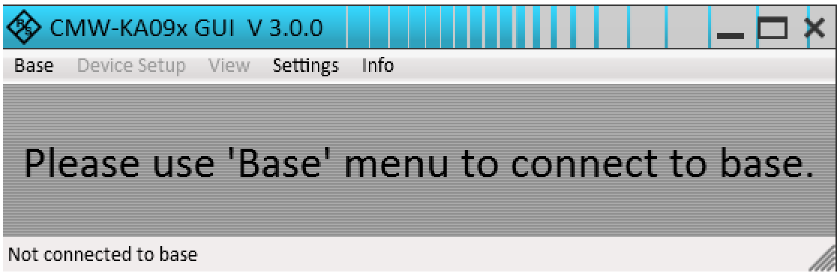
1. Start the ERA-GLONASS test software base:
   "Windows Start" menu > "All Programs" > "R&S CMW-KA09x" > "KA09x Application Base"

   A console window opens. A successful start is indicated as "Startup done!".

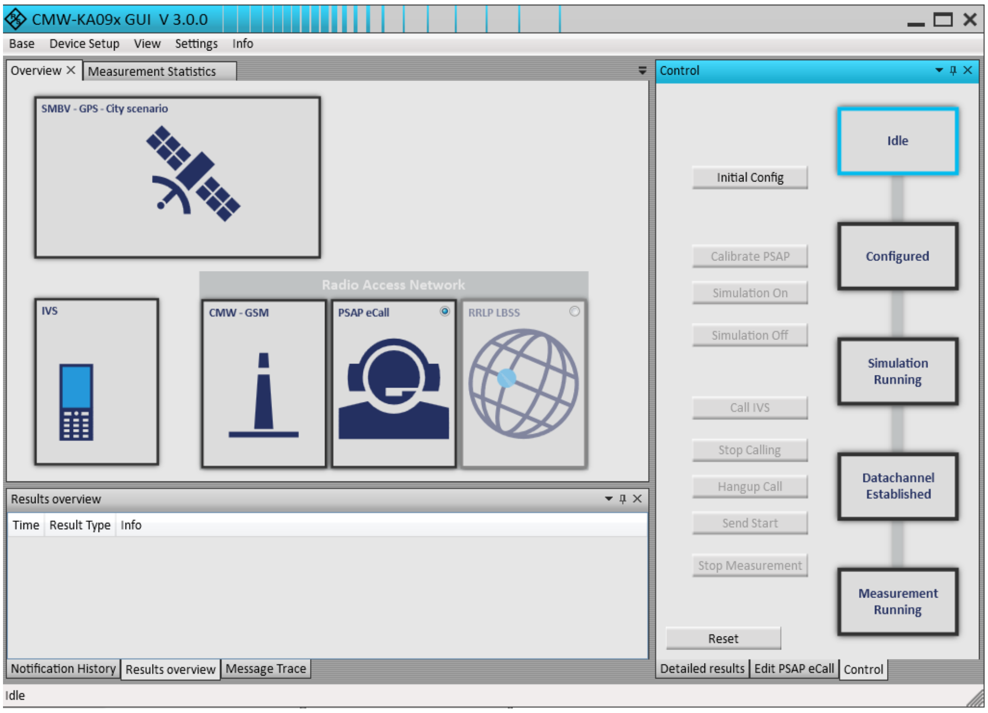


2. Start the ERA-GLONASS test software GUI:
   "Windows Start" menu > "All Programs" > "R&S CMW-KA09x" > "KA09x Appl Software GUI"

   The "CMW-KA09x GUI" window opens.

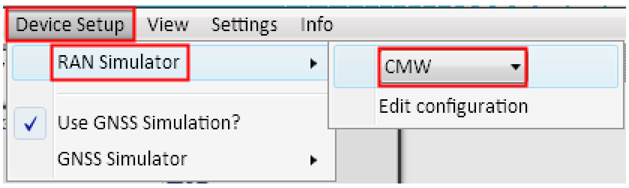


3. Connect the GUI to the base:
   "Base" menu > "Connect"
   After a successful connection, the GUI looks as follows.

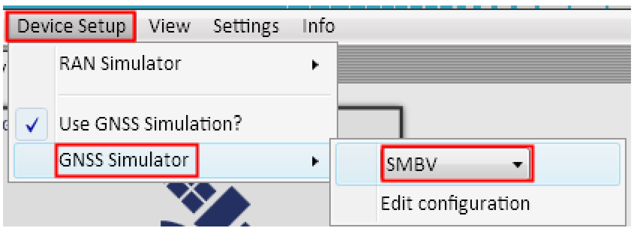


The status bar indicates the base state "Idle".

4. Select the CMW as RAN simulator:
   "Device Setup" menu > "RAN Simulator" > "CMW"

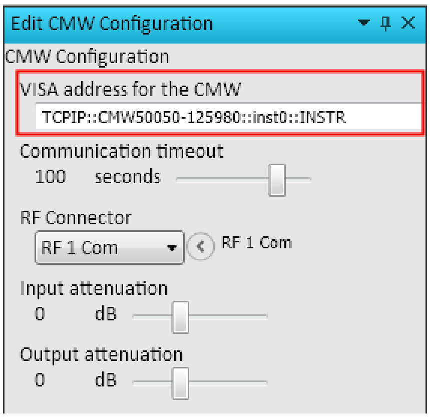


5. Select the SMBV as GNSS simulator:
   "Device Setup" menu > "GNSS Simulator" > "SMBV"

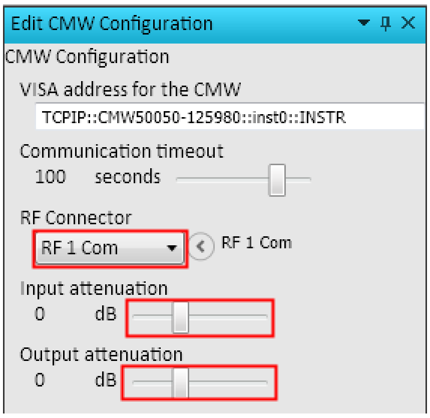


6. Display the CMW settings:
   "Device Setup" menu > "RAN Simulator" > "Edit configuration"

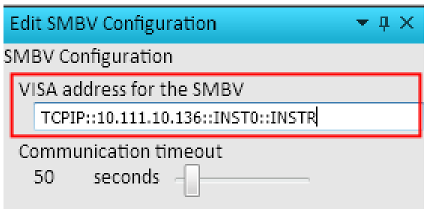
7. Enter the VISA address string or the TCPIP address of your CMW.



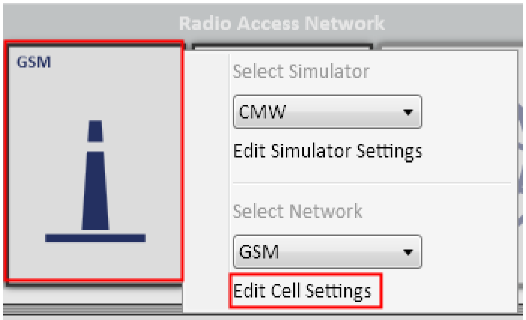
8. Configure the other CMW settings:
   a) Select the RF connector to be used for the GSM signal
   b) Configure the input and output attenuations according to the attenuation of your RF cable.



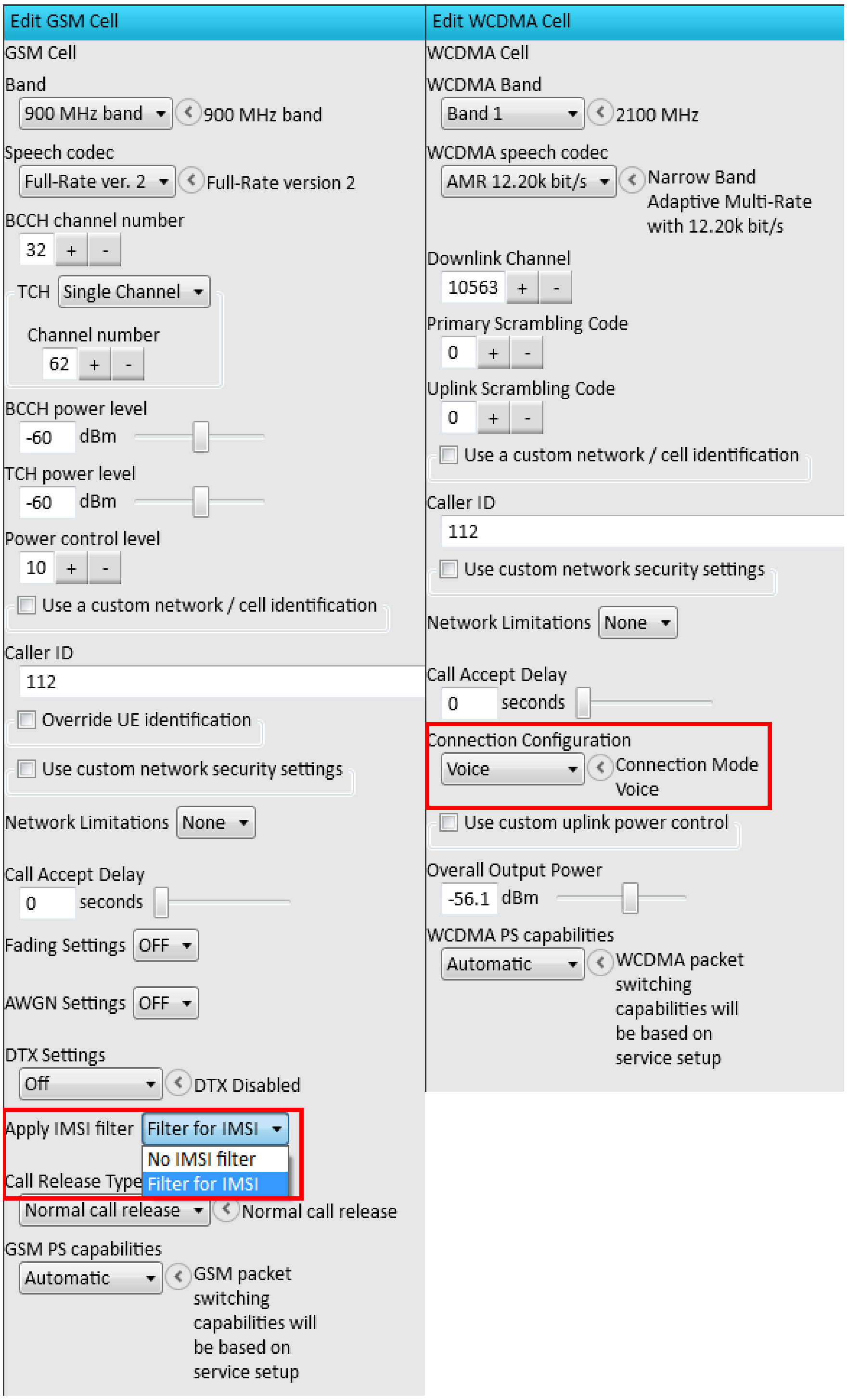
9. Display the SMBV settings:
   "Device Setup" menu > "GNSS Simulator" > "Edit configuration"
10. Enter the VISA address string or the TCPIP address of your SMBV.



11. Right-click the basestation area in the section Radio Access Network on the left. Here select the network (GSM or WCDMA) and klick "Edit Cell Settings".



The cell settings (GSM or WCDMA) are displayed on the right. Please note that in WCDMA the **Connection Configuration** must be set to "Voice" always.



12. Check that the cell settings are compatible to your IVS.
    If required, modify the settings, for example the band, the channel number and the power levels.
    Special settings for SIM-Cards can be handled via "Use custom network security settings"

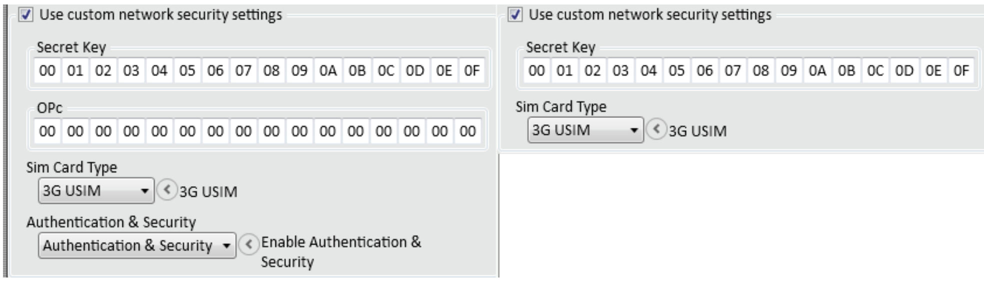
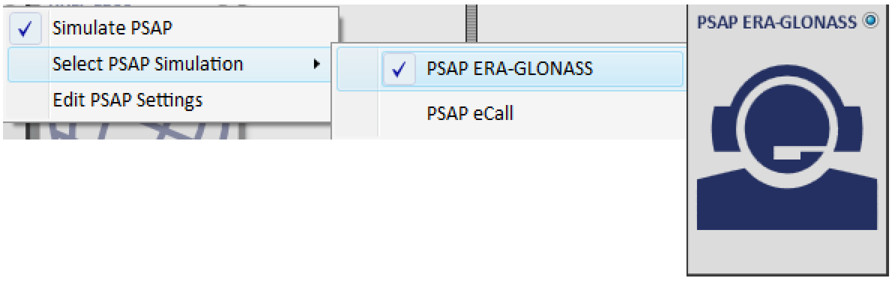


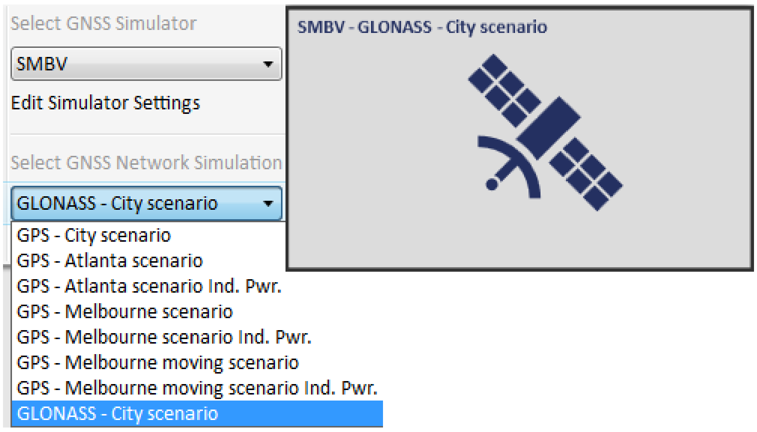
Figure 3-5: Important settings for SIM-Cards. Left: WCDMA; right: GSM

If you subsequently modify settings after the initial configuration, click "Update Configuration" at the bottom to apply the changes.

13. Right-click the PSAP area in the section Radio Access Network on the left. Here select the PSAP simulation "ERA-GLONASS"



14. To use GLONASS, right-click SMx area on the left. Here select the "GLONASS - City Scenario"



a) Configure the PSAP to your Audioboard settings (see also Chapter 3.2, "Test Setup", on page 14).

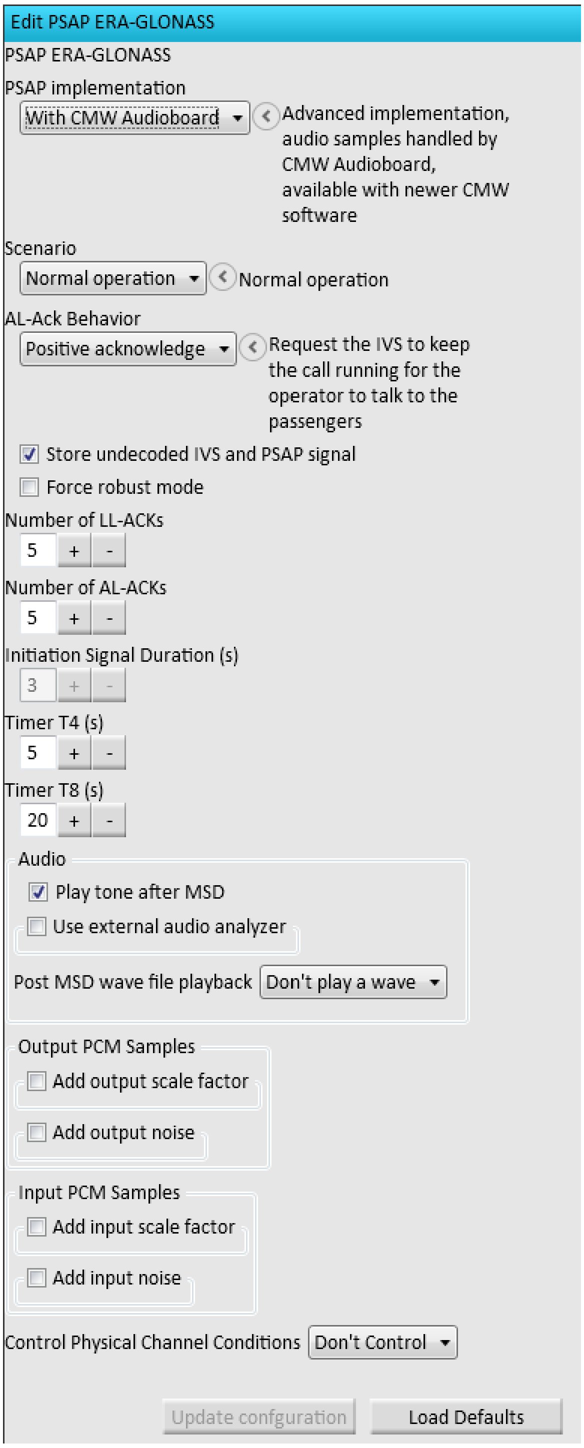
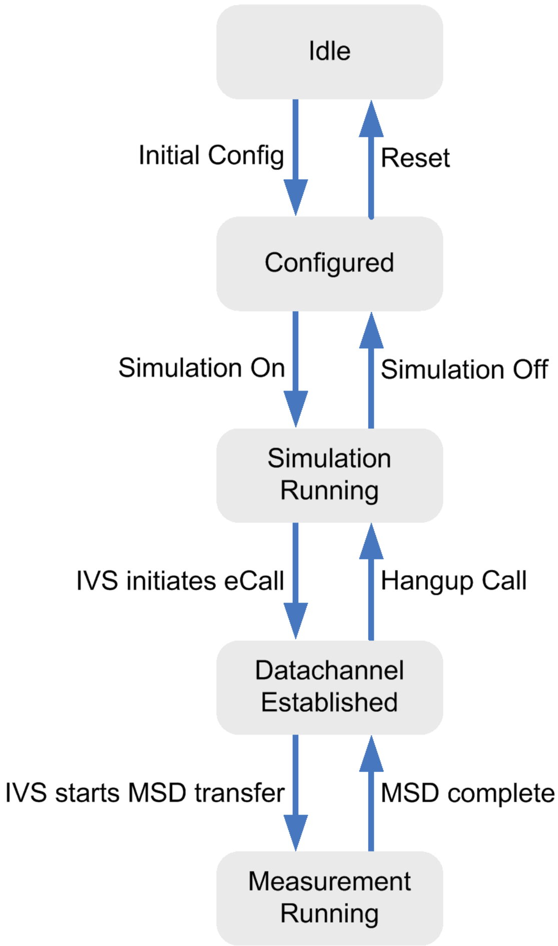


Figure 3-6: PSAP settings: two different test setups are possible: internal CMW audioboard or external soundcard (CMW-Z94)

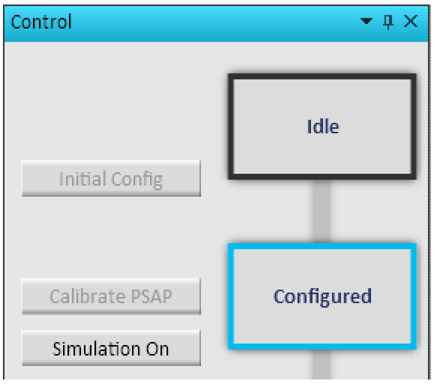
3.3.4 Performing an ERA-GLONASS call

The steps in this section set up a GSM emergency call, transfer an MSD from the IVS to the ERA-Glonass application and analyze the MSD.

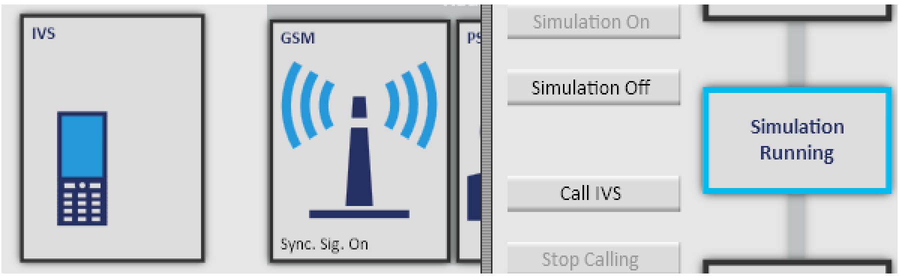
The following figure shows the related ERA-GLONASS application states. The start is in "Idle" state, goes through the entire state machine to "Measurement Running" and ends up in state "Datachannel Established".



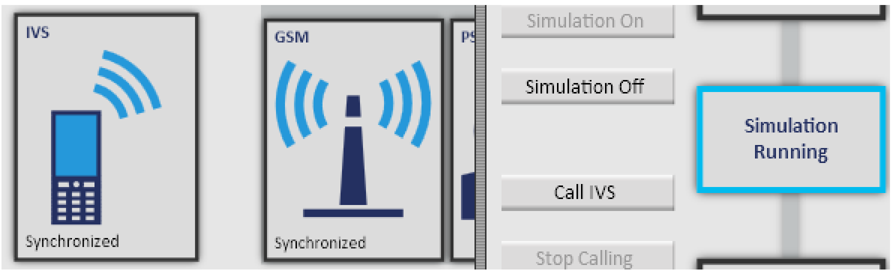
1. On the right, select the "Control" view.
2. Click "Initial Config".
   The application validates the settings and configures the CMW and the SMBV.
   This may take some time, especially if you perform this action for the first time after booting the instruments.
   After a successful initial configuration, the state in the "Control" view changes from "Idle" to "Configured".



3. Click "Simulation On".
   The PSAP, GNSS and GSM simulations are started. The GSM cell signal is switched on. This may take some time.
   After a successful simulation start, the state in the "Control" view changes from "Configured" to "Simulation Running".
   The GSM area displays an active antenna.



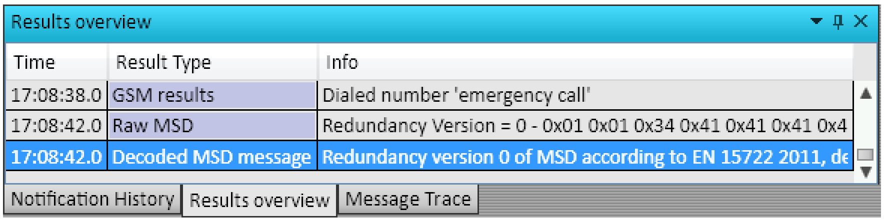
4. Switch on your IVS and let it register.
   A successful registration is indicated in the "Notification History" view. The labels in the IVS area and the GSM area also indicate that the IVS has registered ("synchronized").



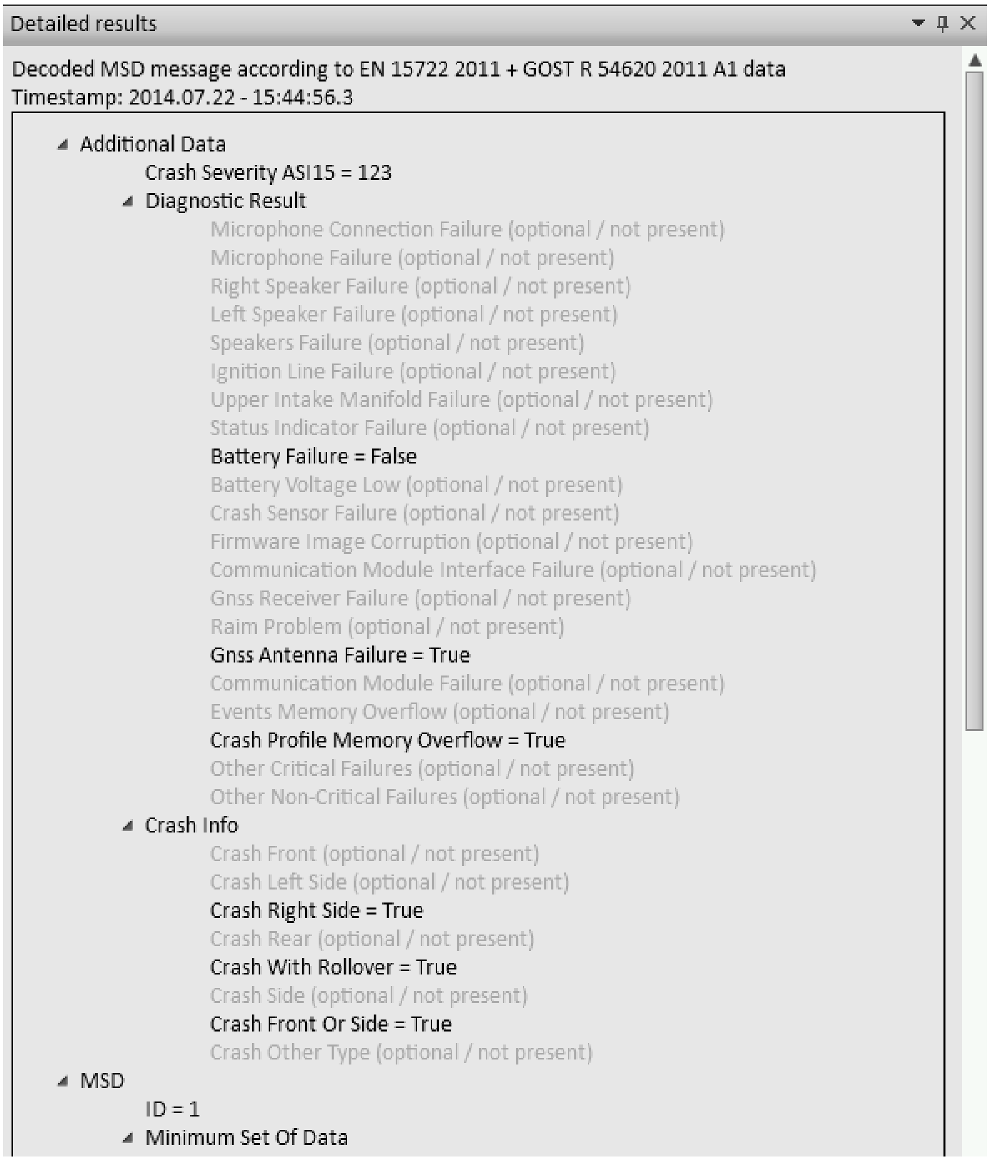
5. Initiate an ERA-GLONASS call at your IVS.
   You can monitor the call progress in the "Overview" tab and the "Control" view. The following table lists the states for a successful ERA-GLONASS call, starting in the first row and ending in the last.

IVS State	GSM State	PSAP State	Control State	Explanation
Synchronized	Synchronized	PSAP initialized[1)]	Simulation Running	IVS registered and synchronized
Connecting	Connecting			Call setup in progress
Call Established	Call Established	Ready for ERA-GLONASS call	Data Channel Established	Data channel between IVS and PSAP established, no MSD transmission
		Sending Start	Measurement Running	PSAP requests MSD from IVS
		Receiving 1st MSD part		PSAP receives first MSD part
		Sending NACK		MSD part received
		Sending LL-ACK		MSD reassembled and sent to higher layers
		Sending HL-ACK		MSD decoded at application layer
		Ready for ERA-GLONASS call	Data Channel Established	Call still established, no MSD transmission Sine tone transmission to the IVS
[1)] For the first ERA-GLONASS call, no state is indicated				

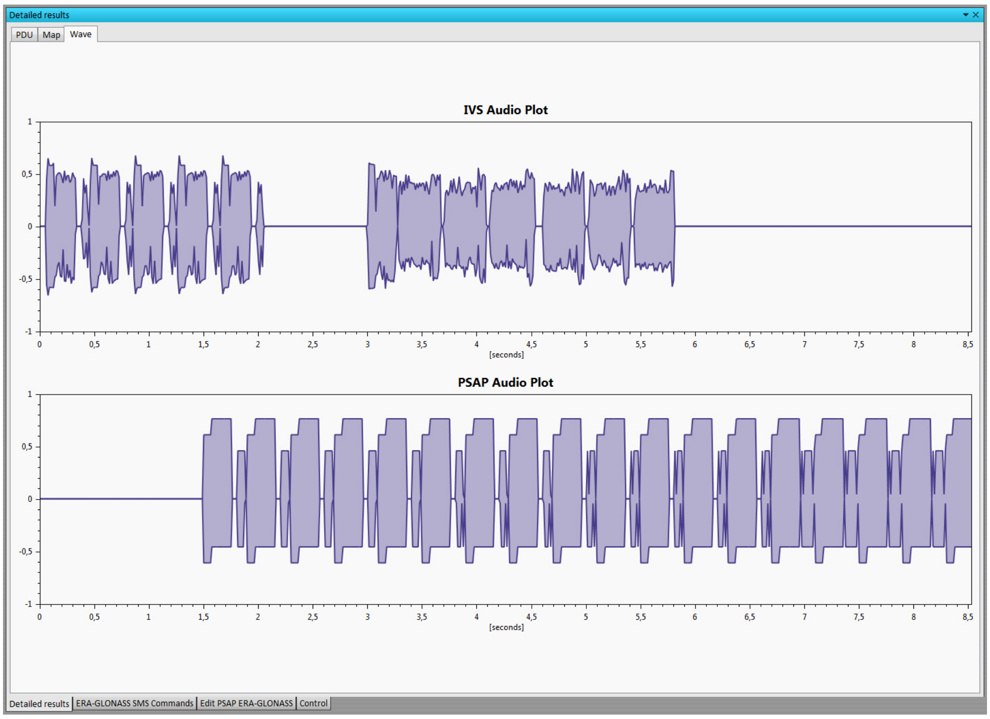
6. Select the "Results Overview" view on the left.
7. Double-click the row "Decoded MSD message".



The "Detailed Results" view is automatically displayed on the right. It lists the MSD message contents. Here the additional data of ERA-GLONASS is shown.

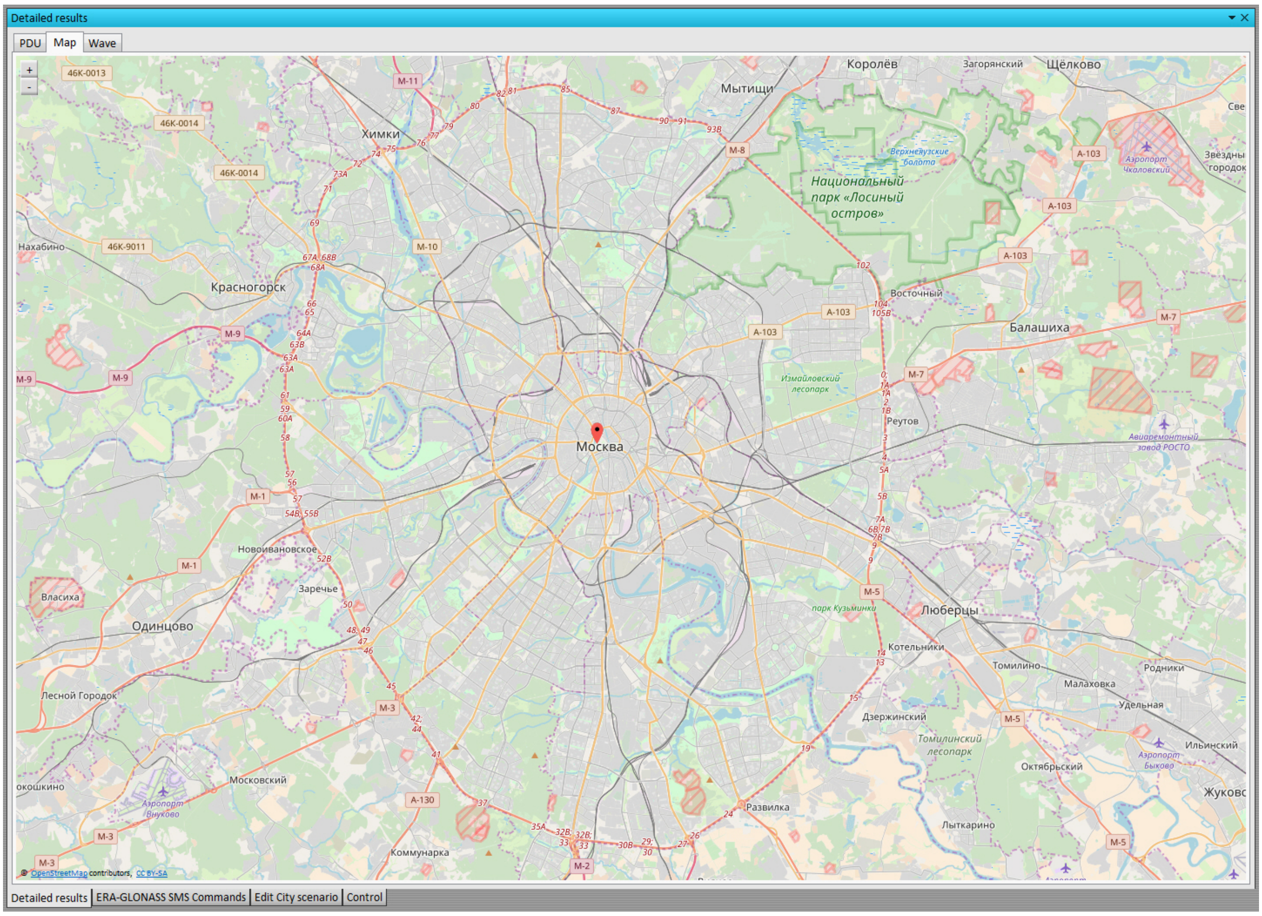


8. The ERA-GLONASS test software records two raw `wav` files for each call, one for the uplink signal from the IVS and one for the downlink signal from the PSAP. "Detailed results" displays the tracks under the tab "Wave". The following example shows the IVS part on top and the PSAP part on bottom. For more detailed explanation refer to Figure 2-2.



The files are stored additionally in the installation directory of the ERA base application, for example at
`%PROGRAMFILES(X86)%\Rohde-Schwarz\CMW-KA09x\Complementary`.
To check the file contents, open the files with an external audio editor.

a) The ERA software shows the recorded position under the tab "Map"

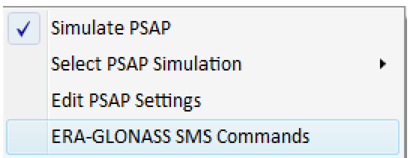


Please note that the software needs an online connection to show the map.

3.3.5 SMS commands with ERA-GLONASS

ERA-GLONASS provides the possibility to communicate between the PSAP and the IVS with the help of SMS, e.g. for sending configuration (change or query parameters) or to transmit e.g. the MSD via SMS.

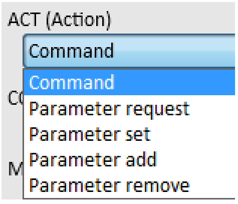
Right-click the PSAP area in the section Radio Access Network on the left. Here select the "ERA-GLONASS SMS commands"



The settings are diplayed on the right under tab "ERA-GLONASS SMS commands"

MSD via SMS

To command the IVS to tranmsit the MSD via SMS, the PSAP has to transmit a command under ACTion.



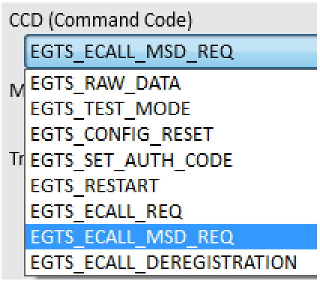
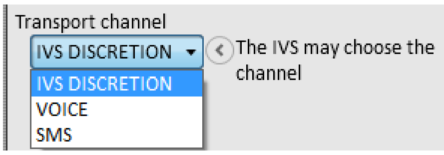


Figure 3-7: Different command codes



The command code (CCD) "EGTS_ECALL_MSD_REQ" tells the IVS to transmit the MSD. The transport channel depends on the used IVS, e.g. set to "SMS"

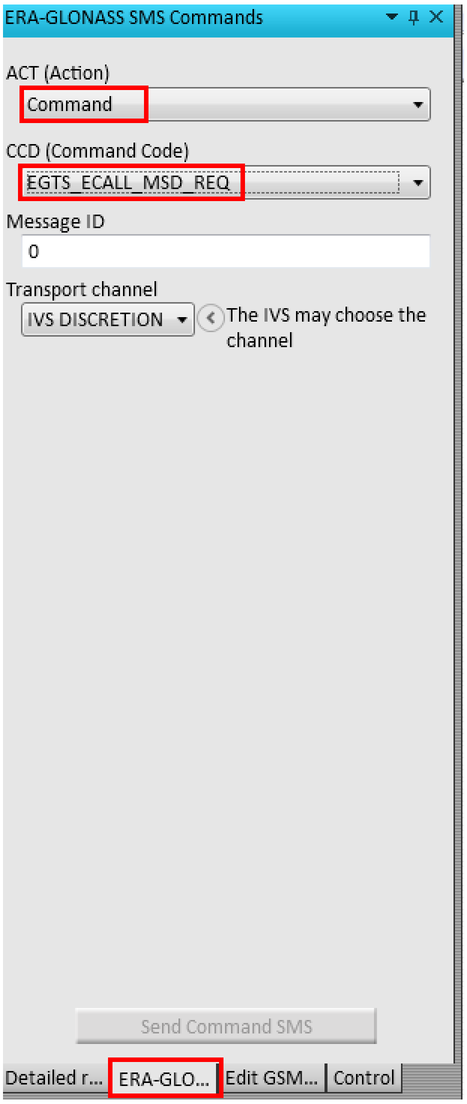
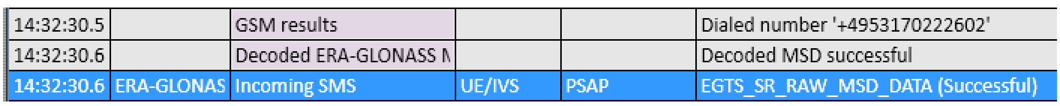


Figure 3-8: The SMS command settings to urge the IVS to send the MSD via SMS. The transport channel may differ and depends on the IVS.

Please note that the IVS must be registered (status "synchronized") before a SMS can be transmitted.

The IVS then shall answer by sending a SMS with MSD. Figure 3-9 shows the entry in the message trace and Figure 3-10 shows the decoded SMS with MSD



14:32:30.5		GSM results			Dialed number '+4953170222602'
14:32:30.6		Decoded ERA-GLONASS N			Decoded MSD successful
14:32:30.6	ERA-GLONAS	Incoming SMS	UE/IVS	PSAP	EGTS_SR_RAW_MSD_DATA (Successful)

Figure 3-9: The incoming SMS with MSD in the message trace

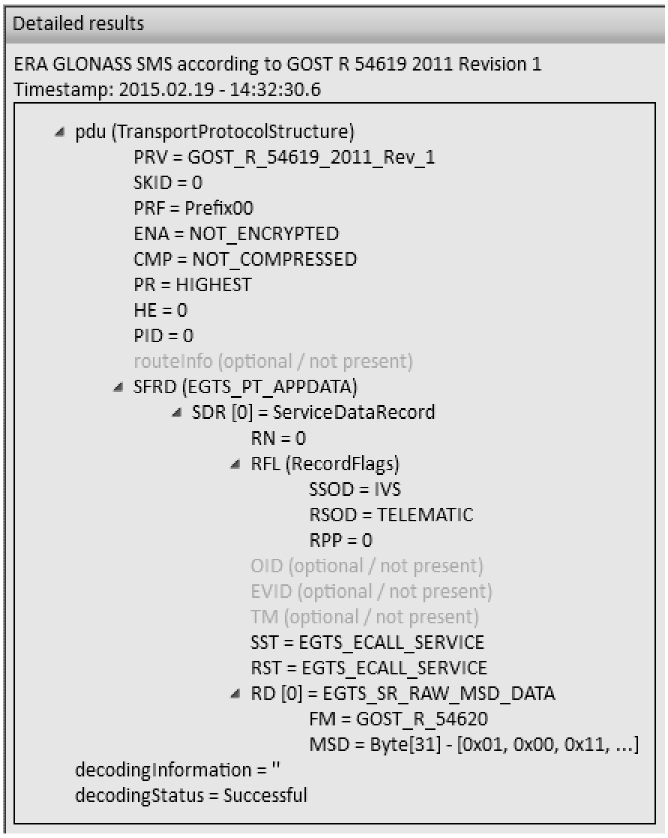
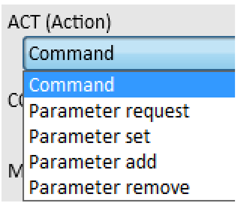


Figure 3-10: The decoded SMS with MSD

Parameter actions via SMS

ERA defines different actions to handle parameters via SMS, the PSAP has to transmit a Parameter action under ACTion. As an example the PSAP sets the Unit_ID of the IVS to '1'



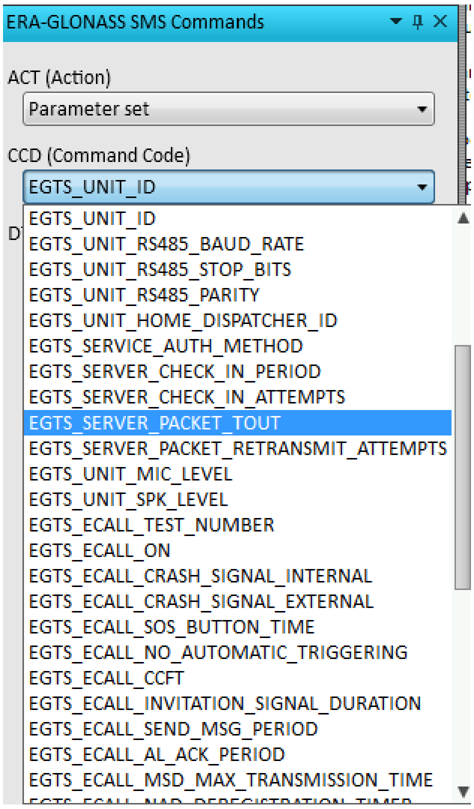


Figure 3-11: Different command codes to set parameters

The command code (CCD) "EGTS_UNIT_ID" tells the IVS to set the unit ID (in this case to '1').

Figure 3-12: The SMS command settings to set the ID parameter.

Please note that the IVS must be registered (status "synchronized") before a SMS can be transmitted.

3.3.6 Troubleshooting

The following troubleshooting hints are related to the procedures in the previous sections.

Connecting GUI to base fails

If the connection fails, check the console window for error messages. Typical errors:

- "No license found": Check that a smartcard with license CMW-KA095 is connected to the PC. You can check the smart card contents with the "R&S License Key Manager", see Windows Start menu. Close the license key manager before you attempt a connect.

- "Client register with <URL> failed": Check that the URLs in the "Base" menu are correct. The default settings assume that the base and the GUI are installed on the same PC and the port 8085 is free.

Initial config fails

If the transition from state "Idle" to state "Configured" fails, check the "Notification History" view. Typical errors:

- "Base rejected CMW... viOpen failed...": Opening a remote control connection to the CMW failed.
  – Check that you can reach the CMW via the LAN. For example send a ping request from the PC to the CMW.
  – Check the VISA address string entered in on page .
- "GSM bcch number invalid for band..." or similar: Check the GSM cell settings.

Registration fails

If the registration of the IVS to the GSM or WCDMA cell fails, check the following:

- Is the IVS switched on?
- Is the RF cabling between the IVS and the CMW ok?
- Is the correct RF connector configured in the CMW settings?
  For CMW settings, see on page .
- Are all GSM or WCDMA cell settings compatible to your IVS? For cell settings, see on page .

MSD transfer fails

If the transfer of an MSD to the PSAP fails, consider the following hints for troubleshooting:

- Observe the IVS and GSM states in the overview. Is the state "Call Established" reached or does the call setup fail?
- If the call is established, monitor the PSAP state in the overview. Which is the highest reached state? Is an MSD part received but cannot be decoded or is no MSD received at all?
- If an MSD is received, but cannot be decoded, you can check the raw MSD data. In the "Results Overview" double-click the row "Raw MSD".
- For further hints, check the "Message Trace" view. It displays the exchanged eCall protocol messages.
  Or check the "Log" view. If it is not visible, activate it via the "View" menu > "Show" > "Log".

3.4 Measurements in Line with Specification

The conformance tests for an IVS are specified in GOST R 55530-2013 [1], chapter 6 "Methods used in tests of conformity to functional requirements". These tests are listed in Table 3-2.

To perform a specific test with the ERA test software, configure the PSAP settings as indicated in the table. To access the settings, right-click the PSAP area on the left and click "Edit PSAP Settings". The "Edit PSAP" view opens on the right.

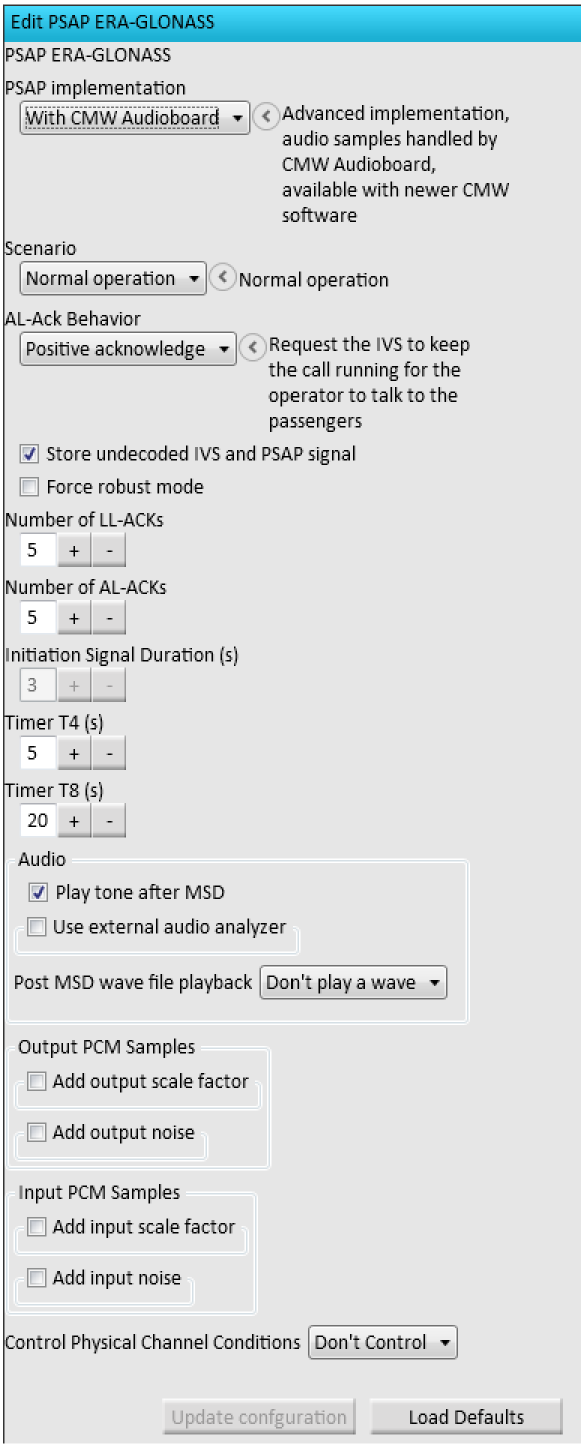


Figure 3-13: Default PSAP settings.

If "Use external audio analyzer" is enabled, the audio signal is automatically switched at the end of the MSD transmission to the analog audio in/out 1 connector of the CMW (instead of S/PDIF in/out). This allows the connection of an external audio analyzer like the UPV, a handset like the CMW-Z50 or artificial heads like need for tests according to ITU-T specification [5] .

Table 3-2: Conformance tests

Number	Name	PSAP Settings	Check / Pass Condition
6.1	Checking MSD transfer in automatic mode		
6.1.1	Checking MSD transfer using in-band modem	Default	Check MSD contents: "Automatic activation = True", "Test call = False"
6.1.2	Checking MSD transfer SMS	Use ERA-GLONASS SMS command: "Action= command", "Command code = EGTS_ECALL_MSD_REQ", "Transport channel = SMS"	Check MSD contents: "Automatic activation = True", "Test call = False"
6.2	Checking MSD transfer in manual mode		
6.2.1	Checking MSD transfer using in-band modem	Default	Check MSD contents: "Automatic activation = False", "Test call = False"
6.2.2	Checking MSD transfer SMS	Use ERA-GLONASS SMS command: "Action= command", "Command code = EGTS_ECALL_MSD_REQ", "Transport channel = SMS"	Check MSD contents: "Automatic activation = False", "Test call = False"
6.3	Checking that transferred MSD contains last known vehicle location as for detection time of RTA event	Default	Check MSD contents: "last known vehicle loacation determined by GNSS (parameter vehicle_location)"
6.4	Checking that transferred MSD contains expected last known vehicle location as for detection time of RTA event	Default	Check MSD contents: no information on the last known vehicle location but "expected last known vehicle loacation(parameter Recent_vehicle_location_n-1)
6.5	Checking that transferred MSD contains valid vehicle location data	Default	Check MSD contents: "information on geographic loacation of the vehicle (coordinates determined by GNSS)"
6.6	Checking that transferred MSD contains vehicle movement direction data	Default	Check MSD contents: "information on geographic loacation of the vehicle (coordinates determined by GNSS), information on movement direction of the vehicle (parameter VehicleDirection)"
6.7	Checking that loud voice communication is possible during emergency calls	Default	check duplex voice communication during call
6.8	Checking IVS status indicators (for IVS in auxiliary equipment configuration)	not available / out of scope	
6.9	Checking IVS operation in Test mode	Default	Check MSD contents: "ECALL_TEST_NUMBER*
6.10	Checking IVS operation in "Service Station" mode (for IVS in auxiliary equipment configuration)	Default	Check MSD contents

Number	Name	PSAP Settings	Check / Pass Condition
6.11	Checking IVS operation in "Software Downloading" mode (for IVS in auxiliary equipment configuration	not available / out of scope	
6.12	Checking UIM (for IVS in auxiliary equipment configuration)	not available / out of scope	
6.13	Checking internal memory of IVS	not available / out of scope	
6.14	Checking operation of IVS backup battery and power supply	not available / out of scope	
6.15	Checking IVS registration in network	Default	Cell Status: Registered ("synchronized")
6.16	Checking electric power supply and power consumption requirements (for IVS in auxiliary equipment configuration)	not available / Out of Scope	

Table 3-3: Conformance tests

Number	Name	PSAP Settings	Check / Pass Condition
6.17	Checking transfer of SMS command for setting SMS transmission number when SMS is used as a redundant data channel	Use ERA-GLONASS SMS command: "Action= Parameter set", "Command code = EGTS_UNIT_ID",set wanted ID (1) and specify SMS number	check new SMS number via diagnostic software
6.18	Checking transfer of SMS command for setting emergency call number used in tests	Use ERA-GLONASS SMS command: "Action= Parameter set", "Command code = EGTS_UNIT_ID",set wanted ID (1) and specify test call number	check new test call number via diagnostic software
6.19	Checking transfer of SMS command for initiation of test emergency call	Use ERA-GLONASS SMS command: "Action= command", "Command code = EGTS_ECALL_MSD_REQ", "Transport channel = VOICE"	Check MSD contents: "Automatic activation = False", "Test call = True"
6.20	Checking transfer of SMS command for repeated MSD transfer in regard to RTA event recorded earlier	Use ERA-GLONASS SMS command 2 times: "Action= command", "Command code = EGTS_ECALL_MSD_REQ", "Transport channel = SMS"	receive 2 MSD's: Check MSD contents are teh same except: location and time
6.21	Checking packet transmission of firmware data specific to a given IVS type (for IVS in auxiliary equipment configuration)'	not available / Out of Scope	
6.22	Checking that emergency call button is protected from accidental pressing	not available / Out of Scope	
6.23	Checking that backlighting of emergency call button is available	not available / Out of Scope	

3.5 Automated tests: remote control

The software CMW-KA09x comes with the KA09x BASE application software which is the main part, and the GUI for manual control (KA09x GUI application software). The GUI part controls the base part via SOAP. To run automated tests, you can control the BASE via SOAP by your own developed program or by CMWrun provide by Rohde & Schwarz.

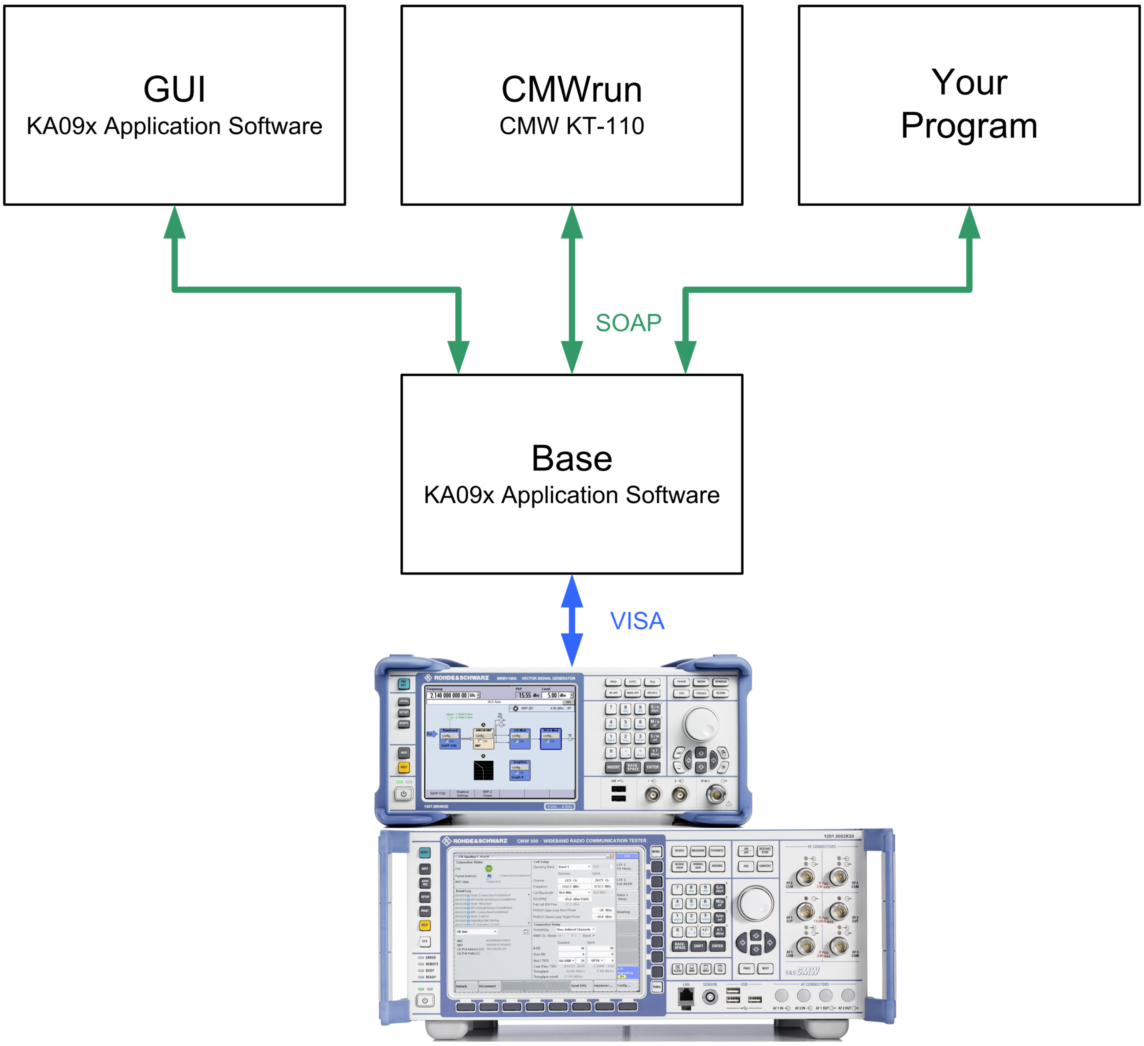


Figure 3-14: Three ways to control the BASE via SOAP. The GUI for manual control is provided together with the BASE as part of the KA09x installation . For automated tests you can use CMWrun or own programs.

3.5.1 CMWrun option KT-110

CMWrun is a ready-to-use automation software for configuring test sequences by remote control for all supported standards in the CMW family. The software engine is based on the execution of test DLLs (plug-in assemblies). This architecture not only allows easy and straightforward configuration of test sequences without knowledge of specific remote programming of the instrument. It also provides full flexibility in configuring parameters and limits of the test items provided in the CMWrun package options for the different standards. At the end of the test, an easy to read test report with limits, test results and verdict is generated and available in several formats, csv, txt, xml and pdf as well.

The option KT110 for CMWrun remote controls the entire setup for ERA-GLONASS as ready-to-go solution for conformance testing in line with the specification CEN/TS 16454:2012 and GOST 33465-2015. It is the right choice for configuring test sequences by remote control, easy handling of different IVS types and receiving the complete past/fail test protocol. The KT110 software allows also the automation and the handling of the tested devices (by AT commands) in an easy and user friendly way.

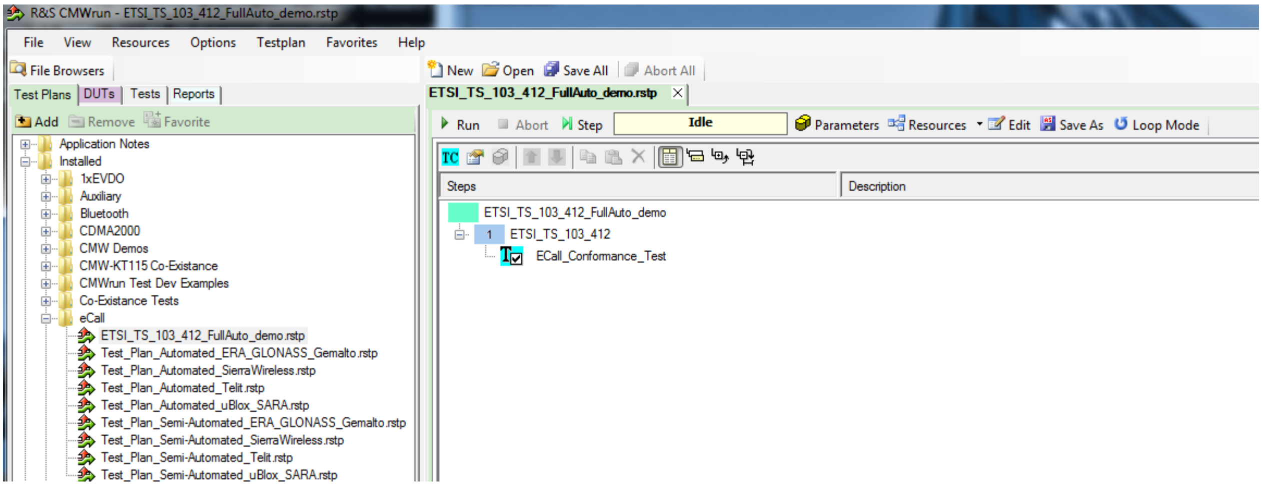


Figure 3-15: The main view of CMWrun. The option KT-110 covers both, eCall and ERA-GLONASS

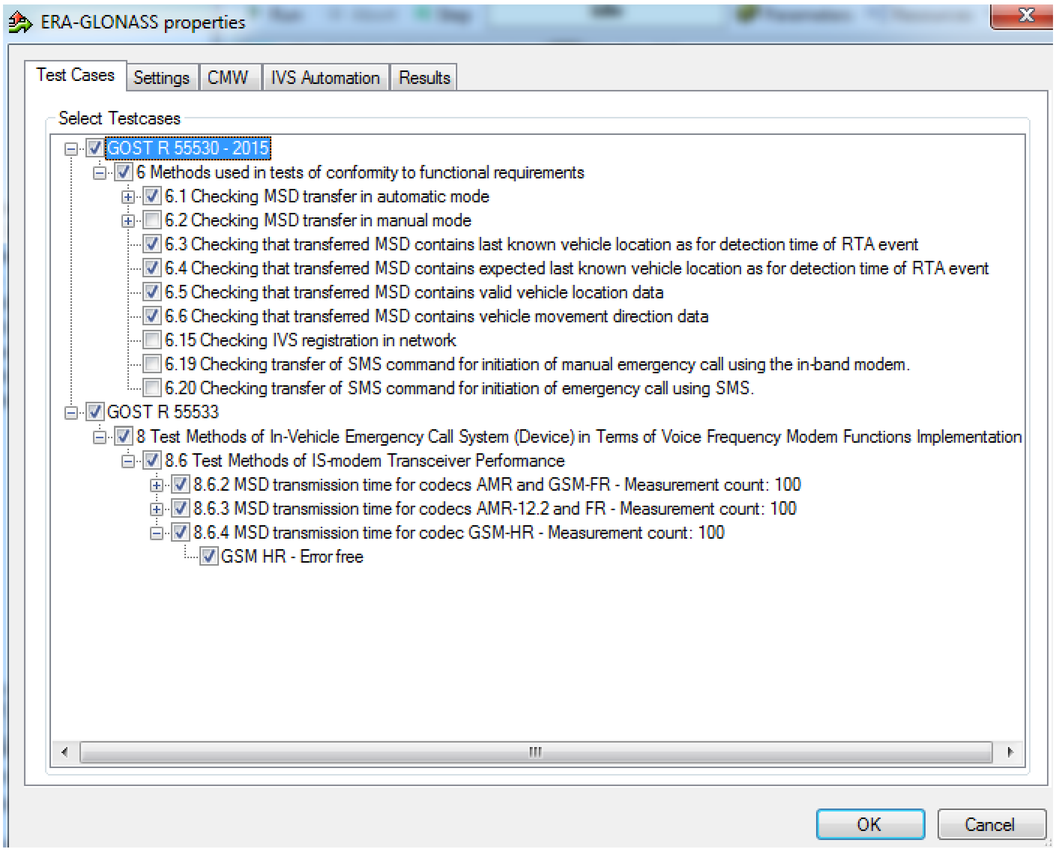


Figure 3-16: Test cases in CMWrun-KT110 according to GOST 55530 and 55533

3.5.2 Remote control of the BASE via SOAP

To use the ERA-GLONASS solution in an automated test system, you have to remote control the KA09x BASE application software. SOAP is an open W3C standard protocol to exchange information and messages. For remote control the base software provides an API. You can find the documentation under `%APPDATA%\Rohde-Schwarz\CMW-KA09x\DOC\CMW-KA09x_API_en.chm`. Thus you can send configuration data (e.g. GSM channel numbers), and receive data (e.g. the transmitted MSD)

Please note that the 'KA09x_Base.exe' with the default uri must be running already. Please also see the manual, chapter 3 [3]

With the installation of the KA09x software, a couple of examples written in C# (.NET) are installed as well under `%APPDATA%\Rohde-Schwarz\CMW-KA09x\Complementary\KA09x_Example1` as a project.

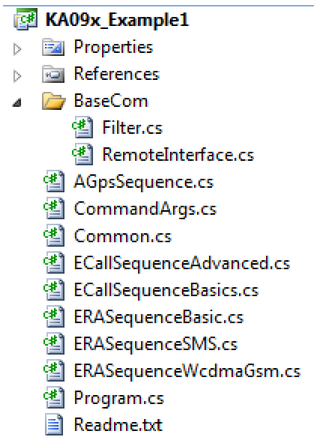


Figure 3-17: The files in the example project.

The project consists of following files:

- **Program**: This is the main program file. It connects to the BASE and then just call one of the example sequences (eCall or ERA, Basic or Advanced).
- **Common**: This class povides the basic settings like the VISA adress for the instruments like CMW or SMBV.
- BaseCom/**RemoteInterface**: This class handles the basic communication (like connection and messages) via SOAP to the Base.
- BaseCom/**Filter**: This class handles the incoming events from the Base to the client. For example you can wait for changes in the state machine of the Base.
- special examples:
  – **ERASequenceBasic**: a basic ERA-GLONASS call (details see below)
  – **ERASequenceSMS**: a ERA-GLONASS SMS example (details see below)
  – **ERASequenceWcdmaGsm**: a ERA-GLONASS example, gets the MSD two times: first via WCDMA, second via GSM
  – **ECallSequenceBasic**: a basic eCall
  – **ECallSequenceAdvanced**: a more advanced eCall

RemoteInterface

This class implements the client interface for the communication with the base. It uses the SOAP implemetation provided by .NET. It handles the incoming events / messages from the base, does the connection / disconnection and sends 'keep alive' to the base. More details are in Chapter 5.2.1, "RemoteInterface", on page 64.

Filter

The class Filter helps to wait for certain incoming events. The events the Base sends to the Client may come asynchronous or in an unexpected order. The class provides a method 'WaitFor' which waits for certain events. This is very helpful for tracking e.g. the state machine of the base software. (see Figure 3-18). For the method calls see the EraSequenceBasic.

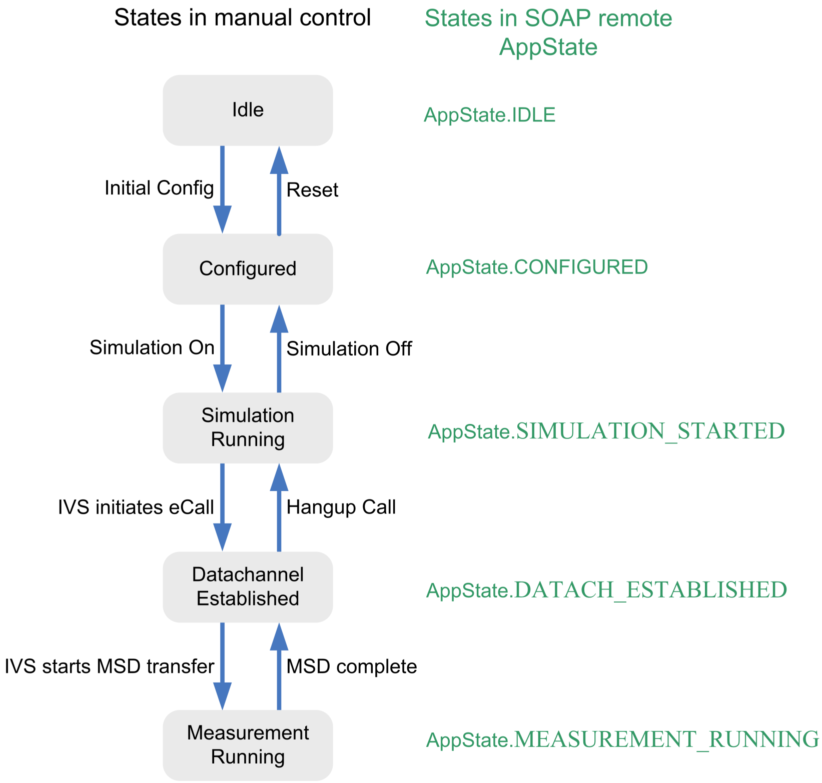


Figure 3-18: The different states of the base. In remote control the appstate can be queried. Every state change creates an event 'NewState' which can be handled in the control software.

Common

The class common provides basic settings of the used instruments common to all examples and which are independent of the used RAN's or GNSS settings, like the VISA address or the used connector.

To talk to the instruments and simulations, certain ID's are defined:

```
// The IDs are used to address the individual systems
//  like RANs, devices, services, etc...
public const UInt16 RAN_SIM_ID = 1;
public const UInt16 RAN_ID = 2;
public const UInt16 PSAP_ID = 3;
public const UInt16 GNSS_SIM_ID = 4;
public const UInt16 GNSS_ID = 5;
```

Some global configuration is done in the class Config:

```
/// Some global configuration concerning CMW and SMBV equipment
/// Could be extended by further parameters concerning IVS or
/// RAN / GNSS configurations
/// </summary>
public static class Config
{
    public static string cmwAddress = "TCPIP::CMW50050-123456::INST0::INSTR";
    public static CmwRfConnector cmwRfConnector = CmwRfConnector.RF1C;
    public static double cmwRfAttenuation = 0.0;
    public static string smbvAddress = "GPIB::28::0::INSTR";
}
```

As an example, the configuration of the CMW is shown. The call is done in the Sequence:

```
public static void useCmw(Filter f, RemoteInterface r)
{
    System.Console.WriteLine("Will send CfgCmw(" + Config.cmwAddress + ", " + Config.cmwRfConnector + ")");
    EventStatus ret = r.toBase.sendEvent(new CfgCmw()
    {
        targetId = RAN_SIM_ID,
        rfConnector = Config.cmwRfConnector,
        visaName = Config.cmwAddress,
        attenuationInputInDb = Config.cmwRfAttenuation,
        attenuationOutputInDb = Config.cmwRfAttenuation,
    });
    CfgResult res = f.waitFor<CfgResult>(TimeOut.CFG_DEVICE);
    throwIfUnsuccessfull(res);
}
```

1. Send event 'CfgCmw' to the Base with attributes ID, Connector and VISA name
2. Wait for result (event = 'CfgResult')

Program

This is the main program. It connects to the Base via SOAP and selects the wanted sequence.

```
class Program
{
    /// <summary>
    /// To use this example a 'KA09x_Base.exe' with the default uri (see below) must be running.
    /// </summary>
    /// <param name="args"></param>
    static void Main(string[] args)
    {
        RemoteInterface r = new RemoteInterface();
        Filter f = new Filter(r);
        Ka09xComponentVersion myVersion = new Ka09xComponentVersion()
        {
            major = 1,
            minor = 0,
            patchLevel = 0
        };
```

First it creates a new object named "r" of type Remoteinterface and a new object named "f" of type filter.

```
try
{
    r = new RemoteInterface();
    Filter f = new Filter(r);

    args = parseCmdArgs(args);
    int choice = chooseSequence(args);

    Console.WriteLine("Will connect now ...");
    r.connect(KA09x_SoapV3.SoapInterfaceConsts.DEFAULT_BASE_SERVICE_URL,
        KA09x_SoapV3.SoapInterfaceConsts.DEFAULT_CLIENT_SERVICE_URL,
        MY_VERSION, "KA09x_Example1");

    Console.WriteLine("Connected, will do sequence {0} ...", choice);
    switch (choice)
    {
        case (1):
            ECallSequenceBasics.doSequence(f, r);
            break;
        case (2):
            ECallSequenceAdvanced.doSequence(f, r); // for a more advanced scenario
            break;
        case (3):
            ERASequenceBasic.doSequence(f, r); // an ERA-GLONASS basic
            break;
        case (4):
            ERASequenceSMS.doSequence(f, r); // an ERA-GLONASS basic
            break;
        case (5):
            ERASequenceWcdmaGsm.doSequence(f, r); // an ERA-GLONASS sequence doing both Gsm and Wcdma
            break;
        default:
            Console.WriteLine("Unknown sequence {0} ...", choice);
            break;
    };
```

It connects to the base via the method 'connect' of "r". The default URL of the base is used. Then it just calls the wanted sequence.

ERASequenceBasic

This sequence executes the ERA basic example. It has following steps:

1. ResetBase
2. InitialConfig
3. DoConfiguration
4. SimOn
5. WaitForCall
6. Waiting for the MSD transfer and show the MSD
7. Waiting for ACK

- Reset of the Base (Call Common.resetBase(f,r);)

```
System.Console.WriteLine("Will reset the base ...");
var ret = r.toBase.sendEvent(new ResetBase());
Common.throwIfUnsuccessfull(ret);
f.waitFor<BaseDidReset>(TimeOut.BASE_RESET);
System.Console.WriteLine("... done base back in idle");
```

1. Send the event 'ResetBase'
2. Wait until the reset is done with a timeout of 10 s (event = 'BaseDidReset')

- The basic configuration is done in 'InitialConfig'

```
// Sets up the most basic configuration
initialConfig(f, r);
```

  – Configuration of the setup (used instruments)

```
// Send the setup. This events tells the base the layout of
// our simulated world.
Common.cfgSetup(f, r, Common.SETUP_RAN_GNSS_PSAP);
```

  Send event 'CfgSetup' to the Base with attributes defined in 'common'

  – Which CMW and SMBV is used (defined in common)

```
// Note: Configure GNSS before RAN
Common.useSmbv(f, r);

// Configure the base to use the CMW, if you want to use the CMU comment the next line
Common.useCmw(f, r);
```

  – Configuration of a GNSS scenario (GLONASS: City Scenario Moscow)

```
// Now configure the GLONASS simulation (again GNSS before RAN)
Console.WriteLine("Will send CfgGlonassCity");
ret = r.toBase.sendEvent(new CfgGlonassCity()
{
    targetId = Common.GNSS_ID,
    location = PredefGnssReceiverCityLocation.MOSCOW,
});
res = f.waitFor<CfgResult>();
Common.throwIfUnsuccessfull(res);
```

  1. Send event 'CfgGlonassCity' to the Base with attributes 'Moscow'
  2. Wait for result (event = 'CfgResult')

  – Configuration of the GSM cell settings

```
// Now configure the GSM cell
Console.WriteLine("Will send CfgGsm");
ret = r.toBase.sendEvent(new CfgGsm()
{
    targetId = Common.RAN_ID,
    band = GsmBand.GSM900,
    bcchNum = 64,
    tchNum = 32,
});
res = f.waitFor<CfgResult>();
Common.throwIfUnsuccessfull(res);
```

  1. Send event 'CfgGSM' to the Base with attributes 'GsmBand and Channels'
  2. Wait for result (event = 'CfgResult')

  – Configuration of the PSAP settings

```
// Now configure the PSAP
Console.WriteLine("Will send CfgPsap");
ret = r.toBase.sendEvent(new CfgEraGlonassPsap()
{
    targetId = Common.PSAP_ID,
    alAck = AlAckBehavior.POSITIVE,
    scenario = PsapModemTestScenario.NORMAL,
    storeRecordedSignal = true,
    playToneAfterMsd = true,
    forceRobustModType = false
});
res = f.waitFor<CfgResult>();
Common.throwIfUnsuccessfull(res);
```

1. Send event 'CfgEraGlonassPsap' to the Base with different attributes
2. Wait for result (event = 'CfgResult')

- Execute all configurations (including a license check and settings of all used instruments) and wait until everything is done (Call Common.DoConfiguration(f,r);)

```
// Transition to configured
Console.WriteLine("Will send DoConfiguration");
var ret = r.toBase.sendEvent(new DoConfiguration());
Common.throwIfUnsuccessfull(ret);
var nState = f.waitFor<NewState>(Common.TimeOut.IDLE_TO_CFG);
if (nState.currentState != AppState.CONFIGURED)
    throw new Exception("DoConfiguration failed!");
```

1. After configuring with InitialConfig, send the event 'DoConfiguration'
2. Wait for a state transition with a timeout (common.TimeOut.IDLE_TO_CFG) (event = 'NewState')
3. Ceck if the current state is 'CONFIGURED' otherwise something went wrong

- Start the simulation (Call Common.simOn(f,r);)

```
// Start the simulation
Console.WriteLine("Will send SimulationOn to start simulation");
var ret = r.toBase.sendEvent(new SimulationOn());
Common.throwIfUnsuccessfull(ret);
var nState = f.waitFor<NewState>(TimeOut.CFG_TO_SIMON);
if (nState.currentState != AppState.SIMULATION_STARTED)
    throw new Exception("SimulationOn failed!");
```

1. Switch on the simulation by the event 'SimulationOn'
2. Wait for a state transition (event = 'NewState', TimeOut.CFG_TO_SIMON)
3. Ceck if the current state is 'SIMULATION_STARTED' otherwise something went wrong

- Wait for the ERA-GLONASS call (Call Common.waitForCall(f,r);)

```
Console.WriteLine("Waiting for call ...");
NewState nState = f.waitFor<NewState>(timeout);
if (nState.currentState != AppState.DATACH_ESTABLISHED)
    throw new Exception("Call somehow failed!");
```

1. No event from the client to send, the call has to be initialized by the IVS
2. Wait for a state transition (event = 'NewState', default timeout)
3. Ceck if the current state is 'DATACH_ESTABLISHED' otherwise something went wrong

- Receive the MSD
  The receiving of the MSD is done in three parts:
  – Wait for the start of the transmission of the MSD

```
// Wait for msd start
Console.WriteLine("Waiting for MSD tranfere ...");
nState = f.waitFor<NewState>(TimeSpan.FromMinutes(1));
if (nState.currentState != AppState.MEASUREMENT_RUNNING)
    throw new Exception("Sending MSD somehow failed!");
```

1. Wait for a state transition (event = 'NewState', default timeout)
2. Ceck if the current state is 'MEASUREMENT_RUNNING' (the MSD transfer is in progress), otherwise something went wrong

– Wait until the whole MSD is successfully received (Call Common.waitForSuccessfullRawMSD(f,r);)

```
KA09x_SoapV3.ECall.RawMsdVoice rawMsd;

do
{
    System.Console.WriteLine("Waiting for raw MSD result ...");
    rawMsd = f.waitFor<KA09x_SoapV3.ECall.RawMsdVoice>(TimeSpan.FromSeconds(10));
    print(rawMsd);

    if (rawMsd.msd != null)
        return rawMsd;

    System.Console.WriteLine("RawMsd: transmission failed");
}
while (rawMsd.redundancyVersion < 8);

throw new Exception("Msd transmission faild after last redundancy version");
```

1. The IVS may transmit the MSD in different redundancy versions at a maximum of eight times. This is handled by the program with the do-while loop.
2. Wait for state 'RawMsdVoice' with a timeout of 10 seconds)
3. If the PSAP is able to receive the MSD successfully, the 'rawMsd.msd' is different from NULL, return to mainprogram, otherwise stay in the loop

– Output the decoded MSD part

```
Console.WriteLine("Waiting for decoded MSD result ...");
DecodedEraGlonassMsdV1 decMsd = f.waitFor<DecodedEraGlonassMsdV1>(); // This expected the IVS to send an
Common.print(decMsd);

// Let's check for additional data
if (decMsd.additionalData.present)
{
    KA09x_SoapV3.EraGlonass.DecodedEraGlonassExtV1 eraExt =
        decMsd.additionalData.optFieldData as KA09x_SoapV3.EraGlonass.DecodedEraGlonassExtV1;
    if (eraExt != null)
    {
        Common.print(eraExt);
    }
    else
    {
        Console.WriteLine("MSD had unknown additional data (" + decMsd.additionalData.optFieldData + ")"
    }
}
```

1. Wait for event 'DecodedEraGlonassMsdV1'
2. Output the decoded MSD (V1 is expected, depends on the IVS)
3. Check for additional data (DecodedEraGlonassExtV1)

- Wait for ACKs

```
// Wait for LL+HL/AL Ack to be fully sent
System.Console.WriteLine("Waiting for LL+AL/HL ...");
nState = f.waitFor<NewState>(TimeSpan.FromSeconds(10));
if (nState.currentState != AppState.DATACH_ESTABLISHED)
    throw new Exception("Call somehow failed!");
```

1. No event from the client to send, the PSAP sends the ACK's to IVS after successful decoding of the MSD.
2. Wait for a state transition (event = 'NewState', timeout 10 seconds)
3. Ceck if the current state is 'DATACH_ESTABLISHED' otherwise something went wrong

ERASequenceSMS

This sequence executes the ERA SMS example with a MSD request.

Please note that the first steps are the same like in the ERASequenceBasic:

- Reset of the Base
- The basic configuration is done in 'InitialConfig'
- Execute all configurations (including a license check and settings of all used instruments) and wait until everything is done
- Start the simulation

The additional steps for the SMS are

- Wait for the registration of the IVS to the network

```
// Please swith on the IVS
System.Console.WriteLine("==========================================================");
System.Console.WriteLine("Please switch on your IVS");
System.Console.WriteLine("==========================================================");

// Wait for register
System.Console.WriteLine("Waiting for registration ...");
do {
     CMWGSMState = f.waitFor<GsmCellSignalStateCmw>(TimeSpan.FromMinutes(1));   // ask for the Status of th

} while (CMWGSMState.currentDetailState != GsmCellSigStateCmw.SYNC);
```

  1. In a do...while loop the status of the Cell is queried (Wait for event 'GsmCellSignalStateCmw')
  2. If the status is 'SYNC', exit the loop
- Request a SMS with a MSD (Call Common.requestMsdViaEgtsEcallMsdReq(f,r);). This contains three parts:
  – Configure the wanted SMS command

```
// Define / configure the SMS command we want to send
KA09x_SoapV3.EraGlonass.EraGlonassSmsCommand eraMsdReq =
    new KA09x_SoapV3.EraGlonass.EraGlonassSmsCommand()
{
    targetId = Common.PSAP_ID,
    // We are requesting an MSD ...
    data = new EGTS_ECALL_MSD_REQ()
    {
        // ... the MSD shall be a new one
        MID = EGTS_ECALL_MSD_REQ.NEW_MESSAGE,
        // ... the MSD shall be reported to the PSAP via SMS
        TRANSPORT = Transport.SMS
    }
};
```

– Send the SMS command from the PSAP to the IVS

```
// Send the SMS command ...
EventStatus ret = r.toBase.sendEvent(eraMsdReq);
Common.throwIfUnsuccessfull(ret);
Console.WriteLine("Waiting for outgoing SMS ...");

// ... and wait until the OutgoingSMS is done
// NOTE this step is not needed and the KA09x reports this outgoing SMS purely for
//   informational purposes. The above command is a reduced / simplified form of
//   the below OutgoingSms which contains the full ERA-GLONASS PDU
OutgoingSms outSms = f.waitFor<OutgoingSms>(Common.TimeOut.SMS_TX);
Console.WriteLine("... sent");
```

– IVS receives SMS command and responds. Wait for the incoming SMS at the PSAP

```
// Next the IVS must respond with the requested MSD via SMS
Console.WriteLine("Waiting for incoming SMS ...");
IncomingSms incSms = null;
try
{
    // Wait for an incoming SMS, may take some time ...
    incSms = f.waitFor<IncomingSms>(Common.TimeOut.SMS_RX);
}
catch (TimeoutException)
{
    // Timeout expired, give IVS a second chance and send SMS command again
    Console.WriteLine("Timeout while waiting for incoming SMS, will send SMS request once more ...");
    r.toBase.sendEvent(eraMsdReq);
    incSms = f.waitFor<IncomingSms>(Common.TimeOut.SMS_RX);
}
```

1. Wait for an incoming SMS (Wait for event 'IncomingSms' for a certain time)
2. if the timeout expires, the PSAP may have to send the SMS command again and again wait for an incoming SMS (Wait for event 'IncomingSms' for a certain time)

- Show the Incoming SMS (raw SMS)

```
// show the incoming SMS (raw data)
 System.Console.WriteLine("IncomingSMS: Information={0} Status ='{1}'",
                            IncSMS.decodingInformation,
                            IncSMS.decodingStatus);
```

- Check and decode the SMS (Call Common.extractRawMsd(inSms);)

```
if (incSms == null)
    return null;

// check the decoding: it must be fully successful or partly successful
if ((incSms.decodingStatus != DecodingInfo.Successful) &&
        (incSms.decodingStatus != DecodingInfo.WithProblem))
    return null;

// see also Table 3 and 14 of GOST 54619
// check if the Service Frame Data (SFRD) is present
if (incSms.pdu.SFRD.present == false)
    return null;


EGTS_PT_APPDATA appData = incSms.pdu.SFRD.optFieldData as EGTS_PT_APPDATA;

// check if the Service Data RECORD (SDR) is present
if (appData == null || appData.SDR.present == false)
    return null;

// check if there are data ind Record Data (RD)
if (appData.SDR.optFieldData[0].RD.Length == 0)
    return null;

// we beleive that the MSD is in RD (see table 14 of GOST 54619)
EGTS_SR_RAW_MSD_DATA subrecordRawMSD =
    appData.SDR.optFieldData[0].RD[0] as EGTS_SR_RAW_MSD_DATA;

// finally check if MSD was actually in the RD and if so return raw data
if (subrecordRawMSD == null || subrecordRawMSD.MSD.present == false)
    return null;
```

1. The decoding status can be 'successful' or 'with problems', both are OK ('with problems' means OK, but CRC not correct)
2. ckeck if the SFRD part is present (this part is optional, see table 3 of [5])
3. check if the SDR part is present. It may consist of several SDR's (SDR's are optional, see table 5 of [5])
4. if a SDR exists, it shall contain (mandatory) one or more RD's
5. If an RD exists, it contains the data part, we expect the MSD in the first one (RD[0])

- Wait for event 'DecodedEraGlonassMsdV1' and show it

```
if( rawMsd != null )
{
    // So we actually got an MSD in this SMS. That means the Base will send us a decoded
    // version of it next. The following waitFor expects the above SMS to contain an MSD
    // in version 1, note however that this solely depends on the IVSs choice.
    Console.WriteLine("Waiting for decoded MSD result ...");
    DecodedEraGlonassMsdV1 decMsd = f.waitFor<DecodedEraGlonassMsdV1>();
    Common.print(decMsd);
}
```

Output the decoded MSD (V1 is expected, depends on the IVS). The data is stored in 'decMsd'. As an example the position (Longitude and Latitude) is shown

4 GNSS Performance Testing

The standard GOST-33471-2015 (55534) [6] defines test methods for the navigation module of the ERA IVS.

The required tests include:

- Position accuracy
- Tracking sensitivity
- Acquisition accuracy
- Time to first fix (TTFF)

The solution presented here is intended for tests at the module level and at the device level. This solution features the software CMWrun for a fully automatic test configuration, scheduling, DUT configuration, data analysis and test report generation. The option SMBV-K360 for CMWrun controls remotely the SMBV and permits all basic parameters settings, such as vehicle simulation, GNSS simulation or level, to be managed directly. As a result, measurements are fully reproducible. The solution covers test cases TC 5.1 to TC 5.15 of the specification.

4.1 Test Setup

Figure 4-1 shows the test setup.

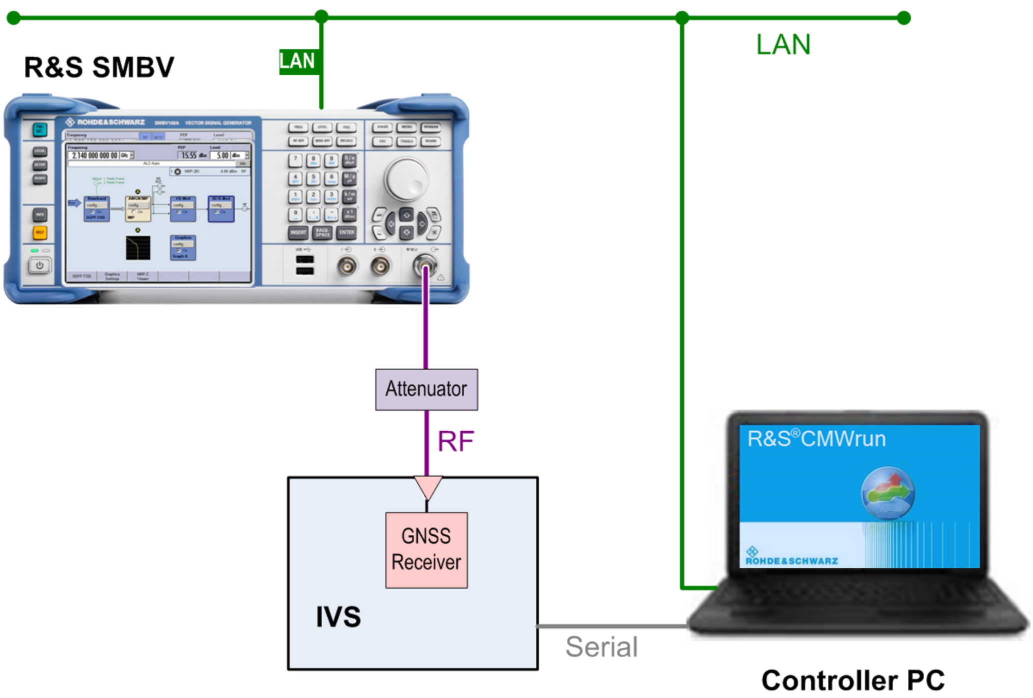


Figure 4-1: Test setup for the GNSS performance test

The SMBV simulates GNSS signals, such as GLONASS and or GPS, that are used by the IVS for positioning. The CMWrun software installed on an external PC sets necessary parameters for ERA-GLONASS tests on the SMBV (which needs the SMBV-K360) and fully remote controls the SMBV (via LAN). CMWrun also controls the IVS via serial interface. It is not necessary here to delve into the operation of the SMBV. For more details on the SMBV see Chapter 3.2.2, "SMBV Vector Signal Generator", on page 17.

Needed GNSS options

Following software options of the SMBV are needed for ERA-GLONASS performance tests:

Table 4-1: Options and test cases

Software configuration	Option SMBV	Comments
Minimum requirements		
GPS	K44	
GLONASS	K94	
GNSS enhanced	K92	
Extension to 12 satellite	K91	
Extension to 24 satellite	K96	
To add for full coverage		
SBAS	K110	for TC 5.5
Antenna pattern	K102	for TC 5.8
Test automation		
ERA -Glonass test suite	K360	with CMWrun

4.2 Performance Tests

The Test Suite SMBV-K360 together with CMWrun supports following test according to GOST-55534 / 33471:

Table 4-2: Covered test cases

TC	Names according to GOST	Name in the Test Suite
5.1	Checking that navigation signals of standard precision in L1 band of GLONASS GNSS may be received and processed in order to determine vehicle coordinates and its velocity components	Availability of position/velocity for GLONASS L1
5.2	Checking that navigation signals of standard precision in L1 band of GPS GNSS may be received and processed in order to determine vehicle coordinates and velocity components	Availability of position/velocity for GPS L1

TC	Names according to GOST	Name in the Test Suite
5.3	Checking that navigation signals of standard precision in L1 band of GLONASS and GPS GNSS may be received and processed in order to determine vehicle coordinates and velocity components	Availability of position/velocity for combined GPS/GLONASS L1 processing
5.4	Checking that navigation parameters data may be output to external devices in NMEA-0183 format	Verify NMEA transmission from DUT
5.5	Checking receiver autonomous integrity monitoring (RAIM) algorithm	Functional RAIM test
5.6	Checking that navigation parameters may be determined in PZ-90 and WGS-84 coordinate systems	Use of different reference systems (PZ-90/WGS-84)
5.7	Estimating error in evaluation of plane view coordinates and altitude in autonomous static mode	Location accuracy (static receiver)
5.8	Estimating error in evaluation of plane view coordinates, altitude and velocity in dynamic mode	Location accuracy (moving receiver)
5.9	Checking minimum time interval of observation data updates	Minimum update rate of NMEA stream
5.10	Checking time to restore tracking signals of working NSC constellation after tracking fault caused by blockage	Reacquisition time
5.11	Evaluating time of navigation fix in cold start mode	Time-to-first fix (TTFF) under cold start conditions
5.12	Evaluating sensitivity of GNSS navigation module in GNSS signal search (locking) mode and in GNSS signal hold (tracking) mode	Tracking and acquisition sensitivity
5.13	Checking that data output rate may be changed in required range using GNSS_DATA_RATE setup parameter	Change update rate of NMEA stream
5.14	Checking minimum elevation (cut-off angle) for navigation spacecrafts	Check cutoff angle settings for navigation satellites
5.15	Checking power-off time of navigation module (GNSS navigation receiver) after ignition is turned off	Check power-off time of navigation module (GNSS navigation receiver)

Please see [7] for the start of the test suite and the general handling of CMWrun.

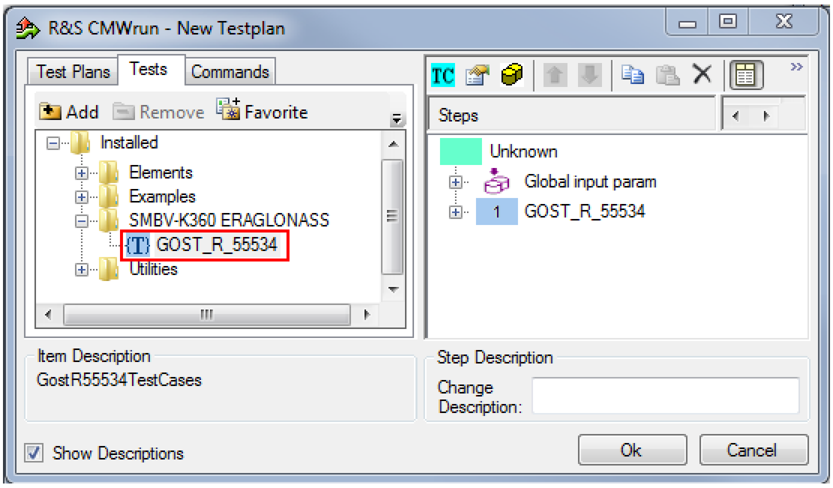


Figure 4-2: CMWrun with the test suite GOST 55534

The settings are available in the properties (Figure 4-3).

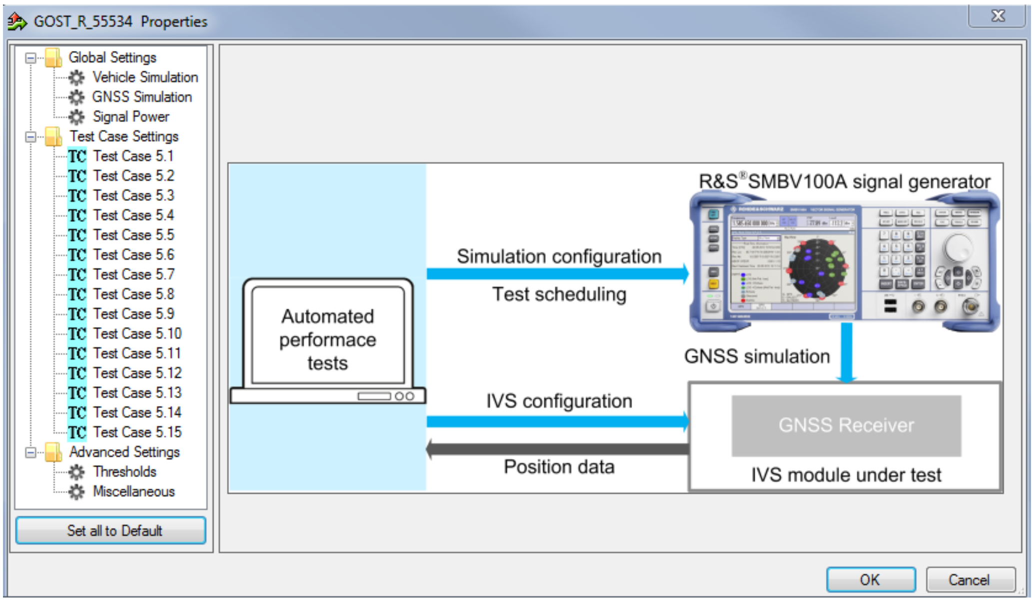


Figure 4-3: Properties of the test suite

Set global settings in the three sections "Vehicle Simulation", GNSS Simulation and "Signal Power". The special, test-dependent settings are under "Test Case 5.x". Figure 4-4 shows an example (TC5.1).

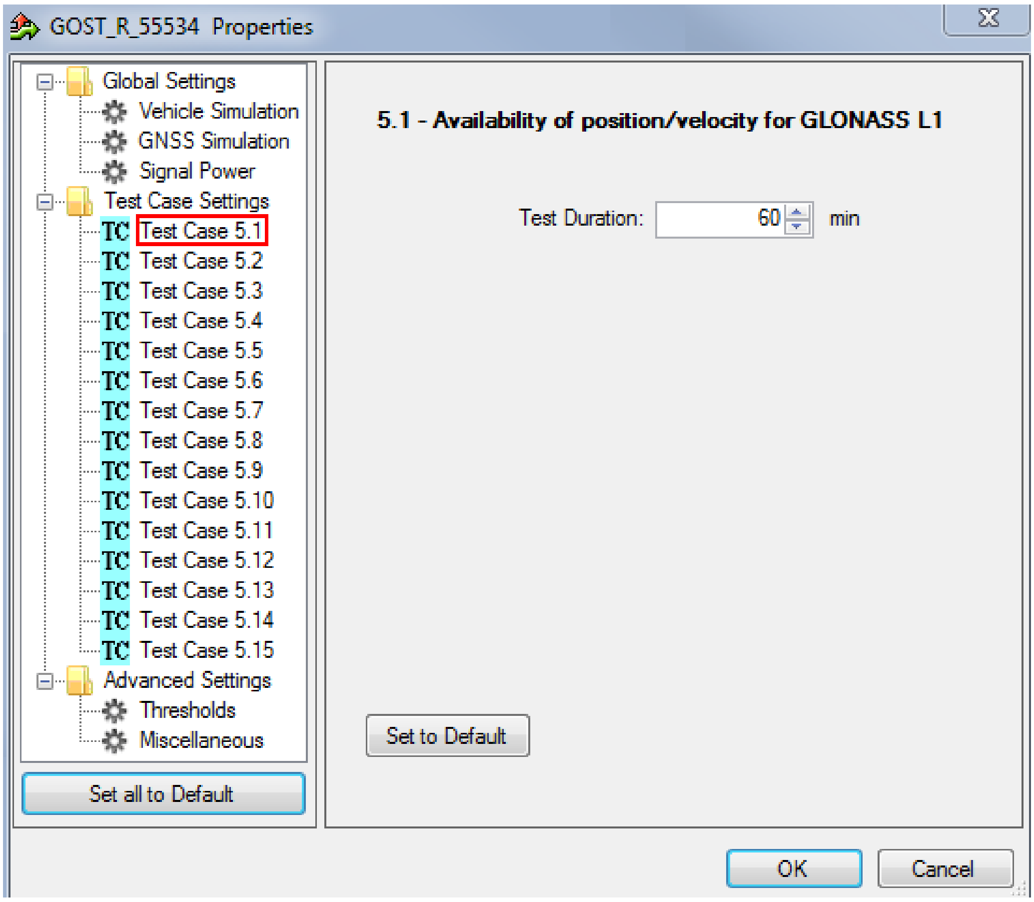


Figure 4-4: Example for the configuration of a test case (TC5.1)

CMWrun shows measurement reports, which also can be saved and exported. Figure 4-5 shows an example.

GOST_R_55534: 5.1 - Availability of position/velocity for GLONASS L1

Test Items and Conditions	DUT		Result	Unit	Status
Plane peak error *GLONASS*	ublox EVK-M8N		7.03	m	
Height peak error *GLONASS*	ublox EVK-M8N		9.72	m	
Velocity peak error *GLONASS*	ublox EVK-M8N		5.69	m/s	
Test result					Passed

Figure 4-5: Example for a report

5 Appendix

5.1 Glossary

AL-ACK	ACKnowledgment on the Application Layer
BPPM	Bipolar Pulse Position Modulation
CRC	Cyclic Redundancy Check
DL	Downlink
ERA	Emergency Road Assistance
GNSS	Global Navigation Satellite System
GLONASS	Globalnaya navigatsionnaya sputnikovaya sistema
GPS	Global Positioning System
GSM	Global System for Mobile Communications
IVS	In-Vehicle System
LL-ACK	ACKnowledgment on the Link Layer
MSD	Minimum Set of Data
PSAP	Public Safety Answering Point
RV	Redundancy Version
UL	Uplink
WCDMA	Wideband Code Division Multiple Access

5.2 The implementation of a SOAP client

The remote control examples described in Chapter 3.5.2, "Remote control of the BASE via SOAP", on page 48 use a SOAP client provided with .NET. It is implemented in the class RemoteInterface. The class Filter simplifies the handling of incoming events. Both are detailed in this chapter.

5.2.1 RemoteInterface

This class implements the client interface for the communication with the base. It handles the incoming events / messages from the base and does the connection / disconnection.

Main members are derived from the SOAP server provided by Windows Communication Foundation (WCF):

```
// Basically that is our soap server for callbacks from base
private ServiceHost host = null;

// Proxy factory and proxy object to send events to base
//  See for details
//   http://msdn.microsoft.com/en-us/library/bb412169.aspx
private ChannelFactory<IToBase34> eventSenderFactory = null;
private IToBase34 eventSender = null;
```

Figure 5-1: 'host' is the the main member of the SOAP implementation via DOT.NET. eventSender of type 'IToBase32' handles the events to send to the base

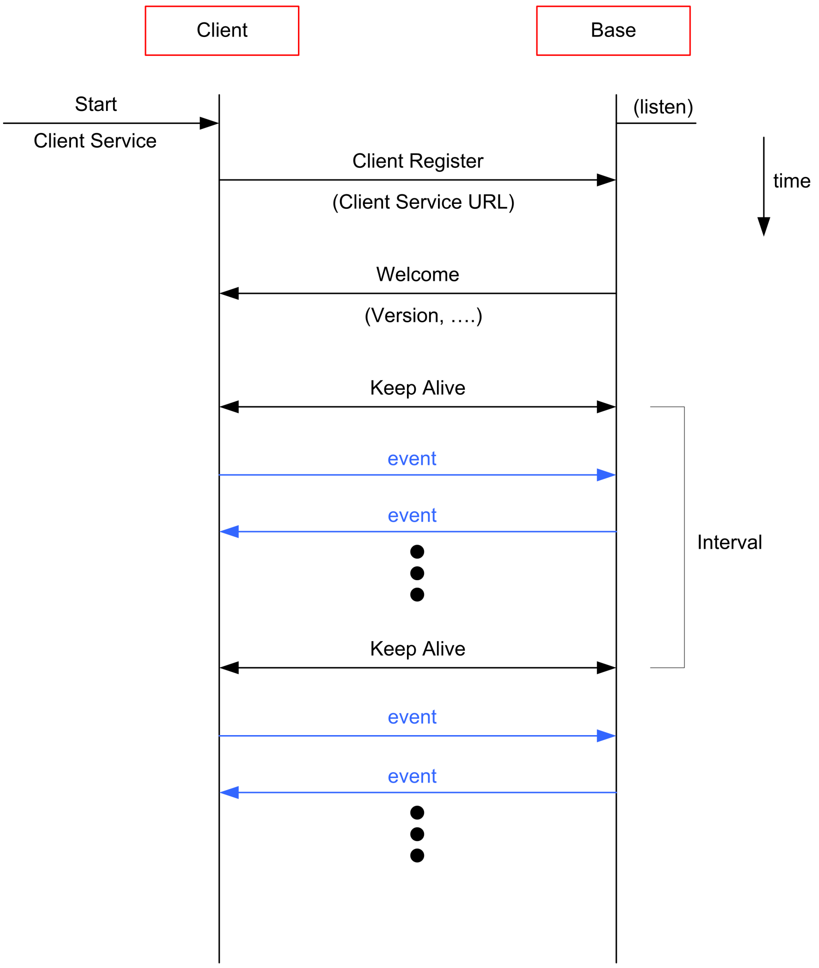


Figure 5-2: Basic communications in SOAP

1. The Base is in idle status and listens to incoming events
2. First thew client side starts its service ('StartClientService')
3. The client has to register at the Base with his own URL
4. The base answers with a 'Welcome'
5. Both, base and client, have to send 'KeepAlive' in certain intervals, otherwise the base severs the connection

6. Now both parts can send events

The first four basic steps (steps 2...5) are handled in the function 'Connect' of the RemoteInterface:

```
public void connect(string baseServiceUrl, string myServiceUrl, Ka09xComponentVersion myVersion, string myName)
{
    if (connected)
        throw new InvalidOperationException("We are alreday connected");

    // First clear the incomming event queue
    clearIncommingEventQueues();

    // Open our client service for callbacks / messages / events from base
    openMySoapServer(myServiceUrl);

    // Build the proxy class for calls / messages / events to the base soap server
    buildToBaseSoapProxy(baseServiceUrl);

    // Finally register with the base and tell it to connect to our soap server
    registerWithBase(myServiceUrl, myVersion, myName);

    // We are connected, but to stay connected, we need to send keepalives
    startKeepAlive();
}
```

Figure 5-3: The method 'Connect'

- First the queue for incoming events is cleared ('clearIncomingEventQueue'), so the client starts with an empty queue
- The method 'openMySoapServer' starts the client service (step 2)
- The method 'buildToBaseSoapProxy' opens a channel to communicate to the base
- The registering at the base is done via 'registerWithBase' (step 3). It automatically waits for the 'Welcome' message sent by the base (step 4)
- After the 'Welcome' message, the clients starts to transmit 'KeepAlive' messages (method 'startKeepAlive'). This is done automatically in certain intervals in a different thread.

```
void startKeepAlive()
{
    lock (keepAliveTimerLock)
    {
        Console.WriteLine("Will start sending keepalive");
        if (keepAliveTimer != null)
            return;

        keepAliveTimer = new System.Threading.Timer(sendKeepalive, null, 0,
                KA09x_SoapV3.SoapInterfaceConsts.KEEP_ALIVE_INTERVAL_IN_MS);
    }
}
```

Events sent by the base are handled with the method 'handleIncomingEvent':

```
EventStatus handleIncommingEvent(EventBase eventData)
{
    try
    {
        Console.Write("Rec: " + eventData.GetType() + " ");
        // Every event we receive, we just put into the queue
        incommingEventQueue.Add(eventData);
        Console.WriteLine("Ok");

        // We just always say everything was ok
        //   the base is not really interested what we do with events;
        //   it's enough to say we received it
        return okEventStatus();
    }
    catch(Exception e)
    {
        Console.WriteLine("Err '{0}'", e.Message);
        return new EventStatus()
        {
            successfullyDelivered = false,
            description = e.Message
        };
    }
}
```

Figure 5-4: Incoming events are just add to the queue

All incoming events are just add to the queue, thus they are serialized. They can be handeld (queried) is a simple way with the class 'Filter'

5.2.2 Filter

The class Filter helps to wait for certain incoming events. The events the Base sends to the Client may come asynchronous or in an unexpected order. The class provides a method 'WaitFor' which waits for certain events. This is very helpful for tracking e.g. the state machine of the base software.

The member 'backlog' is a list of events waiting to be preocessed:

```
const int MAX_BACKLOG_SIZE = 100;

List<EventBase> backlog = new List<EventBase>();
```

Figure 5-5: A list of events called backlog

The method 'waitFor' first checks the 'backlog' for the wanted event. If it is not in the list up to now, it waits until it is received.

```
public EventBase waitFor(Type eventTypeToWaitFor, TimeSpan timeout)
{
    // first check the backlog for the wanted event
    for (int i = 0; i < backlog.Count; ++i)
    {
        EventBase oldEvent = backlog.ElementAt(i);
        if ( eventTypeToWaitFor.IsAssignableFrom( oldEvent.GetType()))
        {
            // that was the event we were waiting for
            backlog.RemoveAt(i);
            Console.WriteLine("waitFor(" + eventTypeToWaitFor + ") -> got it from backlog");
            return oldEvent;
        }
    }

    // The event we are waiting for, is not in the backlog,
    //  so we have to block on the remote interface with
    //  remote.getNextEvent() untill we received it.
    while (true)
    {
        Console.WriteLine("waitFor(" + eventTypeToWaitFor + ") ...");
        EventBase newEvent = remote.receiveNextEvent(timeout); // very long timeout ...
        if (eventTypeToWaitFor.IsAssignableFrom( newEvent.GetType() ))
        {
            // that was the event we were waiting for
            return newEvent;
        }

        // we got an event, but it was not the type we are waiting for ...
        // ... so put into backlog for later processing
        backlog.Add(newEvent);
        if (backlog.Count > MAX_BACKLOG_SIZE)
        {
            // backlog too long, remove oldest entry
            Console.WriteLine("Backlog full, will discard " + backlog.ElementAt(0).GetType().Name);
            backlog.RemoveAt(0);
        }
    }
}
```

Figure 5-6: The method 'waitFor'

To provide a more convenient way, a generic method is also provided as a wrapper:

```
public EVENT_TYPE waitFor<EVENT_TYPE>(TimeSpan timeout)
    where EVENT_TYPE: EventBase
{
    return (EVENT_TYPE)waitFor(typeof(EVENT_TYPE), timeout);
}

/// <see cref="EVENT_TYPE waitFor<EVENT_TYPE>(TimeSpan timeout)"/>
public EVENT_TYPE waitFor<EVENT_TYPE>()
    where EVENT_TYPE : EventBase
{
    return (EVENT_TYPE)waitFor(typeof(EVENT_TYPE), DEFAULT_TIMEOUT);
}
```

Figure 5-7: The generic wrapper 'waitFor'

5.3 References

[1] CEN: CEN/TS 16454:2015: **Intelligent transport systems - ESafety - ECall end to end conformance testing**, September 2015

[2] 3GPP: TS26.267 :**Technical Specification Group Services and System Aspects; eCall Data Transfer; In-band modem solution**; General description V12.0.0., December 2012

[3] User Manual: **Test Software for eCall**, CMW-KA094,Rohde & Schwarz, 2015

[4] GOST R 55530-2013: **Road Accident Emergency Response System: Functional test methods of IVS and data transfer protocols**

[5] GOST R 54619-2011: **Road Accident Emergency Response System: Protocol of Data Transmission from In-Vehicle Emergency Call System/Device to Emergency Response System Infrastructure**

[6] GOST R 55534-2013: **Road Accident Emergency Response System: Test methods for navigation modules of in-vehicle emergency call systems**

[7] User Manual: **ERA-GLONASS Test Suite**, SMBV-K360,Rohde & Schwarz, 2017

5.4 Additional Information

Please send your comments and suggestions regarding this white paper to

TM-Applications@rohde-schwarz.com

5.5 Ordering Information

Please contact your local Rohde & Schwarz sales office for further assistance.

CMW500 Radio Communication Tester		
Base Unit	CMW500	
CMW500 Mainframe 03	CMW-PS503	1208.7154.02
Front Panel with Display H600B	CMW-S600B	1201.0102.03
BB Flexible Link H550B	CMW-S550B	1202.4801.03
RF Frontend (Basic) H590A	CMW-S590A	1202.5108.02
Signaling Unit Universal B200A	CMW-B200A	1202.6104.02
GSM Signaling option	CMW-B210A	1202.6204.02
Signaling Unit Wideband, B300B	CMW-B300B	1202.6304.03
Audio Analyzer/Generator H400B	CMW-B400B	1207.8457.02
Speech codec H405A	CMW-B405A	1207.8257.02
GSM GPRS EDGE Rel. 6, Basic signaling	CMW-KS200	1203.0600.02
GSM GPRS EDGE R6, advanced signaling	CMW-KS210	1203.9759.02

CMW500 Radio Communication Tester		
WCDMA R99, basic signaling	CMW-KS400	1203.0751.02
WCDMA R99, advanced signaling	CMW-KS410	1203.9807.02
Hardware set for eCall applications (acc.)	CMW-Z94	1208.6906.02

CMW500 Software Licensing		
CMWPC Smartcard	CMWPC	1201.0002K90
USB Smartcard for CMWPC	CMW-S089A	1202.7900.02
eCall test software	CMW-KA094	1208.4703.02
ERA-GLONASS test software	CMW-KA095	1208.8844.02

CMWrun		
CMWrun eCall	CMW-KT110	1208.7431.02

SMBV100A Vector Signal Generator		
Base Unit	SMBV100A	1407.6004.02
Baseband Generator	SMBV-B10	1407.8607.02
Hard Disk (removable)	SMBV-B92	1407.9403.02
RF Path	SMBV-B103	1407.9603.02
GPS	SMBV-K44	1415.8060.02
Assisted GPS	SMBV-K65	1415.8560.02
GLONASS	SMBV-K94	1415.8677.02
Assisted GLONASS	SMBV-K95	1419.2521.02
Galileo	SMBV-K66	1415.8590.02
Beidou	SMBV-K107	1419.2709.02
QZSS	SMBV-K105	1419.2350.02
GNSS Enhanced	SMBV-K92	1415.8583.02
GNSS Extension to 12 Satellites	SMBV-K91	1415.8577.02
GNSS Extension to 24 Satellites	SMBV-K96	1415.8790.02
ERA-GLONASS Test Suite	SMBV-K360	1419.1890.02

6 Rohde & Schwarz

The Rohde & Schwarz electronics group offers innovative solutions in the following business fields: test and measurement, broadcast and media, secure communications, cybersecurity, monitoring and network testing. Founded more than 80 years ago, the independent company has an extensive sales and service network with locations in more than 70 countries.

The electronics group ranks among the world market leaders in its established business fields. The company is headquartered in Munich, Germany. It also has regional headquarters in Singapore and Columbia, Maryland, USA, to manage its operations in these regions.

Sustainable product design

- Environmental compatibility and eco-footprint
- Energy efficiency and low emissions
- Longevity and optimized total cost of ownership

Certified Quality Management
ISO 9001

Certified Environmental Management
ISO 14001

Contact us

- Europe, Africa, Middle East | customersupport@rohde-schwarz.com
  +49 89 4129 12345
- North America | customer.support@rsa.rohde-schwarz.com
  1-888-TEST-RSA (1-888-837-8772)
- Latin America | customersupport.la@rohde-schwarz.com
  +1-410-910-7988
- Asia Pacific | customersupport.asia@rohde-schwarz.com
  +65 65 13 04 88
- China | customersupport.china@rohde-schwarz.com
  +86-800-810-8228 / +86-400-650-5896

Rohde & Schwarz GmbH & Co. KG

Mühldorfstraße 15 | D - 81671 München

+ 49 89 4129 - 0 | Fax + 49 89 4129 – 13777

www.rohde-schwarz.com

This application note and the supplied programs may only be used subject to observance of the conditions of use set forth in the download area of the Rohde & Schwarz website.

R&S® is a registered trademark of Rohde & Schwarz GmbH & Co. KG. Trade names are trademarks of their owners.