

# NEXT GENERATION ECALL CONFORMANCE TESTING

## Products:

- ▶ R&S®CMW500
- ▶ R&S®CMW-KA096
- ▶ R&S®SMBV100B
- ▶ R&S®SMW200A



Christian Wicke, Bernhard Schulz, Fabian Bette | GFM312 | Version 1e | 12.2020

<https://www.rohde-schwarz.com/appnote/GFM312>

**ROHDE & SCHWARZ**

Make ideas real



# Contents

<b>1</b>	<b>Overview</b> .....	<b>3</b>
<b>2</b>	<b>Introduction</b> .....	<b>3</b>
<b>3</b>	<b>What is next generation eCall?</b> .....	<b>5</b>
3.1	Excursus: Enabling voice in LTE: IP multimedia subsystem (IMS) .....	5
3.2	eCall and NG eCall: System and concepts .....	7
3.3	Sequences .....	9
3.4	Standards.....	10
<b>4</b>	<b>Conformance testing of IVS</b> .....	<b>10</b>
4.1	Why test in the lab?.....	10
4.2	Test setup .....	11
4.2.1	CMW radio communication tester .....	12
4.2.2	SMBV100B vector signal generator .....	13
4.2.3	KA096 PSAP simulator software (NG eCall test software).....	14
4.3	The first NG eCall measurement: Basic operation .....	15
4.3.1	Preparing the test setup .....	15
4.3.2	SIM card (UICC).....	15
4.3.3	Starting and configuring the NG eCall test software (KA096).....	16
4.3.4	Performing an NG eCall .....	22
4.3.5	Troubleshooting .....	26
4.4	Measurements in line with specification CEN TS17240 .....	27
4.5	Automated tests: remote control .....	33
4.5.1	CMWrun option KT-111.....	34
4.5.2	Remote control of the BASE via SOAP .....	36
<b>5</b>	<b>Literature</b> .....	<b>41</b>
<b>6</b>	<b>Ordering Information</b> .....	<b>42</b>
<b>7</b>	<b>Appendix</b> .....	<b>45</b>
A	The implementation of a SOAP client.....	45
A.1	RemoteInterface.....	45
A.2	Filter .....	48
B	Glossary.....	50

# 1 Overview

Next generation emergency call (NG eCall) is an extension of eCall, a service provided in Europe with the goal of reducing response times for accidents or other emergencies on the roadways. This application note briefly describes the technology behind NG eCall and presents conformance tests for NG eCall using the R&S®CMW500 RF tester and the R&S®SMBV100B or R&S®SMW200A vector signal generator. A test software for NG eCall makes it quick and easy to perform these tests with the LTE wireless communications standard.

## Note

The eCall conformance and performance testing is described in application note [1MA241](#).

The ERA-GLONASS conformance and performance testing is described in application note [1MA251](#).

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

# 2 Introduction

Emergency call (eCall) is a service provided in Europe with the goal of reducing response times for accidents or other emergencies on the roadways. It is required by law in new cars as of March 2018. ECall uses GSM and UMTS networks, but operators in Europe have already announced they will phase out support of the GSM and UMTS networks over the next decade. Next Generation eCall (NG eCall) extends the existing eCall for future proof IP-based calls. It uses Long Term Evolution (LTE) networks in addition to circuit-switched networks GSM and UMTS.



Although eCall 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 LTE IP-based call to the public safety answering point (PSAP).

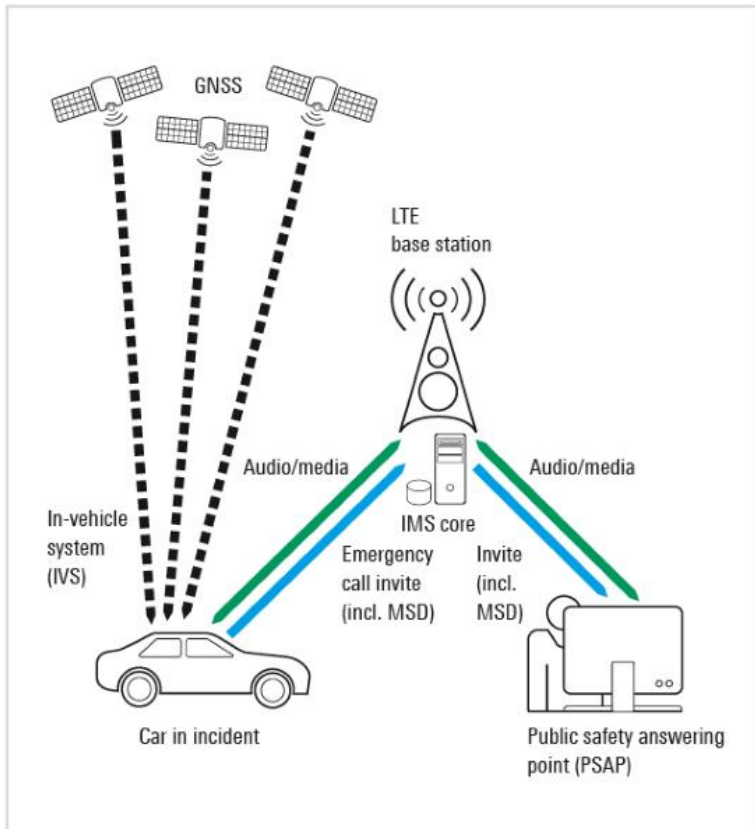


Figure 1: NG eCall system

NG eCall system: In an emergency situation, the car makes an IP-based LTE 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.

This application note describes the principles behind NG eCall and explains conformance testing of the IVS using the test solution offered by Rohde & Schwarz.



Figure 2: Overview - The test solution with GNSS simulator, wireless radio communication tester and an IVS (DUT)

## IVS conformance tests

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 NG eCall via LTE
- ▶ 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 the global navigation satellite system (GNSS) is not necessary
- ▶ Deep familiarity with the operation of the test instrument is likewise unnecessary
- ▶ All relevant LTE cellular network parameters can be modified
- ▶ This allows reproducible measurement results
- ▶ A true emergency call using the emergency number 112 is possible

Please note that the **GNSS performance test for eCall is covered in [1MA241](#)**.

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®SMBV100B vector signal generator is referred to as the SMBV

## 3 What is next generation eCall?

### 3.1 Excursus: Enabling voice in LTE: IP multimedia subsystem (IMS)

Networks like GSM (2G) and UMTS (3G) provide circuit switched services for voice connections. In contrast, networks like LTE (4G) and the upcoming 5G NR are all IP networks. That means no circuit-switch mechanisms are implemented. To support services like voice (in this context necessary for the MSD coded as voice), 3GPP has developed an architecture called IP multimedia subsystem (IMS). It provides functions for multimedia services independent of the radio access technology. For detailed information on voice and SMS in LTE see the white paper [1].

The IMS framework is very complex. For a basic understanding of the concept of voice in LTE, Figure 3 shows a simple architecture with all entities that are required for voice services (and so for NG eCall).

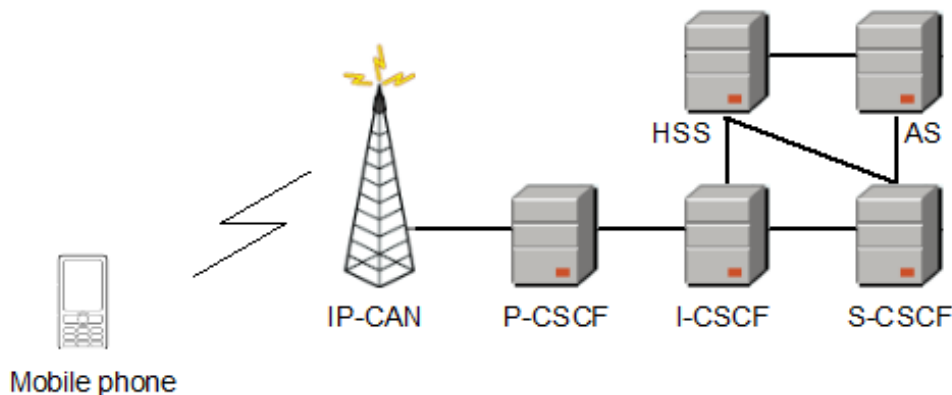


Figure 3: Schematic view of a part of the IMS architecture relevant for voice [1]

- ▶ IP-CAN
  - IP connectivity access network (here: LTE)
- ▶ Call session control functions (CSCF)
  - Proxy (P-CSCF): The first contact for a user
  - Interrogating (I-CSCF): The entry contact within a network for all connections destined to a subscriber
  - Serving (S-CSCF): Responsible for handling the registration process, making routing decisions, maintaining sessions and downloading user information and service profiles from the HSS
- ▶ Home subscriber server (HSS): The master database for a user which contains the subscription related information
- ▶ AS application server: Provides the specific IP application, here voice service

### Protocols

To support different multimedia services, the IMS uses a set of internet-based protocols (they are not part of the IMS) like:

- ▶ The session initiation protocol (SIP) creates, modifies and terminates multimedia sessions
- ▶ The session description protocol (SDP) describes a multimedia session with the help of parameters like media type, codec type, bandwidth, IP address and ports
- ▶ The real-time transport protocol (RTP) and RTP control protocol (RTCP) for transport of real-time applications (e.g. audio).

The transport layer uses UDP and/or TCP.

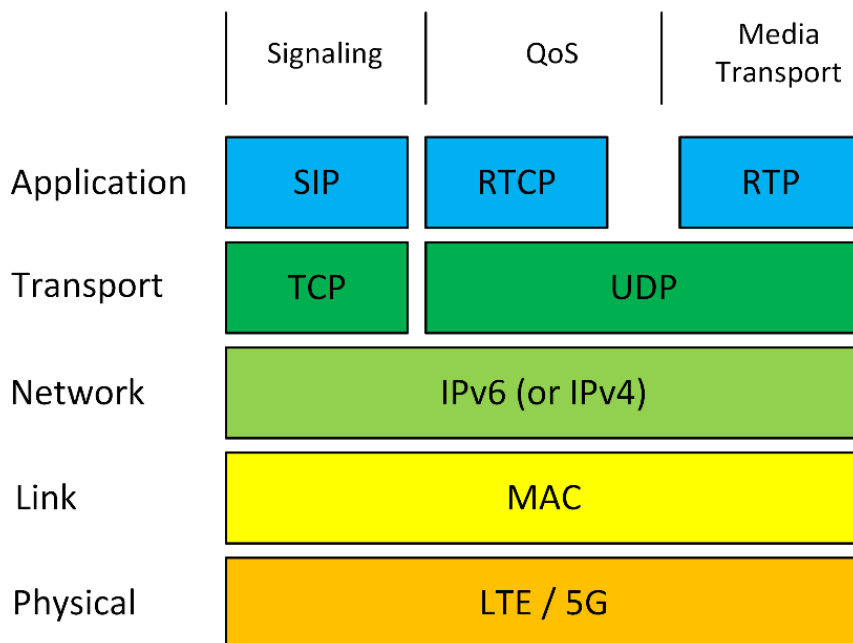


Figure 4: Layer model for NG eCall using IMS

### 3.2 eCall and NG eCall: System and concepts

Emergency call (eCall) is a service provided in Europe with the goal of reducing response times for accidents or other emergencies on the roadways. It is required by law in new cars as of March 2018. Beyond the EU, eCall is also to be introduced in Iceland, Norway and Switzerland. NG eCall provides LTE support for eCall. NG eCall is part of LTE release 14.

Although eCall cannot 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, eCall can also be used to place a call manually (e.g. in another type of emergency).

See Figure 1 for an overview.

An in-vehicle system (IVS) is integrated as a key element into every automobile (Figure 5). The primary components of the IVS are a GNSS receiver (typically GPS) to determine the current position and a cellular module (GSM or WCDMA for legacy eCall and LTE or 5G NR for NG eCall) to permit transmission of the minimum set of data (MSD) via a cellular network to a public safety answering point (PSAP).

Table 1: eCall and NG eCall

	eCall	NG eCall
<b>Radio access network</b>	GSM (2G) / UMTS (3G)	LTE (4G) / 5G NR
<b>Call</b>	Voice call (circuit-switched)	Voice via IP connection (VoLTE) (packet-switched)
<b>Specials</b>	Inband modem required in IVS	Uses IP multimedia subsystem (IMS) of LTE
<b>MSD</b>	Transmission after call setup	Transmission with call setup (SIP invite)
<b>Additional data</b>	No, only 140 bytes in MSD	Extensible, e.g.: <ul style="list-style-type: none"> <li>▶ more data</li> <li>▶ videos, etc.</li> </ul>

NG eCall uses a LTE network to transmit the MSD to the PSAP. The voice call in LTE (VoLTE) uses an IMS (see chapter 3.1) which is an integral part of the network. The MSD is transmitted very fast inside the call setup (SIP invite) as pure data. If no LTE network is available, the standard eCall procedures via GSM or UMTS are triggered. For more details on "legacy" eCall procedure see [2].

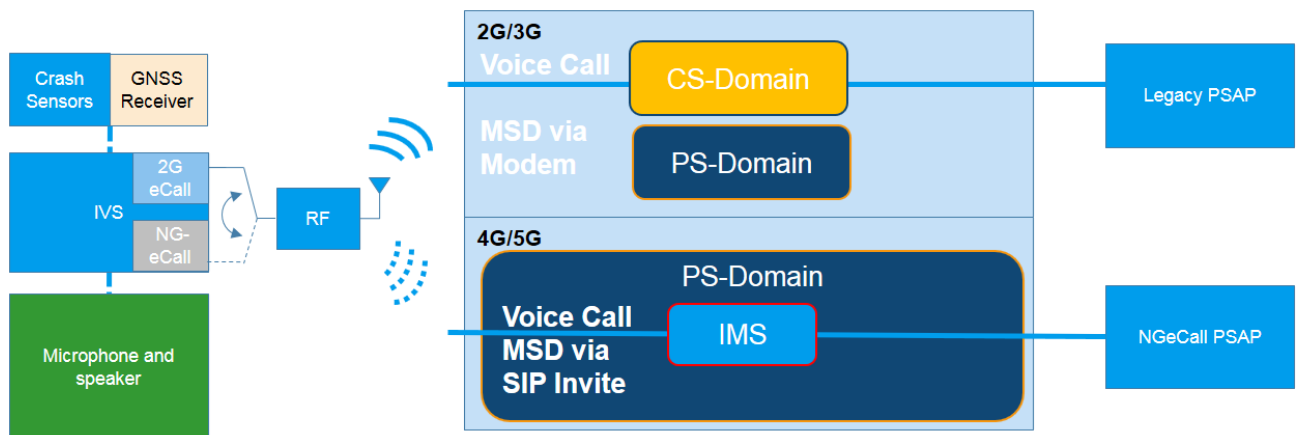


Figure 5: Coexistence eCall and NG eCall

Coexistence eCall and NG eCall: The IVS (left side) uses in NG eCall a voice call in the public-switched (PS)-domain to transmit the MSD to the (NG eCall)-PSAP. The LTE network indicates the support of NG eCall via eCallOverIMS-support flag.

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). Then the voice connection can be used to request additional information.

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

Table 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



### 3.3 Sequences

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

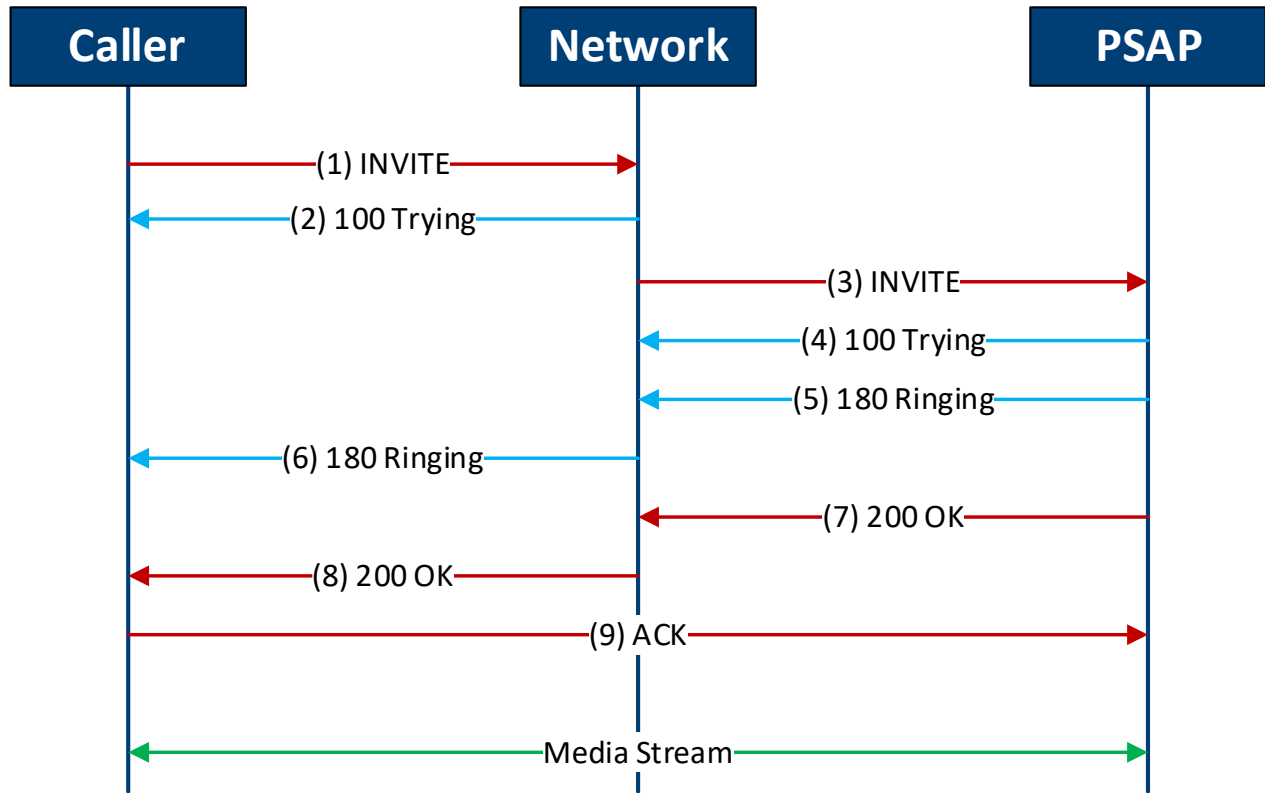


Figure 6: Typical NG eCall sequence. The call setup is done via a SIP request message INVITE

1. When an emergency occurs, the IVS automatically sets up a connection (emergency call) to the PSAP over LTE ("SIP message INVITE"). The answers by the network and the PSAP are TRYING and RINGING
2. The caller transmits the MSD inside the INVITE message
3. Once the PSAP has received the message, it sends back OK and the Caller answers with an ACK
4. The call is established. Any other data like a voice call, additional data or videos can be transmitted

The routing is handled by resource names in SIP:

- ▶ urn:service:sos.ecall.manual - Manual eCall
- ▶ urn:service:sos.ecall.automatic - Automatic eCall
- ▶ urn:service:test.sos.ecall - Test eCall

The LTE network indicates "eCallOverIMS-Support" in System Information Broadcast 1 (SIB1) message (Release 14). If there is no support inside the LTE network or there is no LTE network available at all, a circuit-switched fall back (CSFB) with GSM or UMTS is used. If there is a LTE network only which does not support IMS, the IVS uses the in-band modem and tries a voice call via VoLTE.

## 3.4 Standards

NG eCall is defined by several different standards. (Please note that this list is not complete):

- ▶ **IETF rfc8147**: Next-generation Pan-European eCall
- ▶ **IETF rfc7852**: Additional data related to an emergency call
- ▶ **IETF rfc8148**: Next-generation vehicle-initiated emergency calls
- ▶ **3GPP TS 23.167**: IP Multimedia Subsystem (IMS) emergency session
- ▶ **3GPP TS 23.401**: General Packet Radio Services (GPRS) enhancements for evolved universal terrestrial radio access network (E-UTRAN) access
- ▶ **3GPP TS 24.229**: IP multimedia call control protocol based on session initiation protocol (SIP) and session description protocol (SDP)
- ▶ **3GPP TS 26.269**: IMS eCall conformance of in-band modem
- ▶ **CEN TS 17184**: Intelligent transport systems - eSafety - eCall high level application protocols (HLAP) using IMS packet switched networks
- ▶ **CEN TS 17240**: End-to-end conformance test specification for IMS based packet switched systems

The solution from Rohde & Schwarz presented here fulfills all requirements from these standards and additionally ensures that the end-to-end conformance test complies with CEN TS 17240 [3] as discussed in the next chapter.

# 4 Conformance testing of IVS

## 4.1 Why test in the lab?

Testing of NG eCall 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 LTE 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.

## 4.2 Test setup

The test solution presented here covers the conformance test for NG eCall in line with CEN TS 17240. Figure 7 shows the test setup.

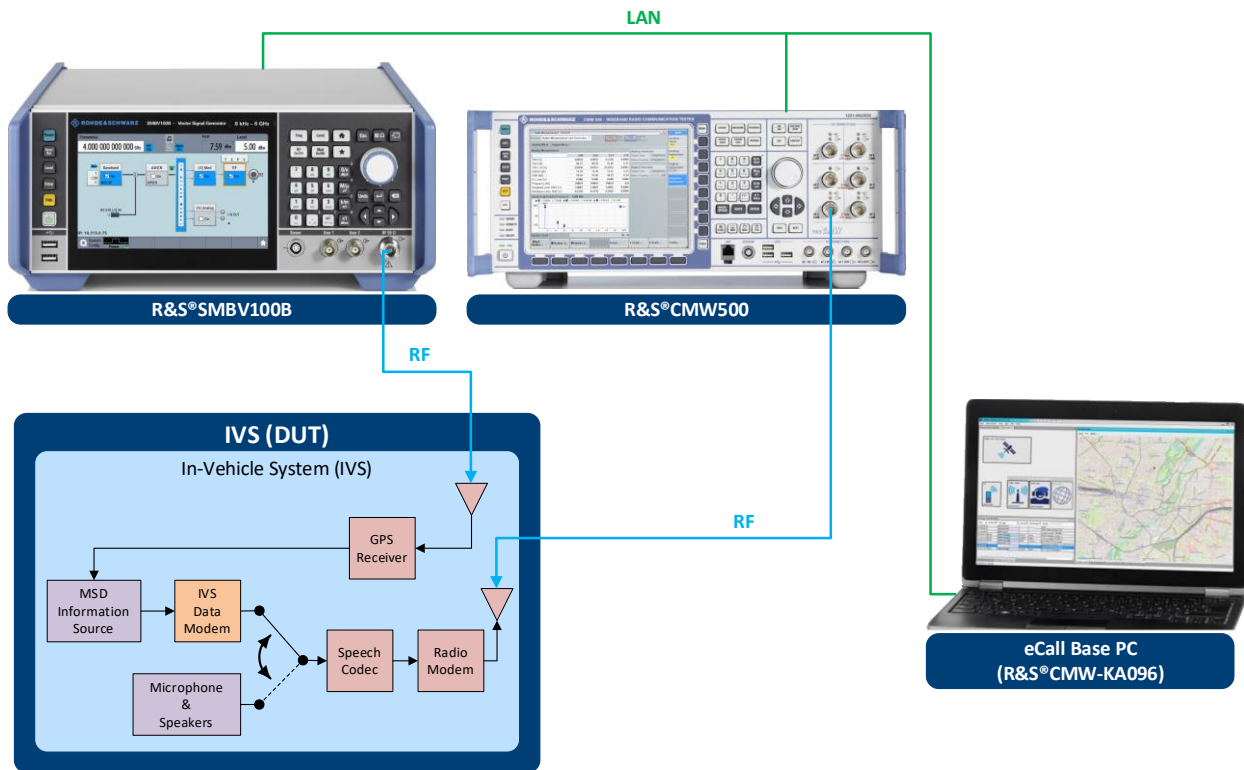


Figure 7: Test setup for the IVS conformance test

The CMW simulates the cellular network and provides an LTE (4G) cell. It also provides the IMS. Via the RF connection to the IVS the signaling (e.g. call setup) over LTE as well as the packet data connection takes place. The SMBV supplies simulated GNSS signals, such as GPS and Galileo, that are used by the IVS for positioning. The CMWKA096 software installed on an external PC simulates the PSAP, sets necessary parameters for NG eCall 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 NG eCall protocol and the MSD transmission are also handled via the data 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.

## 4.2.1 CMW radio communication tester



Figure 8: CMW

The CMW is the all-in-one test solution for radio communications 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. It also has a built-in IMS to support testing of IP multimedia services.

## 4.2.2 SMBV100B vector signal generator



Figure 9: SMBV100B 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 signal for eCall.

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 in auto localization mode
- ▶ 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

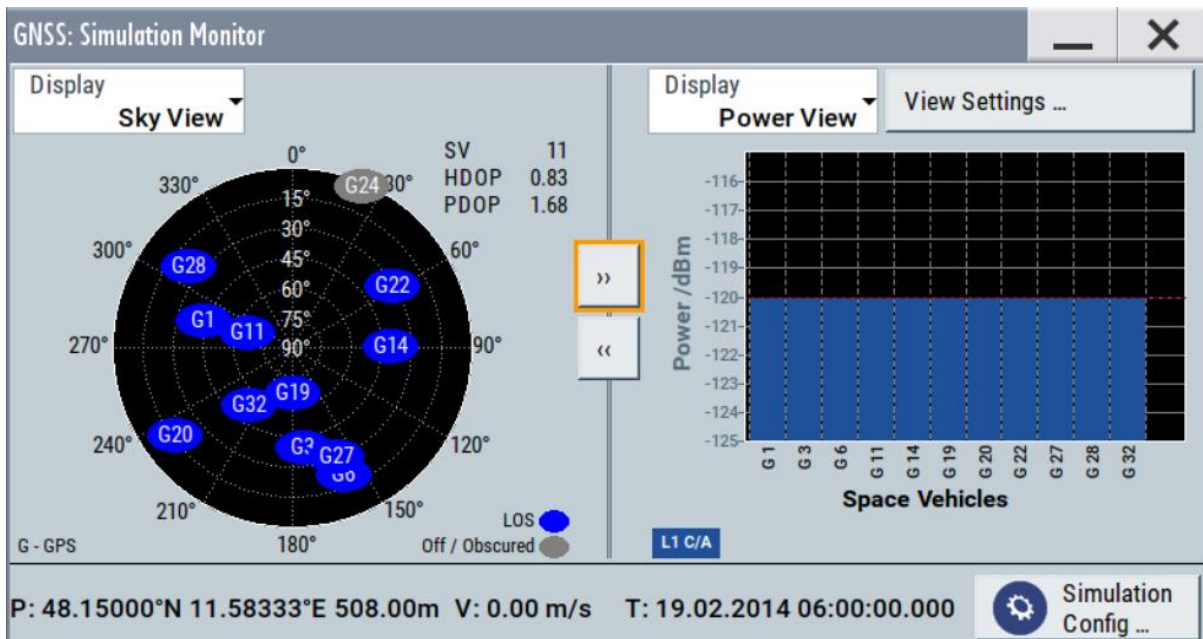


Figure 10: The sky view shows the distribution of the satellites above the local horizon

## Needed GNSS options

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

Table 3: GNSS scenarios

Scenario	Option SMBV100A	Configuration	Satellites
GPS - City	K44	Static, selectable city	Fixed satellite constellation
GLONASS	K94	Static, selectable city	Fixed satellite constellation
Galileo	K66	Customize	
GNSS channel extension to 24 channels	K99		
GNSS real world scenarios	K108		

The GPS city and GLONASS city scenarios both support the following cities:

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

### 4.2.3 KA096 PSAP simulator software (NG eCall test software)



Figure 11: CMW-KA096 software

The CMW-KA096 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 NG eCall over LTE
- ▶ Call ID
- ▶ MSD MIME type
- ▶ Raw MSD
- ▶ Dialed number/URN

- ▶ Trigger type
- ▶ MSD v2 decoding according to CEN EN 15722:2015 for every redundancy version and uplink data part
- ▶ Optional: Fixed position GPS/GLONASS simulation / GPS moving scenario

## 4.3 The first NG eCall measurement: Basic operation

The following sections describe the general process for setting up an NG eCall. The procedures describe the test system preparation and how an NG eCall with MSD transfer from the IVS to the simulated PSAP can be done.

Use the test setup shown in Figure 7.

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

To perform over-the-air tests (OTA tests), it is now possible to allow only dedicated IMSI numbers to connect to the CMW (GSM only).

The eCall application base and GUI are installed on a PC. The application simulates a PSAP and remote controls the CMW via LAN.

### 4.3.1 Preparing the test setup

The following steps prepare the test setup and start the eCall application.

1. Prepare the test setup:
  - 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 LTE 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
2. Setup the PC. The required steps are described in detail in the CMW-KA096 user manual, section "Preparing the test system for use"

The following list gives an overview:

- Install the eCall application software
- Install the .NET framework 4.0
- Install a VISA library with development support for .NET framework 4.0
- Make the license CMW-KA096 available, for example by attaching a smart card with the license

### 4.3.2 SIM card (UICC)

The Subscriber Identity Module (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 base stations 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 UEs 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.

### 4.3.3 Starting and configuring the NG eCall test software (KA096)

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 some settings are already modified it is recommended to reset the settings to their default values by deleting or renaming the following configuration file before starting the application:

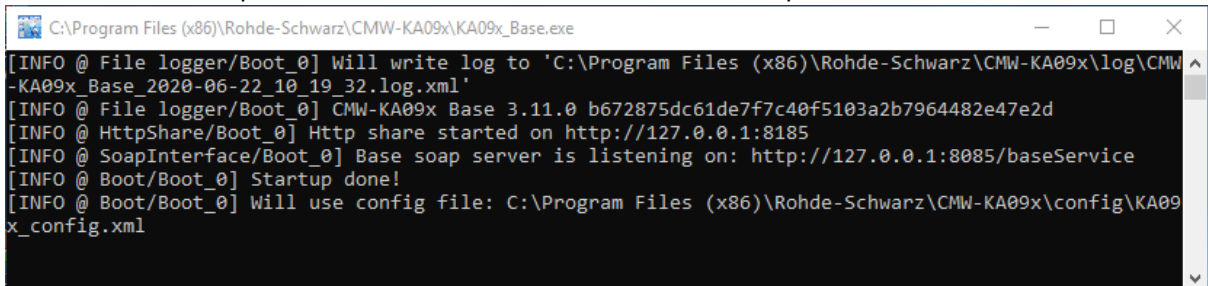
```
%APPDATA%\Rohde-Schwarz\CMW-KA09x_GUI\<version>\user.config
```

The eCall test software consists of two parts:

1. Start the eCall test software base:

"Windows start" menu → "All programs" → "R&S CMW-KA09x" → "eCall application base"

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

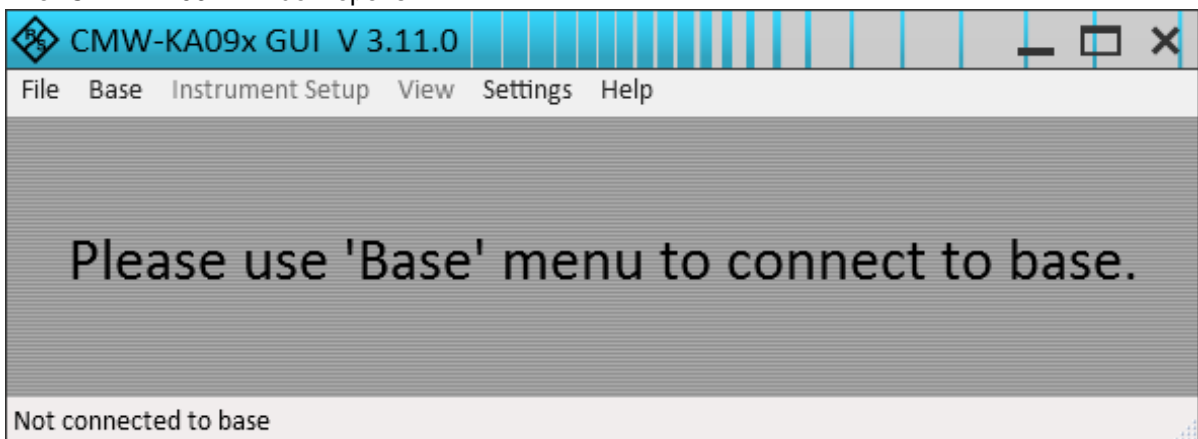


```
C:\Program Files (x86)\Rohde-Schwarz\CMW-KA09x\KA09x_Base.exe
[INFO @ File logger/Boot_0] Will write log to 'C:\Program Files (x86)\Rohde-Schwarz\CMW-KA09x\log\CMW
-KA09x_Base_2020-06-22_10_19_32.log.xml'
[INFO @ File logger/Boot_0] CMW-KA09x Base 3.11.0 b672875dc61de7f7c40f5103a2b7964482e47e2d
[INFO @ HttpShare/Boot_0] Http share started on http://127.0.0.1:8185
[INFO @ SoapInterface/Boot_0] Base soap server is listening on: http://127.0.0.1:8085/baseService
[INFO @ Boot/Boot_0] Startup done!
[INFO @ Boot/Boot_0] Will use config file: C:\Program Files (x86)\Rohde-Schwarz\CMW-KA09x\config\KA09
x_config.xml
```

2. Start the eCall test software GUI:

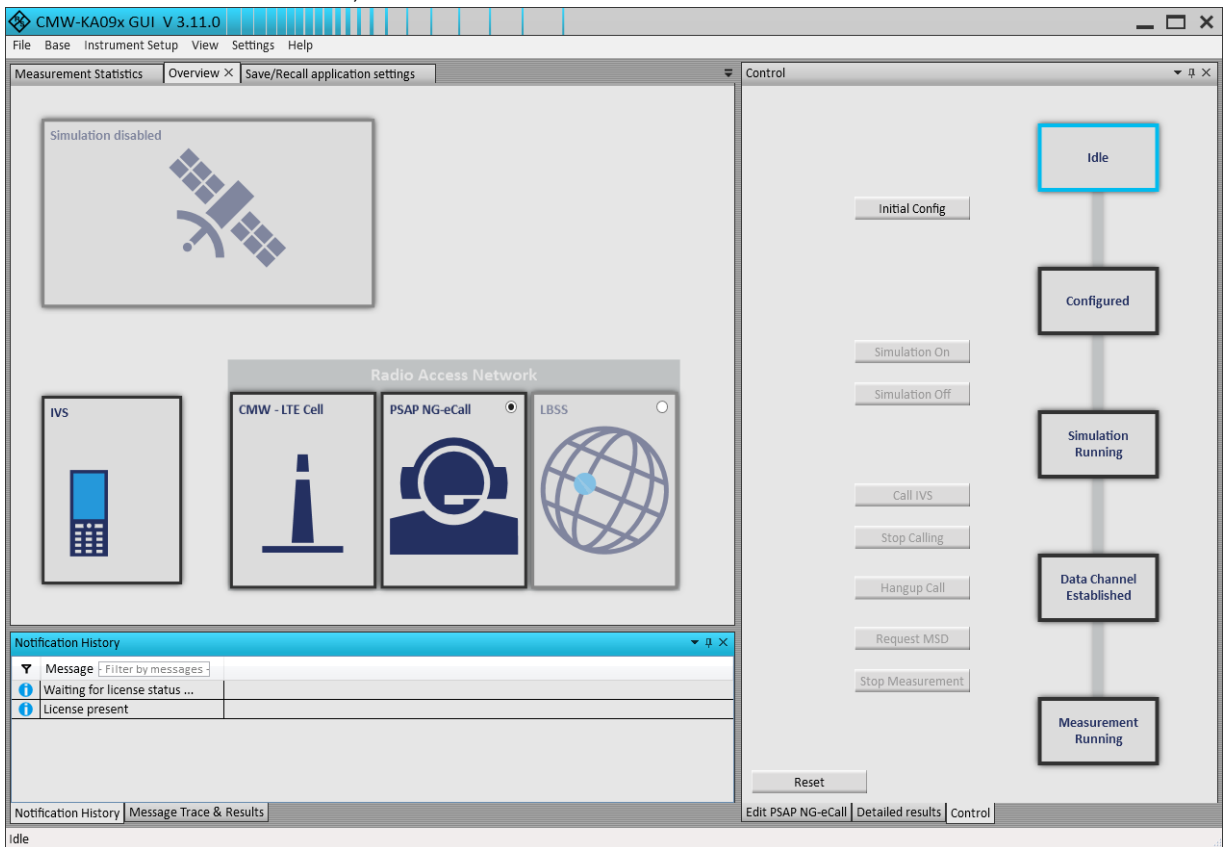
"Windows start" menu → "All programs" → R&S CMW-KA09x" → "eCall application GUI"

The "CMW-KA09x" window opens



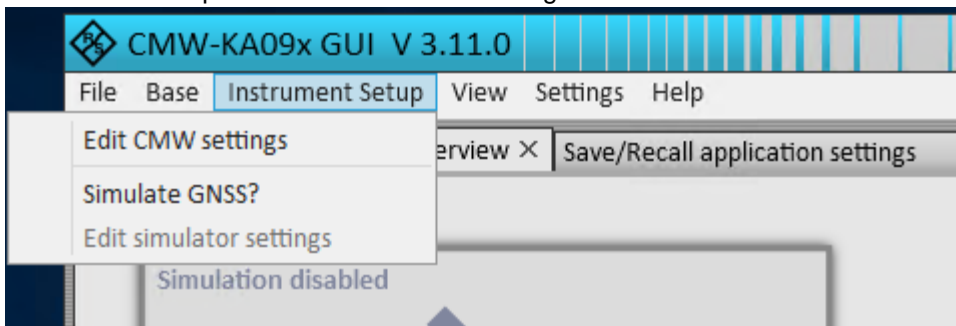


3. Connect the GUI to the base:  
 "Base" menu → "Connect"  
 After a successful connection, the GUI looks as follows

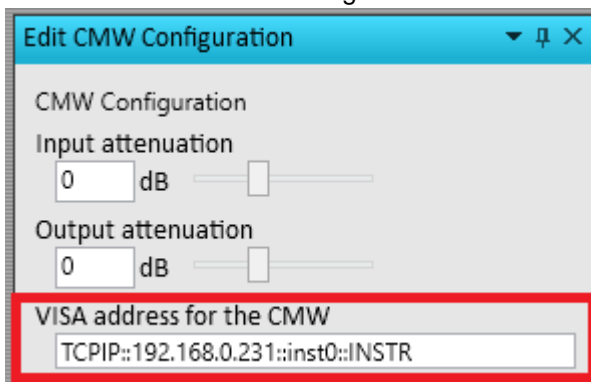


The status bar indicates the base state "Idle"

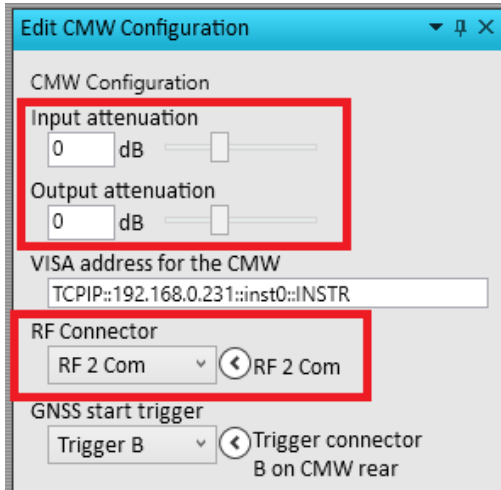
4. Configure the CMW settings:  
 "Instrument Setup" menu → "Edit CMW settings"



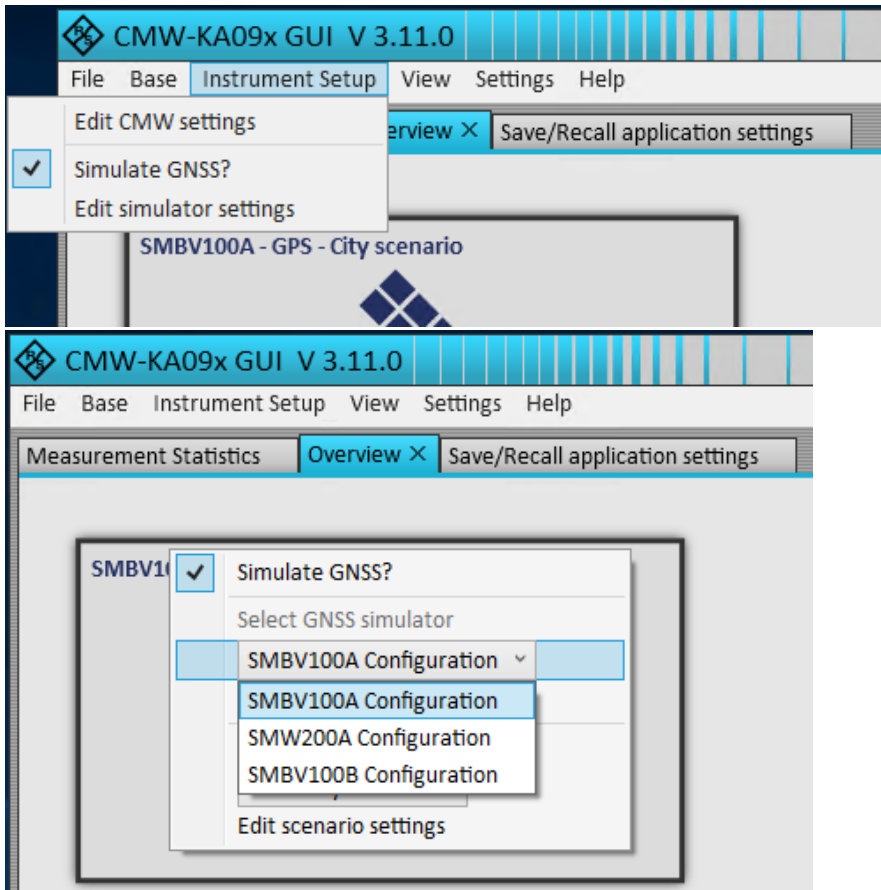
- a) Enter the VISA address string or the TCPIP address of CMW



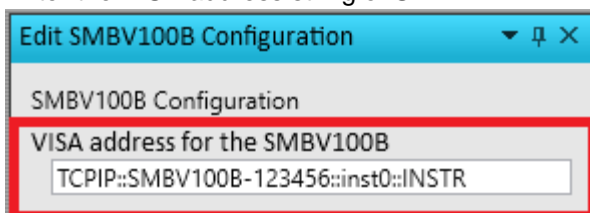
- b) Select the RF connector to be used for the LTE signal and configure the input and output attenuations according to the attenuation of the used RF cable



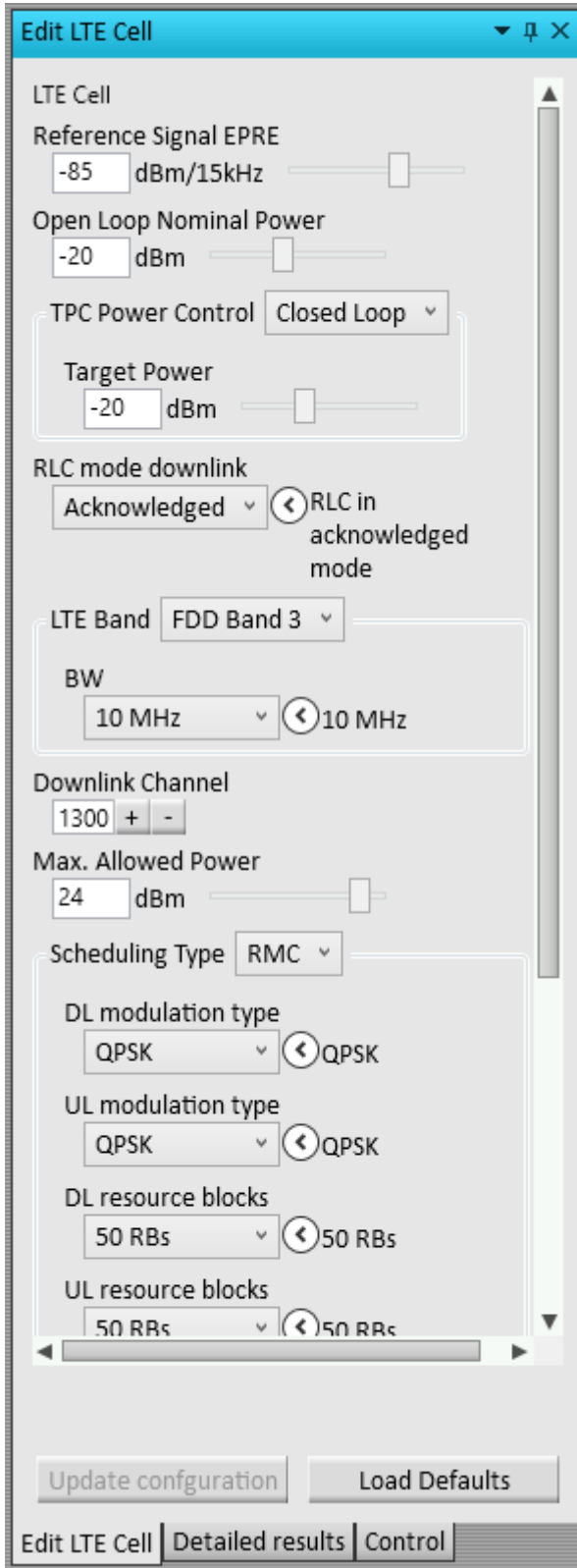
5. Select the SMBV as GNSS simulator:



6. Enter the VISA address string of SMBV



7. Right-click the base station area in the section Radio Access Network on the left  
Here select the network LTE and click "Edit Cell Settings"



8. Check that the cell settings are compatible to the IVS.  
If required, modify the settings, for example the band, the channel number and the power levels.  
If some settings were modified after the initial configuration, click "Update Configuration" at the bottom to apply the changes.

9. Configure the PSAP

**Edit PSAP NG-eCall**

PSAP NG-eCall

IMS Configuration

IMS Authentication XOR

Authentication Scheme  
AKA1 AKA1

Key  
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

AMF  
0 + -

IPsec Enable

Integrity Algorithm  
HmacSha1\_96 RFC 2404

Encryption Algorithm  
AesCbc RFC 3602

Call QoS signaling type  
Preconditions With preconditions

Dedicated bearer  
Request dedicated bearer A dedicated bearer is automatically setup for calls

Private user ID  
001010123456789@test.3gpp.com

Audio codec AMR narrow band

Force codec on MO call

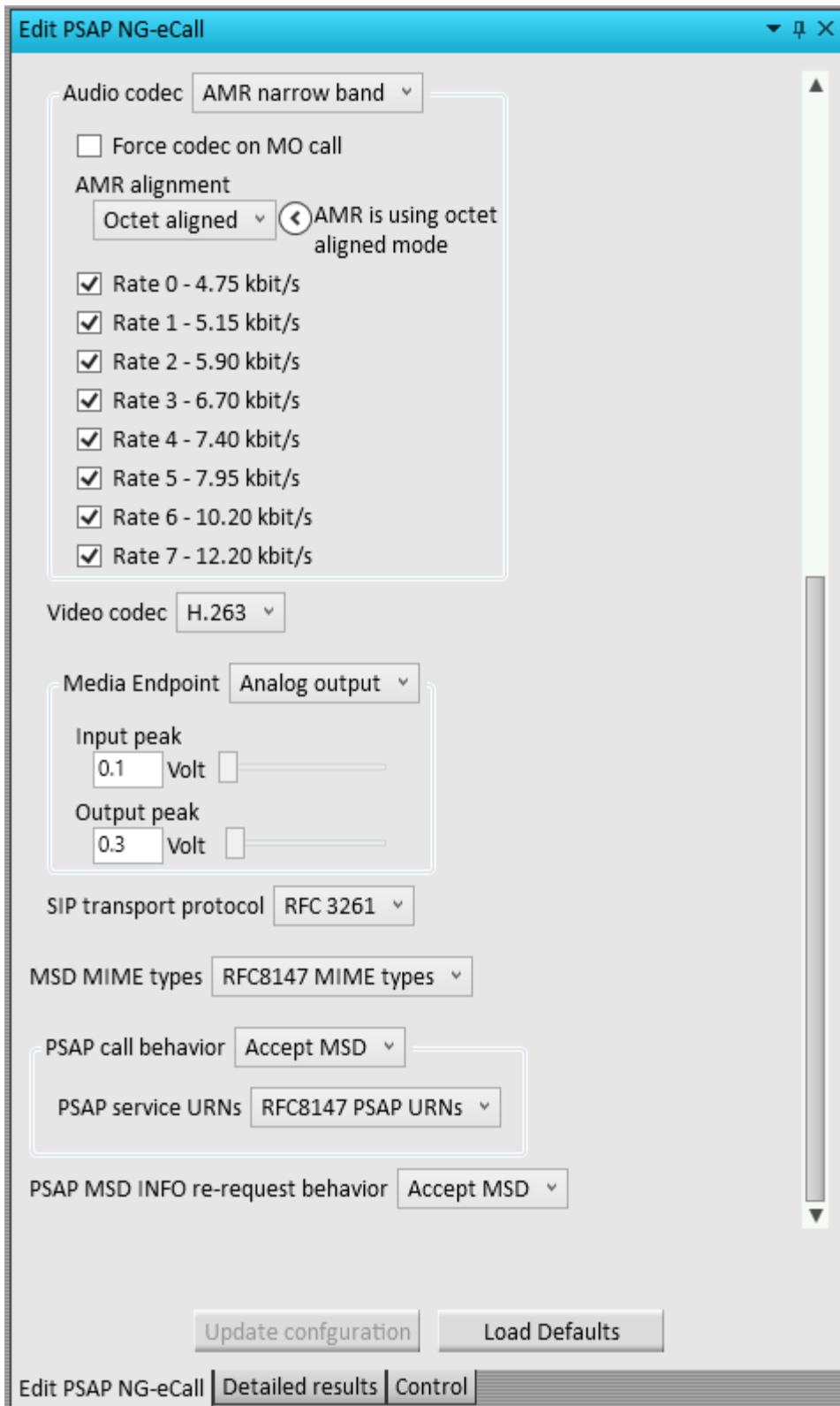
AMR alignment  
Octet aligned AMR is using octet aligned mode

Rate 0 - 4.75 kbit/s

Rate 1 - 5.15 kbit/s

Update configuration Load Defaults

Edit PSAP NG-eCall Detailed results Control



Special settings for IMS can be handled in section "IMS Authentication".

### 4.3.4 Performing an NG eCall

The steps in this section set up an NG emergency call, transfer an MSD from the IVS to the eCall application and analyze the MSD.

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

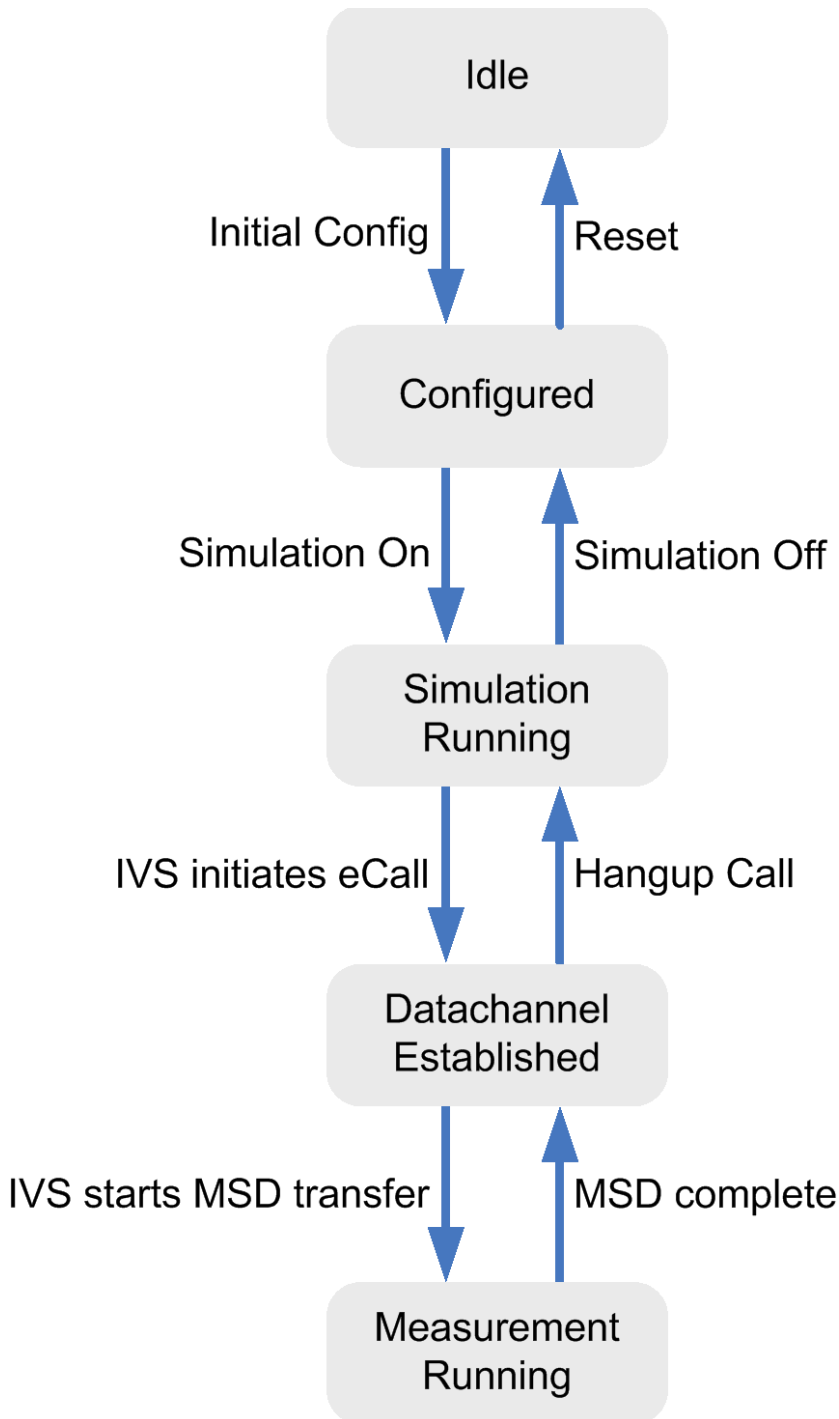
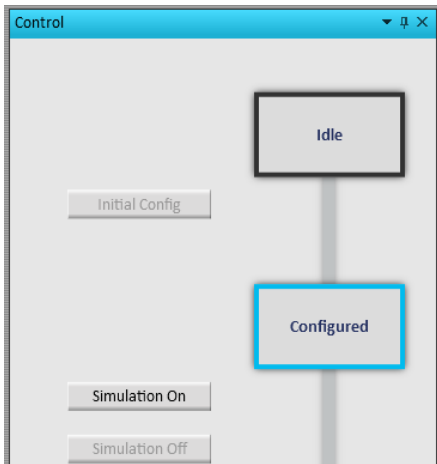
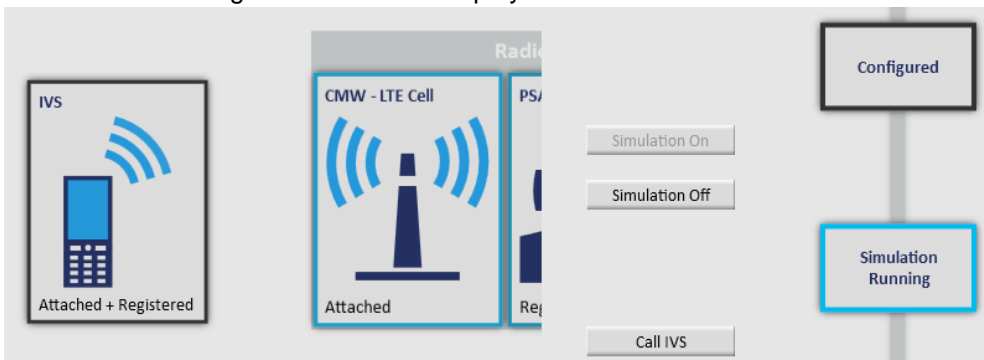


Figure 12: eCall application states

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 this action is performed 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 LTE simulations are started. The LTE 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 LTE area displays an active antenna.



4. Switch the IVS on and let it register.  
A successful registration is indicated in the "Notification History" view. The labels in the IVS area and the LTE area also indicate that the IVS has registered ("synchronized").
5. Initiate an eCall at IVS  
The call progress can be monitored in the "Overview" tab and the "Control" view. The following table lists the states for a successful eCall, starting in the first row and ending in the last.

IVS state	LTE state	PSAP state	Control state	Explanation
Attached + Registered	Attached	Registered	Simulation running	IVS is attached to LTE cell and registered to IMS
Connecting	Connecting			Call setup in progress
Connection established + Call established	Connection established	Call established	Data channel established	Data channel between IVS and PSAP established, MSD received

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

Time	Protocol	Source	Destination	Message	Details
11:07:26.6				LTE results	No further info
11:07:29.4				IMS Registration	Registered for normal calls
11:12:01.4				IMS Call	Call ID 'b0369547eba70008037623566037@fc01:abab:...
11:12:02.2				Raw MSD over SIP	SIP Invite - 0x02231804000000000000...
11:12:02.2				IMS NG-eCall	Manual NG-eCall from 'b0369547eba70008037623566'
11:12:02.2				Decoded eCall MSD V2	Decoded MSD successful
11:12:02.4	NG-eCall	UE/IVS	PSAP	NG-eCall MSD message	SIP Invite
11:12:03.0				IMS Registration	Registered for emergency calls

The "Detailed Results" view is automatically displayed on the right. It lists the MSD message contents.

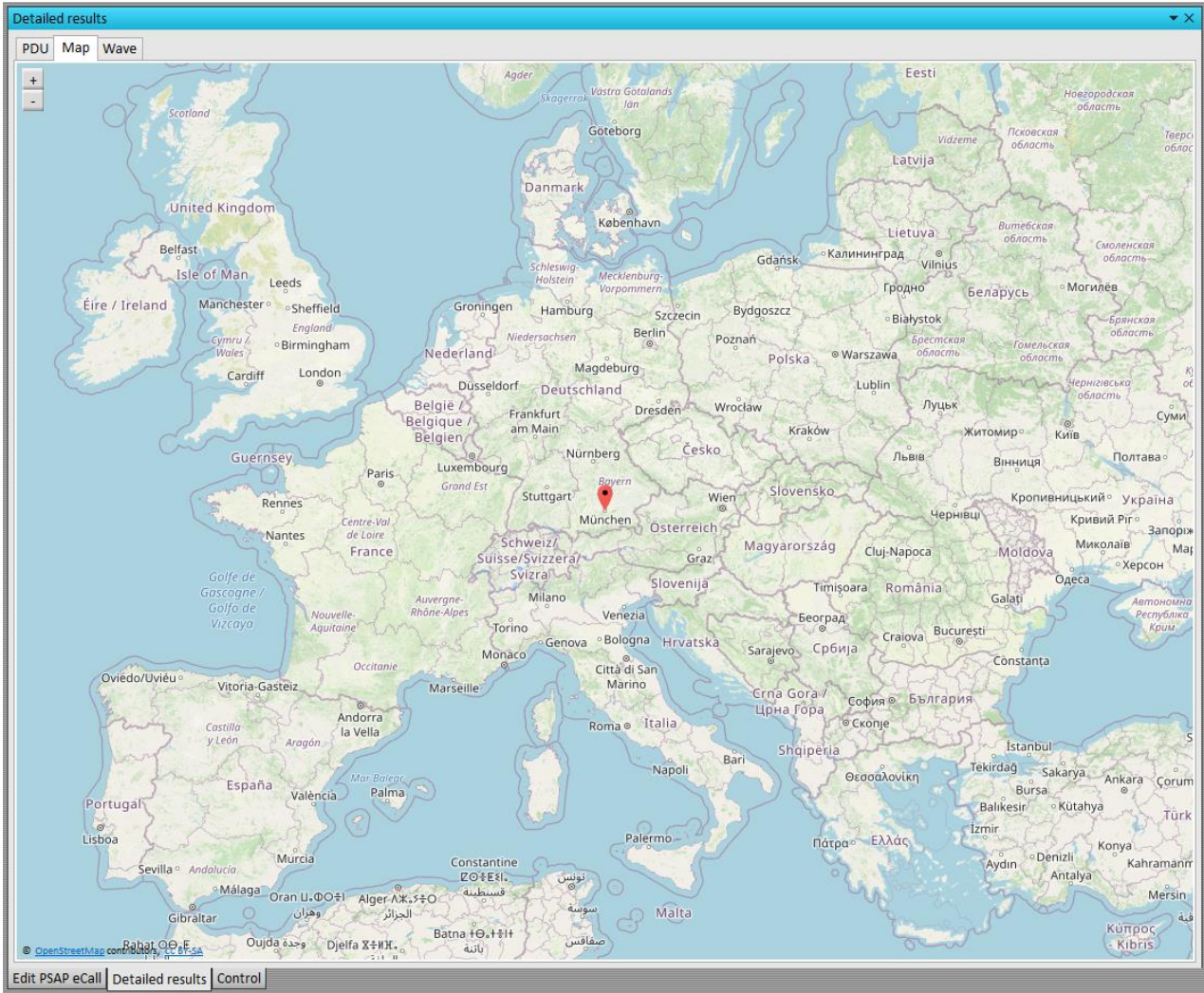
```

Decoded MSD message according to EN 15722 version 2015
Timestamp: 2020.06.22 - 11:12:02.2

Measurement ID = 1002
Source ID = 0
Decoding Information = 'Raw MSD from Invite, decoded according to EN 15722 20'
Decoding Status = Successful
  Vehicle location (MsdPositionInformation)
  Recent vehicle location 1 (MsdPositionInformation)
  Recent vehicle location 2 (MsdPositionInformation)
  message
    msdVersion = 2
    msd
      msdStructure
        messageIdentifier = 1
        control
          automaticActivation = False
          testCall = False
          positionCanBeTrusted = False
          vehicleType = passengerVehicleClassM1
        vehicleIdentificationNumber (VIN)
          isowmi = '000'
          isovds = '000000'
          isovisModelyear = '0'
          isovisSeqPlant = '0000000'
        vehiclePropulsionStorageType
          gasolineTankPresent = False
          dieselTankPresent = False
          compressedNaturalGas = False
          liquidPropaneGas = False
          electricEnergyStorage = False
          hydrogenStorage = False
          otherStorage = False
        timestamp = 0
        vehicleLocation
          positionLatitude = 0
          positionLongitude = 0
          vehicleDirection = 255
        recentVehicleLocationN1 (VehicleLocationDelta)
          latitudeDelta = 0
          longitudeDelta = 0
        recentVehicleLocationN2 (VehicleLocationDelta)
          latitudeDelta = 0
          longitudeDelta = 0
          numberOfPassengers (optional / not present)
          optionalAdditionalData (optional / not present)
          additionalData (optional / not present)
  
```



9. The eCall software shows the recorded position under the tab "Map"  
Please note that the software needs an online connection to show the map.



### 4.3.5 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-KA096 is connected to the PC. The smart card contents can be checked with the "R&S License Key Manager", see Windows Start menu. Close the license key manager before attempting a connection.
- ▶ "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 in 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 the CMW can be reached via LAN. For example, send a ping request from the PC to the CMW
  - Check the VISA address string entered in

#### Registration fails

If the registration of the IVS to the LTE 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?
- ▶ Are all LTE cell settings compatible to the IVS?

#### MSD transfer fails

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

- ▶ Observe the IVS, LTE and PSAP 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? Has a SIP invite been sent with MSD inside?
- ▶ If a MSD is received, but cannot be decoded, the raw MSD data can be checked. In the "Results Overview" double-click the row "Raw MSD over SIP"
- ▶ 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"

## 4.4 Measurements in line with specification CEN TS17240

The conformance tests for an IVS are specified in CEN TS 17240, chapter 9 "eCall end to end conformance testing". These tests are listed in Table 6.

Each test is identified via a conformance test point (CTP) number, listed in table column "CTP" and a chapter title listed in column "Name".

Some tests can be considered to be passed as soon as other tests have been passed. For more information, see [4], Section 7.4. The tests in the right column are considered to be passed as soon as the corresponding tests in the left column have been passed.

Table 4: Implicitly passed tests for general eCall IVS

Essential tests	Tests which may be assumed to be passed
1.1.2.2	1.1.1.1 1.1.2.1
1.1.3.2	1.1.3.1
1.1.9.1	1.1.4.1 1.1.5.1
1.1.17.2	1.1.17.1

Following tests required in EN 16454 for circuit switched IVS are not necessary in packet switched mode. They marked as "not required":

- ▶ 1.1.0.1
- ▶ 1.1.5.7
- ▶ 1.1.11.1
- ▶ 1.1.12.1
- ▶ 1.1.13.1
- ▶ 1.1.14.1
- ▶ 1.1.15.4

Table 5: Implicitly passed tests for "eCall-only" IVS

Essential tests	Tests which may be assumed to be passed
1.1.1.3	1.1.10.4

To perform a specific test with the eCall 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.

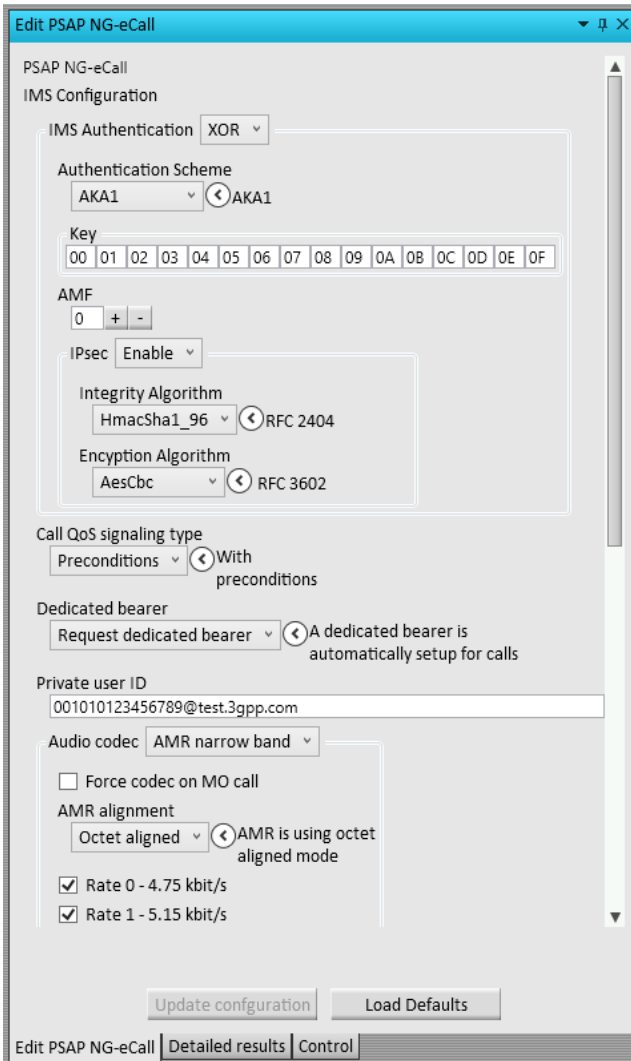


Figure 13: Default PSAP settings - page 1

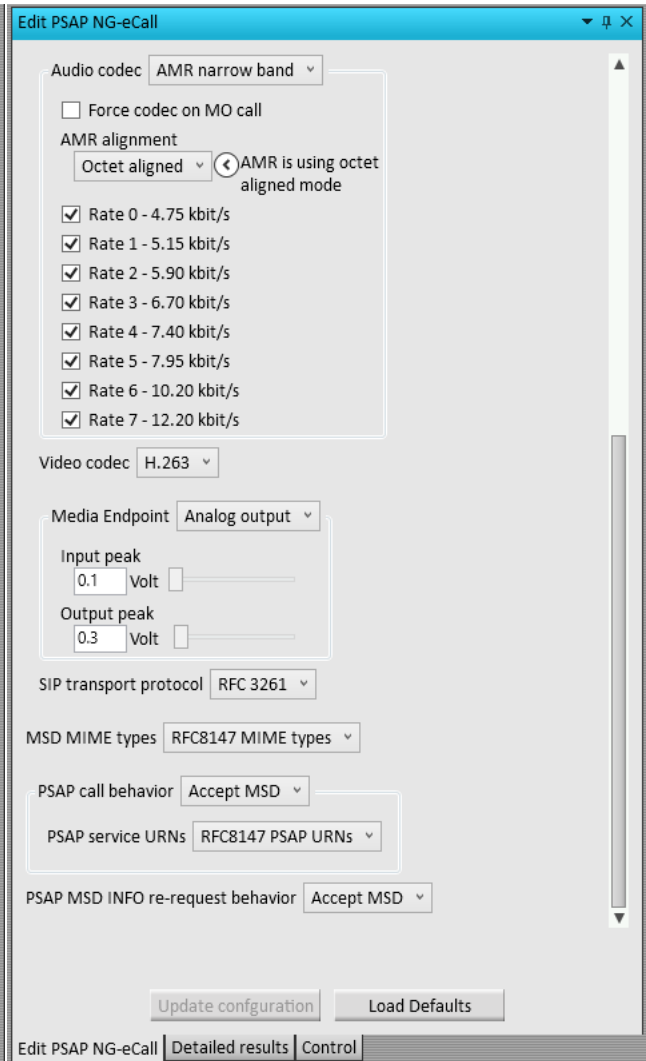


Figure 14: Default PSAP settings - page 2

Most of the tests can be performed with the "Default" PSAP settings shown in the figure. If different settings are required, the required modifications are listed in the table column "PSAP Settings".

The last table column indicates the check required for each test. Where to find the results to check are explained below the table.

Table 6: Conformance tests

CTP	Name	PSAP settings	Check / pass condition
1.1.0.1	IMS Conformance to ETSI TS 136 523 and ETSI TS 134 229 – 112-eCall IVS via IMS		
1.1.0.2	Test for conformance to valid SIM/USIM	Default	Cell state must change to "Synchronized"
1.1.0.3	Automatic eCall triggering does not occur when ignition OFF	Default	Cell state must stay in "Sync. Sig. On"
1.1.1.1	Power on and self test	Test out of scope of CMW-KA096	
1.1.1.2	IVS does not perform registration after power-up	Default	Cell state must stay in "Sync. Sig. On"
1.1.1.3	IVS periodically scans and maintains a list of available PLMNs	No test required. Covered by ETSI TS 122 011 and ETSI TS 123 122, therefore if compliance has been achieved to CTP 1.1.1.0.1, is automatically compliant.	
1.1.2.1	Test for automatic activation of eCall	Default	Cell state must change to "Call Established"
1.1.2.2	Automatically triggered eCall in progress was not disconnected upon a new eCall trigger	Default	Cell state must stay in "Call Established" on second trigger
1.1.2.3	Post-Lateral-crash performance of automatic trigger	Default	MSD received successfully at PSAP
1.1.2.4	Post-frontal-crash performance of automatic trigger	Default	MSD received successfully at PSAP
1.1.2.5	Performance of automatic trigger - different crash types	Default	MSD received successfully at PSAP if trigger above threshold
1.1.3.1	eCall manually activated	Default	Cell state must change to "Call Established"
1.1.3.2	Manually triggered eCall in progress was not disconnected upon a new eCall trigger	Default	Cell state must change to "Call Established" on second trigger
1.1.4.1	Test eCall activated	Default	Cell state must change to "Call Established"
1.1.5.1	Network registration	Default	Cell state must change to "Synchronized" or "Call Established"
1.1.5.2	Manual termination of eCall by vehicle occupants not allowed (automatically triggered eCall)	Default	MSD received successfully at PSAP Cell state = "Call Established"
1.1.5.3	Manual termination of eCall by vehicle occupants not allowed (manually triggered eCall)	Default	MSD received successfully at PSAP Cell state = "Call Established"

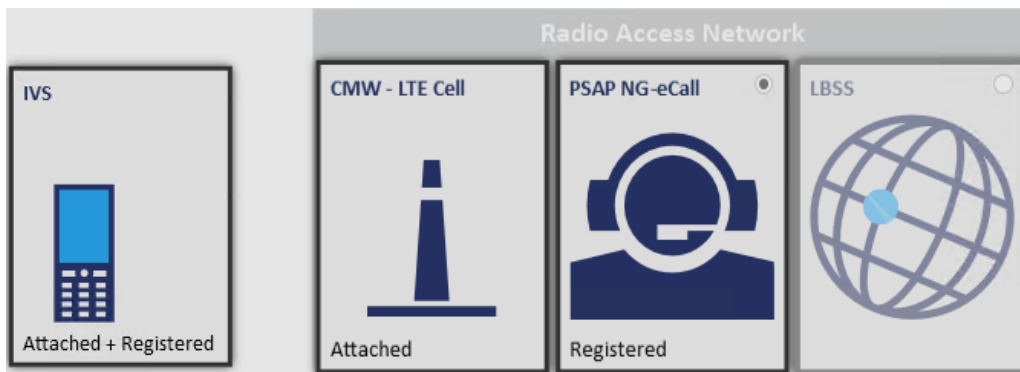
1.1.5.4	Automatically triggered eCall in progress was not disconnected when ignition is switched to OFF	Default	MSD received successfully at PSAP Cell state = "Call Established"
1.1.5.5	Manually triggered eCall in progress was not disconnected when ignition is switched to OFF	Default	MSD received successfully at PSAP Cell state = "Call Established"
1.1.5.6	Priority over conflicting communication	Default	MSD received successfully at PSAP Cell state = "Call Established"
1.1.5.7	Network registration is re-tried when network registration attempt was not successful	No test required. (This eventuality is covered by ETSI TS 134 123-1, clause 6).	
1.1.6.1	SIP Invite sent	Default	Check MSD reception and audio
1.1.7.1	Establish session with urn:service:sos.ecall.automatic	Default	Check call info: automatic eCall
1.1.8.1	Establish session with urn:service:sos.ecall.manual	Default	Check call info: manual eCall
1.1.9.1	Set-up call to test address	Default	Check call info: non-emergency call
1.1.10.1	eCall is attempted when no networks are available (limited service condition)	Not implemented yet	
1.1.10.2	Re-dial attempt completed within 2 minutes after eCall is dropped	No longer required. Using IMS, if the call succeeds to connect, the MSD is already in the hands of the PSAP	
1.1.10.3	ALLOW ACK Received	This test is not required in IMS-eCall (MSD part of SIP invite header)	
1.1.10.4	Verify that PLMN registration procedure is executed upon initiating an eCall	Default	Cell state must change to "Synchronized" and "Call Established"
1.1.15.1	Voice link Established	Default	Check MSD reception and audio
1.1.15.2	Verify MSD Received	Default	Check MSD reception and audio
1.1.16.1	Clear down call automatically	Test not required. PSAP operator simply hangs up	
1.1.16.2	IVS clears down the eCall upon T2 expiry	Default	Cell state must change from "Call Established" to "Synchronized" after 1 h
1.1.16.3	IVS registers recent eCalls	Default	Check IVS internal memory All recent eCalls must be registered
1.1.17.1	Call-back allowed and able to be answered by IVS	Default	Cell state must change to "Call Established"
1.1.17.2	Call-back answered by IVS in the event of abnormal termination	Not implemented yet	
1.1.17.3	MSD transfer occurs upon PSAP request during call-back	Default	MSD received successfully at PSAP
1.1.17.4	Remain registered for $\geq 1$ hr	Default	Cell state must stay in "Synchronized" for at least one hour
1.1.17.5	Remain registered for $\geq 1$ hr $\leq 12$ hr	Default	Cell state must change from "Synchronized" to "Sync. Sig. On" between 1 and 12 hours



The following sections explain how the simple checks listed in the last table column can be performed with the eCall test software.

### Checking the cell state

Check the cell state in the LTE area on the left. The state is indicated below the antenna image.



The following values are relevant for conformance test checks:

- ▶ "On": The cell signal is on
- ▶ "Attached": The IVS has registered to the mobile network
- ▶ "Call Established": An eCall has been established

### Checking MSD Reception and MSD Contents

To check the reception of an MSD, open the view "Message Trace & Results". After successful MSD reception, there is an entry "Raw MSD over SIP" and an entry "Decoded MSD successful".

Time	Protocol	Source	Destination	Message	Details
11:07:26.6				LTE results	No further info
11:07:29.4				IMS Registration	Registered for normal calls
11:12:01.4				IMS Call	Call ID 'b0369547eba70008037623566037@fc01:abab:
11:12:02.2				Raw MSD over SIP	SIP Invite - 0x02231804000000000000...
11:12:02.2				IMS NG-ECall	Manual NG-ECall from 'b0369547eba70008037623566
11:12:02.2				Decoded eCall MSD V2	Decoded MSD successful
11:12:02.4	NG-eCall	UE/IVS	PSAP	NG-ECall MSD message	SIP Invite
11:12:03.0				IMS Registration	Registered for emergency calls

Figure 15: Results overview view: The received MSD is shown in the decoded easy-to-read form

To check the MSD contents, double-click the entry "Decoded MSD message". The message contents are displayed.

The following message contents are relevant for conformance test checks:

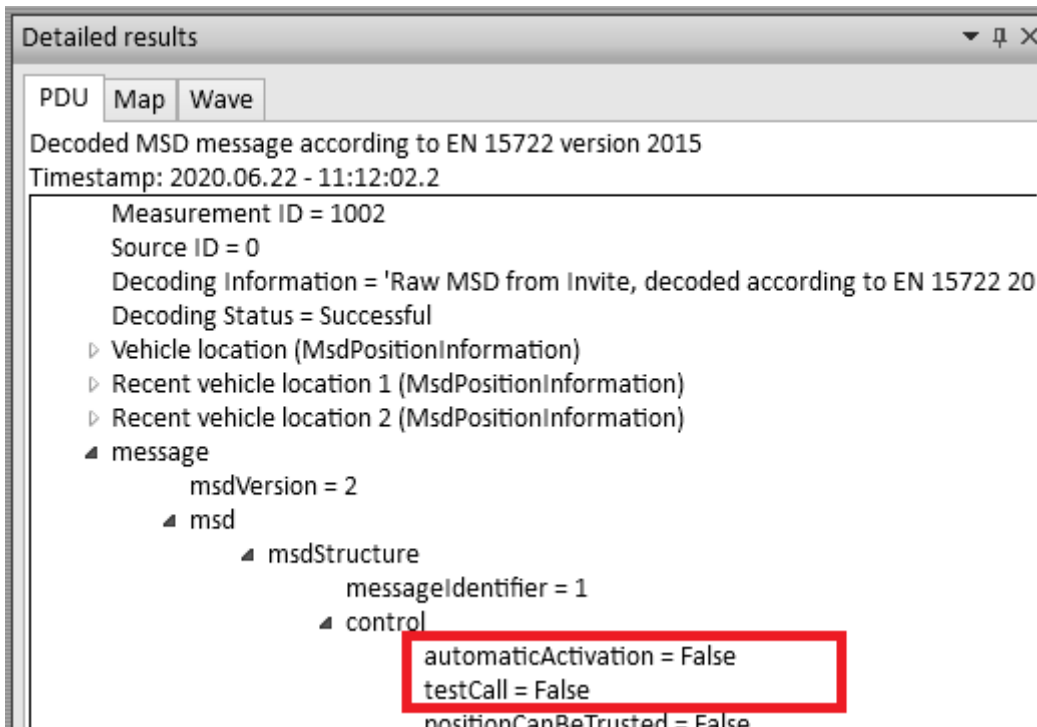


Figure 16: MSD contents "Manual activation" and "Test call"

### Checking call information

To check call information, open view "Message Trace & Results". Check for IMS NG-ECall and double click.

Message Trace & Results

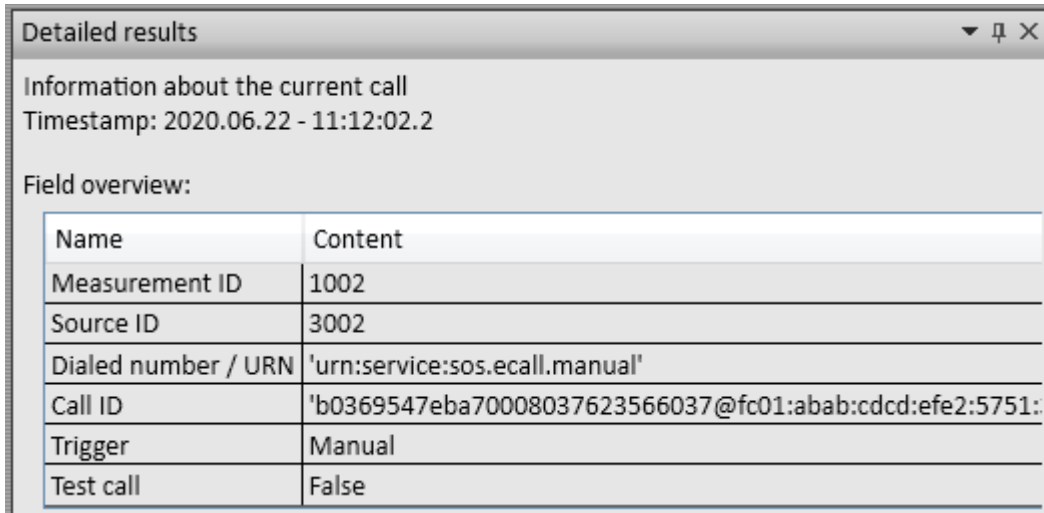
Time	Protocol	Source	Destination	Message	Details
11:07:26.6				LTE results	No further info
11:07:29.4				IMS Registration	Registered for normal calls
11:12:01.4				IMS Call	Call ID 'b0369547eba70008037623566037@fc01:abab:
11:12:02.2				Raw MSD over SIP	SIP Invite - 0x02231804000000000000...
11:12:02.2				IMS NG-ECall	Manual NG-ECall from 'b0369547eba70008037623566
11:12:02.2				Decoded eCall MSD V2	Decoded MSD successful
11:12:02.4	NG-eCall	UE/IVS	PSAP	NG-ECall MSD message	SIP Invite
11:12:03.0				IMS Registration	Registered for emergency calls

Notification History | Message Trace & Results

Figure 17: Results overview



Information about the call is displayed:



Detailed results

Information about the current call  
Timestamp: 2020.06.22 - 11:12:02.2

Field overview:

Name	Content
Measurement ID	1002
Source ID	3002
Dialed number / URN	'urn:service:sos.ecall.manual'
Call ID	'b0369547eba70008037623566037@fc01:abab:cdcd:efe2:5751:'
Trigger	Manual
Test call	False

Figure 18: Call information

For conformance test checks, three types of calls must be distinguished:

- ▶ Automatic eCall  
"Dialed number/URN" must be "urn:service:sos.ecall.automatic", Test call must be false
- ▶ Manual eCall  
"Dialed number/URN" must be "urn:service:sos.ecall.manual", Test call must be false.
- ▶ Non-emergency calls to test number  
"Dialed number/URN" may be "urn:service:test.sos.ecall" or dialed number representation, Test call must be true

## 4.5 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, the BASE can be controlled via SOAP by your own developed program or by CMWrun provided by Rohde & Schwarz.

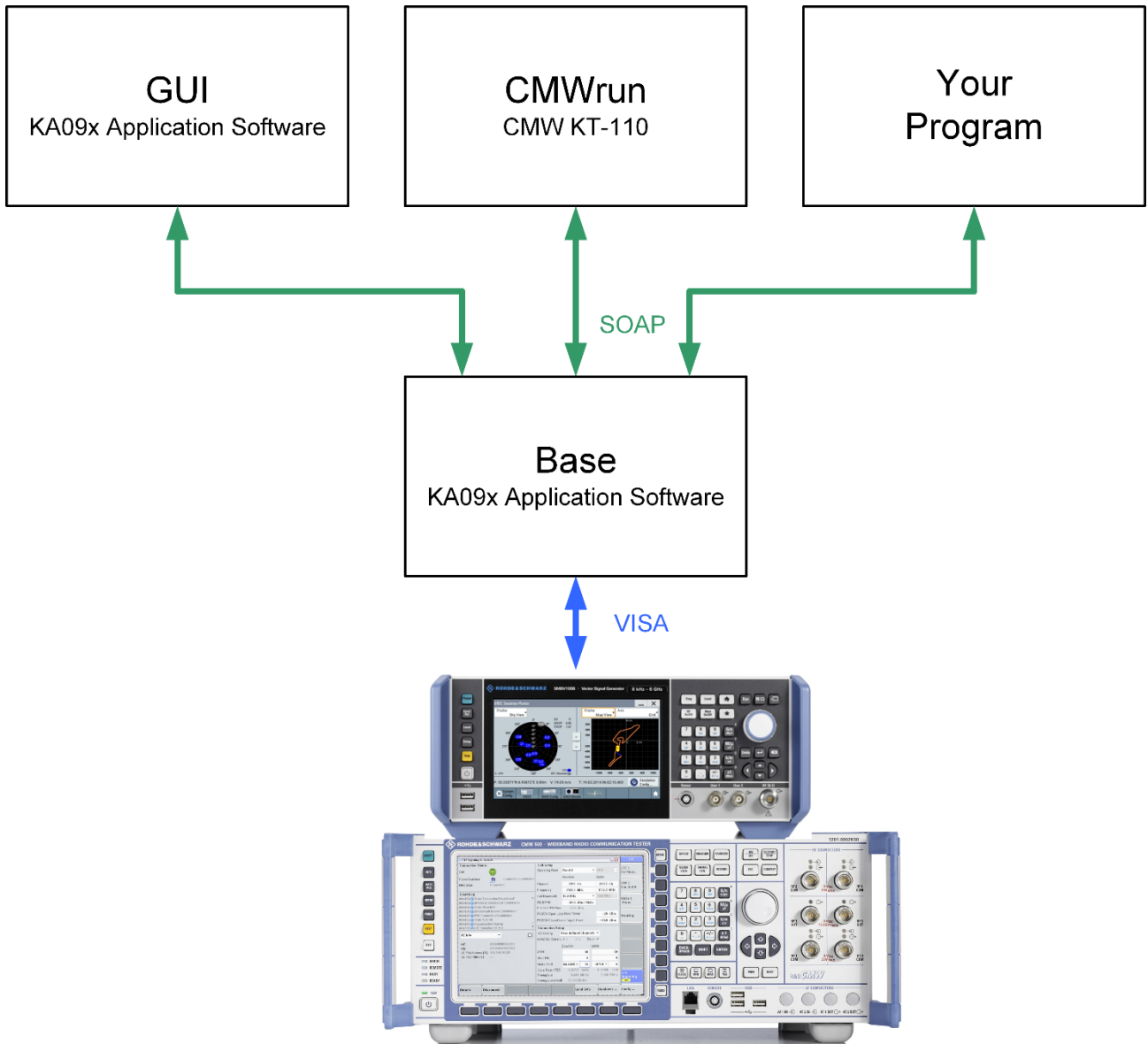


Figure 19: Three ways to control the BASE via SOAP. The GUI for manual control is provided together with the BASE as part of the KA09x installation. For automated tests CMWrun or own programs can be used.

#### 4.5.1 CMWrun option KT-111

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 KT111 for CMWrun remote controls the entire setup for NG eCall as ready-to-go solution for conformance testing in line with the specification CEN TS 17240. It is the right choice for configuring test sequences by remote control, easy handling of different IVS types and receiving the complete pass/fail test protocol. The KT111 software allows also the automation and the handling of the tested devices (by AT commands) in an easy and user friendly way.

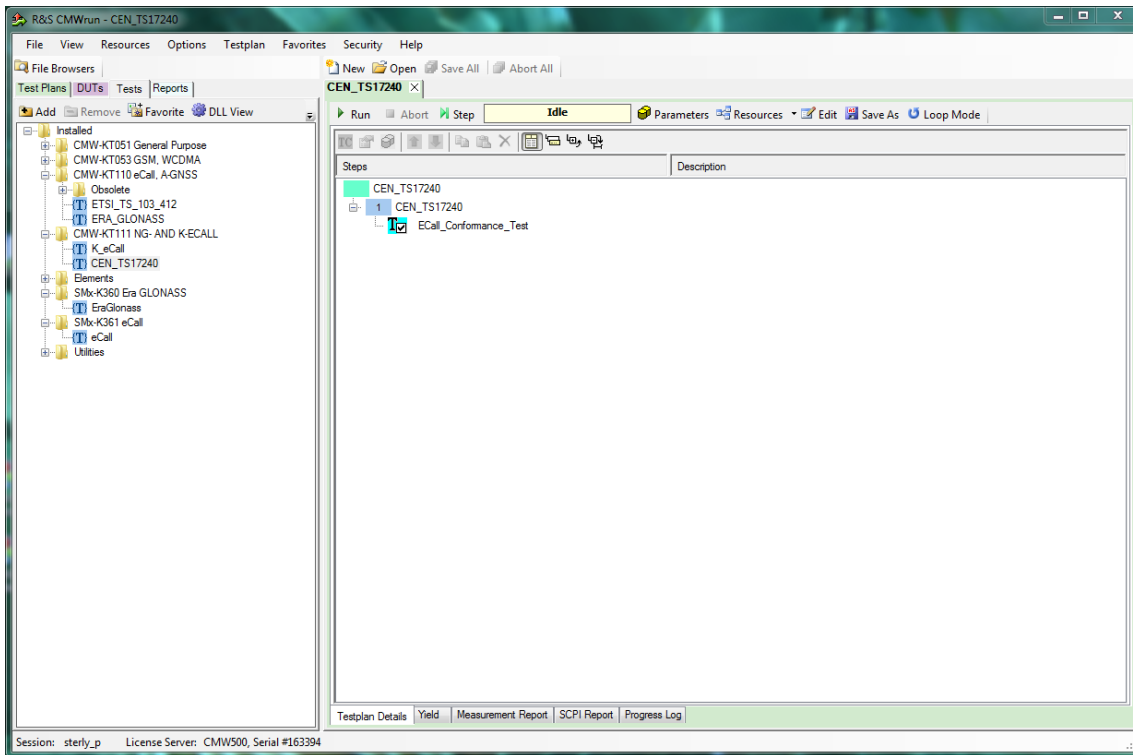


Figure 20: The main window of CMWrun. The option KT-111 covers NG eCall

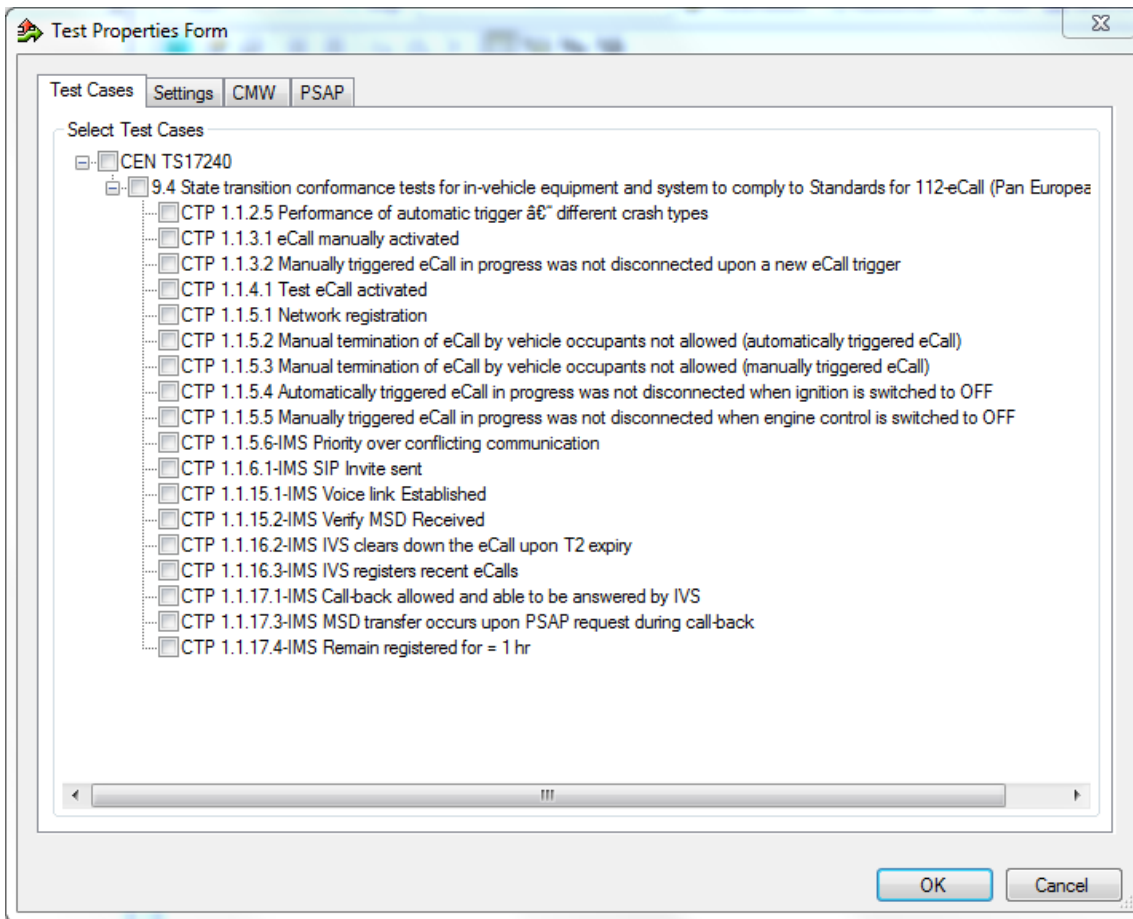


Figure 21: Test cases in CMWrun-KT111 according to CEN TS17240

## 4.5.2 Remote control of the BASE via SOAP

To use the eCall solution in an automated test system, the KA09x BASE application software can be remote controlled. SOAP is an open W3C standard protocol to exchange information and messages. For remote control the base software provides an API. A detailed documentation can be found under %APPDATA%\Rohde-Schwarz\CMW-KA09x\DOC\CMW-KA09x\_API\_en.chm. Thus configuration data (e.g. channel numbers) can be sent and data (e.g. the transmitted MSD) can be received.

Please note that the 'KA09x\_Base.exe' with the default uri must be running already. Please also see the manual, chapter 3 [5].

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.

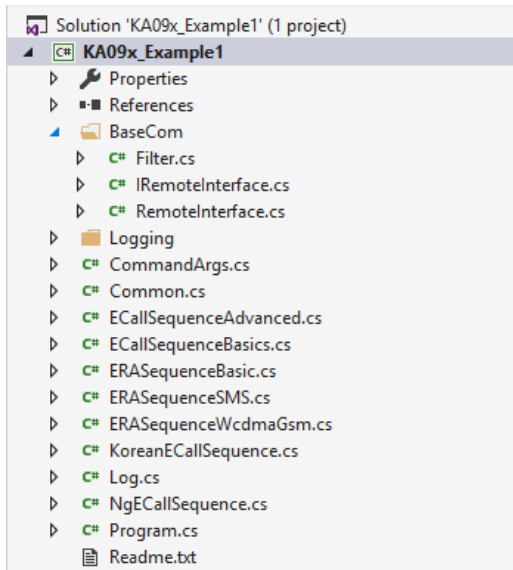


Figure 22: The files in the example project

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 provides the basic settings like the VISA address for the instruments like CMW or SMBV
- ▶ **BaseCom/Filter:** This class handles the incoming events from the Base to the client (e.g. you can wait for changes in the state machine of the Base)
- ▶ **BaseCom/RemotelInterface:** This class handles the basic communication (like connection and messages) via SOAP to the Base
- ▶ special examples:
  - **ERASequenceBasic:** a basic ERA-GLONASS call
  - **ERASequenceSMS:** an ERA-GLONASS SMS example
  - **ERASequenceWcdmaGsm:** an ERA-GLONASS example, gets the MSD two times: first via WCDMA, second via GSM
  - **ECallSequenceBasic:** a basic eCall
  - **ECallSequenceAdvanced:** a more advanced eCall
  - **NgECallSequence:** a basic NGeCall example (Details below)

## RemoteInterface

This class implements the client interface for the communication with the base. It uses the SOAP implementation 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 can be found in annex A.

## 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 25). For the method calls see the ECallSequenceAdvanced.

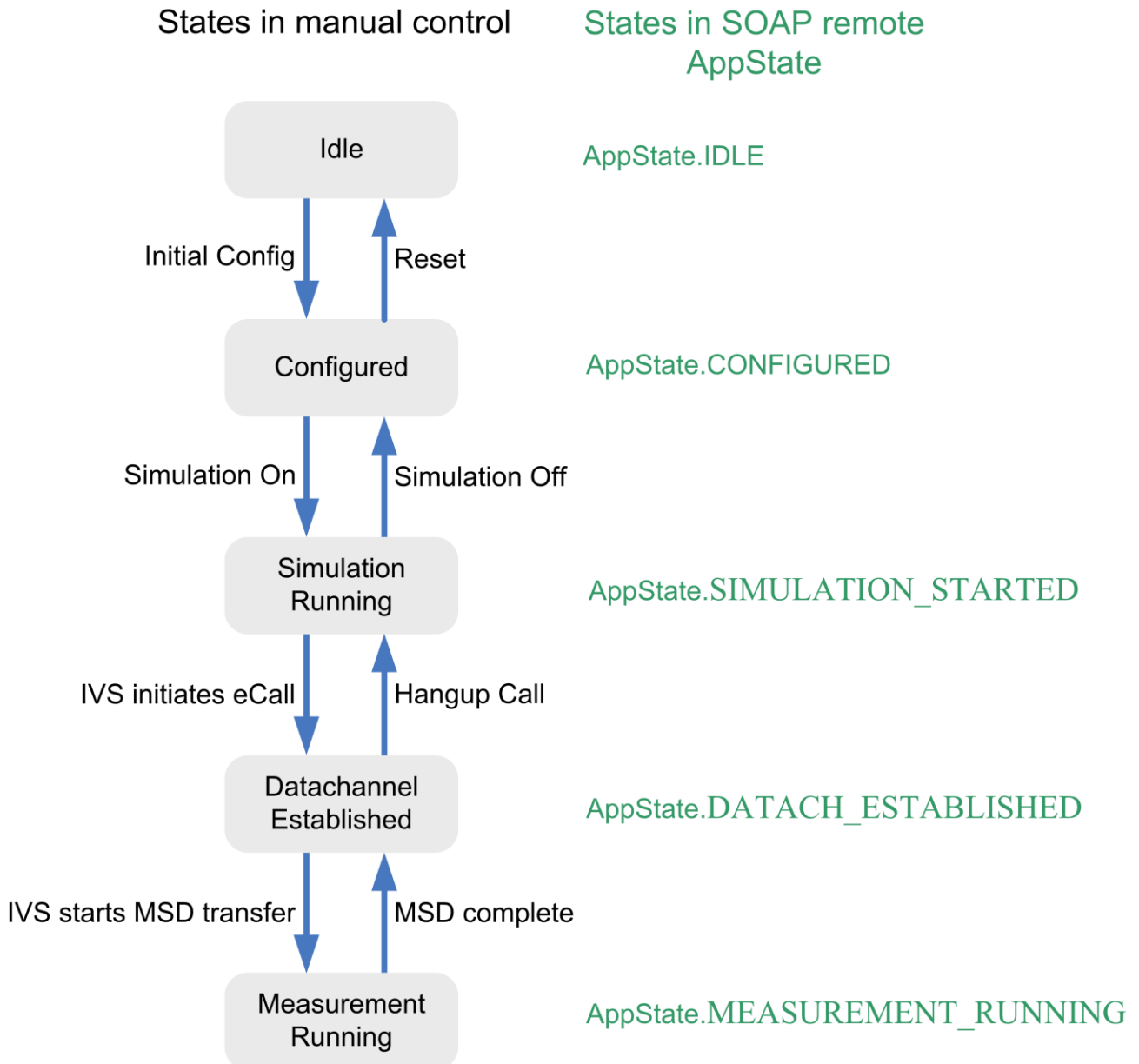


Figure 23: The different states of the Base. In remote control the appstate can be queried. Every state change creates an event 'NewState' that can be handled in the control software.

## 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, some global configuration is done in the class config:

```
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 smbv100aAddress = "GPIB::28::0::INSTR";
    public static string smw200aAddress = "GPIB::28::0::INSTR";
    public static string smbv100bAddress = "TCPIP::smbv100b-123456::INST0::INSTR";

    5 references
    public static KA09x_SoapV3.Ran.CfgGsm cfgGsm
    {
        get...
    }

    // Default to a 2G Universal SIM
    public static KA09x_SoapV3.Ran.Supplementaries.WcdmaSimCardType wcdmaSimType = WcdmaSimCardType.C2GUSIM;

    1 reference
    public static KA09x_SoapV3.Ran.CfgWcdma cfgWcdma
    {
        get...
    }

    2 references
    public static KA09x_SoapV3.Ran.CfgLte cfgLte
    {
        get
        {
            return new KA09x_SoapV3.Ran.CfgLte()
            {
                targetId = Common.RAN_ID,
                band = new LteBand3(),
                dlChannelNum = 1575,
            };
        }
    }
}
```

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)
{
    doCfgResultStep(f, r, new CfgCmw()
    {
        targetId = RAN_SIM_ID,
        rfConnector = Config.cmwRfConnector,
        visaName = Config.cmwAddress,
        attenuationInputInDb = Config.cmwRfAttenuation,
        attenuationOutputInDb = Config.cmwRfAttenuation,
    });
}
```

## Program

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

```
static void Main(string[] args)
{
    // We want console output
    Log.add(new Logging.ConsoleLogger());
    // We also want all the output to go to a log file
    Log.add(new Logging.FileLogger());

    try
    {
        doSequence(args);
    }
    catch (Exception e)
    {
        Log.that(e);
    }

    // And this is interactive, we'll use Console instead of Log here
    Console.WriteLine("Press enter to exit...");
    Console.ReadLine();
}
```

## ECallSequencyAdvances

NGeCallSequence: This sequence executes the NG eCall example. It has the following steps:

1. ResetBase
2. IntialConfig
3. DoConfiguration
4. SimOn
5. WaitForAll
6. ResetBase

```
static void doSequence(string[] args)
{
    Log.that(ProjectInfo.TITLE + " v" + ProjectVersion.MAJOR + "." + ProjectVersion.MINOR + "." + ProjectVersion.PATCH);
    Log.that("args := [" + string.Join(", ", args) + "']");
    Log.that("To use this example a 'KA09x_Base.exe' with the default uri must be running:");
    Log.that(" Base service: " + KA09x_SoapV3.SoapInterfaceConsts.DEFAULT_BASE_SERVICE_URL + "");
    Log.that(" Client service: " + KA09x_SoapV3.SoapInterfaceConsts.DEFAULT_CLIENT_SERVICE_URL + "");

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

    var r = new RemoteInterface();
    var f = new Filter(r);

    Log.that("Will connect now ...");
    r.connect(KA09x_SoapV3.SoapInterfaceConsts.DEFAULT_BASE_SERVICE_URL, KA09x_SoapV3.SoapInterfaceConsts.DEFAULT_CLIENT_SERVICE_URL, MY_VERSION, ProjectInfo.TI

    try
    {
        Log.that("Connected, will do sequence " + choice + " ...");
        switch (choice)
        {
            case (1):
                ECallSequenceBasics.doSequence(f, r);
                break;
            case (2):
                // for a more advanced scenario
                ECallSequenceAdvanced.doSequence(f, r);
                break;
            case (3):
                // a basic ERA-GLONASS sequence with an SMW200a
                ERASequenceBasic.doSequence(f, r, ERASequenceBasic.GnssSimulator.Smw200a);
                break;
            case (4):
                // a basic ERA-GLONASS sequence with an SMBV100b
                ERASequenceBasic.doSequence(f, r, ERASequenceBasic.GnssSimulator.Smbv100b);
                break;
            case (5):
                // something with ERA-GLONASS SMS
                ERASequenceSMS.doSequence(f, r);
                break;
            case (6):
                // an ERA-GLONASS sequence doing both Gsm and Wcdma
                ERASequenceWcdmaGsm.doSequence(f, r);
                break;
            case (7):
                // a simple NG-eCall sequence
                NgECallSequence.doSequence(f, r);
                break;
            case (8):
                // a simple Korean-eCall sequence
                KoreanECallSequence.doSequence(f, r);
                break;
            default:
                Log.that(Severity.ERROR, "Unknown sequence " + choice + " ...");
                break;
        }
    }
    catch (Exception e)
    {
        Log.that(e);
    }

    Log.that("Sequence done, will disconnect ...");
    if (r != null)
        r.disconnect();

    Log.that("Finished, bye!");
}
```



```

class NgECallSequence
{
    1 reference
    public static void doSequence(Filter f, RemoteInterface r)
    {
        // Reset first to get to defined state
        Common.resetBase(f, r);

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

        // Transition to configured
        Common.doConfiguration(f, r);

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

        // Wait for call
        Common.bannerPrint("Please trigger an NG-eCall on your IVS");

        // In other examples we wait for the KA09x application state to change to deal with incoming eCalls,
        // but lets do something different this time. So here we just wait for the MSD and the call info
        // in one go, and ignore anything else that happens.
        var callEvents = f.waitForAll(TimeSpan.FromMinutes(30), typeof(ImsNgEcall), typeof(RawMsdSip), typeof(DecodedMsd));

        var call = callEvents.Select(e => e as ImsNgEcall).First(e => e != null);
        var rawMsd = callEvents.Select(e => e as RawMsdSip).First(e => e != null);
        var decMsd = callEvents.Select(e => e as DecodedMsd).First(e => e != null);

        Log.that(Severity.INFO, "Received MSD via SIP " + rawMsd.source + " from call '" + call.callId + "' and it was an " + call.trigger + " " + (call.testCall ?
        Log.that(Severity.INFO, "Raw MSD was " + rawMsd.msd.Length + " bytes: " + toHex(rawMsd.msd));
        Common.print(decMsd);

        // We are done, reset the base
        Common.resetBase(f, r);
    }
}

```

## 5 Literature

- [1] Rohde & Schwarz, White paper: Voice and SMS in LTE, [1MA197](#), 2011
- [2] Rohde & Schwarz, eCall conformance and performance testing, [1MA241](#), 2018
- [3] CEN: CEN/TS 17240:2017: Intelligent transport systems - ESafety - ECall end, 2017
- [4] CEN: CEN/TS 16454:2015: Intelligent transport systems - ESafety - ECall end to, 2015
- [5] Rohde & Schwarz, User Manual: Test Software for eCall, CMW-KA094, 2015

## 6 Ordering Information

### CMW500 Radio communication tester (SUA configuration)

Designation	Type	Order No.
CMW500 Basic Assembly (Mainframe), 70MHz to 3.3GHz	R&S®CMW-PS505	1208.8921.06
Baseband Measurement Unit with 1GByte digitizer memory	R&S®CMW-S100H	1202.4701.09
Baseband interconnection, flexible link	R&S®CMW-S550N	1202.4801.15
First RF Converter (TRX), BW160 MHz	R&S®CMW-S570H	1202.5008.09
Solid State Drive (SSD)	R&S®CMW-S052S	1202.4201.20
RF Frontend, advanced functionality	R&S®CMW-S590D	1202.5108.03
CMW500 front panel with display/keypad	R&S®CMW-S600B	1201.0102.03
Extra RF Converter (TRX), BW160 MHz (hardware option) (second TRX for second cell is required)	R&S®CMW-B570H	1202.8659.09
OCXO, high stability	R&S®CMW-B690B	1202.6004.02
Signaling Unit Advanced (SUA) for GSM, WCDMA, LTE, WLAN	R&S®CMW-B500I	1208.7954.10
Option Carrier	R&S®CMW-B660H	1202.7000.09
Ethernet switch	R&S®CMW-B661H	1202.7100.09
Audio Analyzer/Generator H400B	R&S®CMW-B400B	1207.8457.02
Speech H405A Codec Board	R&S®CMW-B405A	1207.8257.02
6GHz Flat Rate, for up to 4 RF converters (TRXs) (SL)	R&S®CMW-PK364	1208.7319.02
GSM GPRS EDGE R6, basic signaling	R&S®CMW-KS200	1203.0600.02
GSM GPRS EDGE R6, advanced signaling/network emulation	R&S®CMW-KS210	1203.9759.02
WCDMA Release 99, signaling/network emulation	R&S®CMW-KS400	1203.0751.02
WCDMA Release 99, signaling/network emulation, advanced functionality	R&S®CMW-KS410	1203.9807.02

### Fading extensions for ETSI TS 103 412 (SUA configuration)

Designation	Type	Order No.
Basic Fading support, AWGN generator	R&S®CMW-KE100	1207.5506.02
GSM Fading profiles TS45.005 C.3 (excerpts)	R&S®CMW-KE200	1207.5558.02

**CMW extensions for Next Generation eCall (NGeCall) test capabilities (SUA configuration)**

Designation	Type	Order No.
Data Application Unit, retrofittable in R&S service,(hardware option)	R&S®CMW-B450I	1202.8759.10
Enabling of IP-Data interface for IPV4 (software license)	R&S®CMW-KA100	1207.2607.02
IP based measurements, in combination with technology specific IP data enabling (software license)	R&S®CMW-KM050	1203.9359.02
IMS Basic Service (software-license)	R&S®CMW-KAA20	1207.8657.02
LTE FDD Release 8, SISO, signaling/network emulation, basic functionality (software license)	R&S®CMW-KS500	1203.6108.02
LTE TDD R8, SISO, basic signaling	R&S®CMW-KS550	1204.8904.02
LTE Release 8, SISO, signaling/network emulation, advanced functionality (software license)	R&S®CMW-KS510	1203.9859.02

**CMW-KA09x PSAP software and CMWrun sequencer software**

Designation	Type	Order No.
PC based CMW applications	R&S®CMWPC	1201.0002.K90
USB Smartcard for PC based CMW applications	R&S®CMW-S089A	1202.7900.02
Enabling test software for eCall	R&S®CMW-KA094	1208.4703.02
Enabling test software for ERA GLONASS (requires CMW-KA094 license)	R&S®CMW-KA095	1208.8844.02
Enabling test software for NGeCall (standalone option - no other KA09x licenses required)	R&S®CMW-KA096	1211.2450.02
CMWrun sequencer software for eCall, ERA-GLONASS and A-GNSS (requires dedicated CMW-KA09x licenses)	R&S®CMW-KT110	1208.7431.02
CMWrun sequencer tool, NG-eCall and K-eCall (SL)	R&S®CMW-KT111	1211.3733.02

## SMBV100B Vector Signal Generator

Designation	Type	Order No.
Vector signal generator, base unit, frequency option req.	R&S®SMBV100B	1423.1003.02
Frequency range 8 kHz to 3 GHz (HW opt.)	R&S®SMBVB-B103	1423.6270.02
Real- Time Extension	R&S®SMBVB-K520	1423.7676.02
GPS (6 SVs) (SL)	R&S®SMBVB-K44	1423.7753.02
GLONASS (6 SVs) (SL)	R&S®SMBVB-K94	1423.7953.02
GALILEO (6 SVs) (SL)	R&S®SMBVB-K66	1423.7882.02
SBAS/QZSS (SL)	R&S®SMBVB-K106	1423.7982.02
GNSS channel extension to 24 channels (SL)	R&S®SMBVB-K99	1423.7976.02
GNSS Real world scenarios (SL)	R&S®SMBVB-K108	1423.8008.02

# 7 Appendix

## A The implementation of a SOAP client

The remote control examples described in chapter 4.5 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.

### A.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 24: 'host' is the main member of the SOAP implementation via DOT.NET.; eventSender of type 'IToBase32' handles the events to send to the base

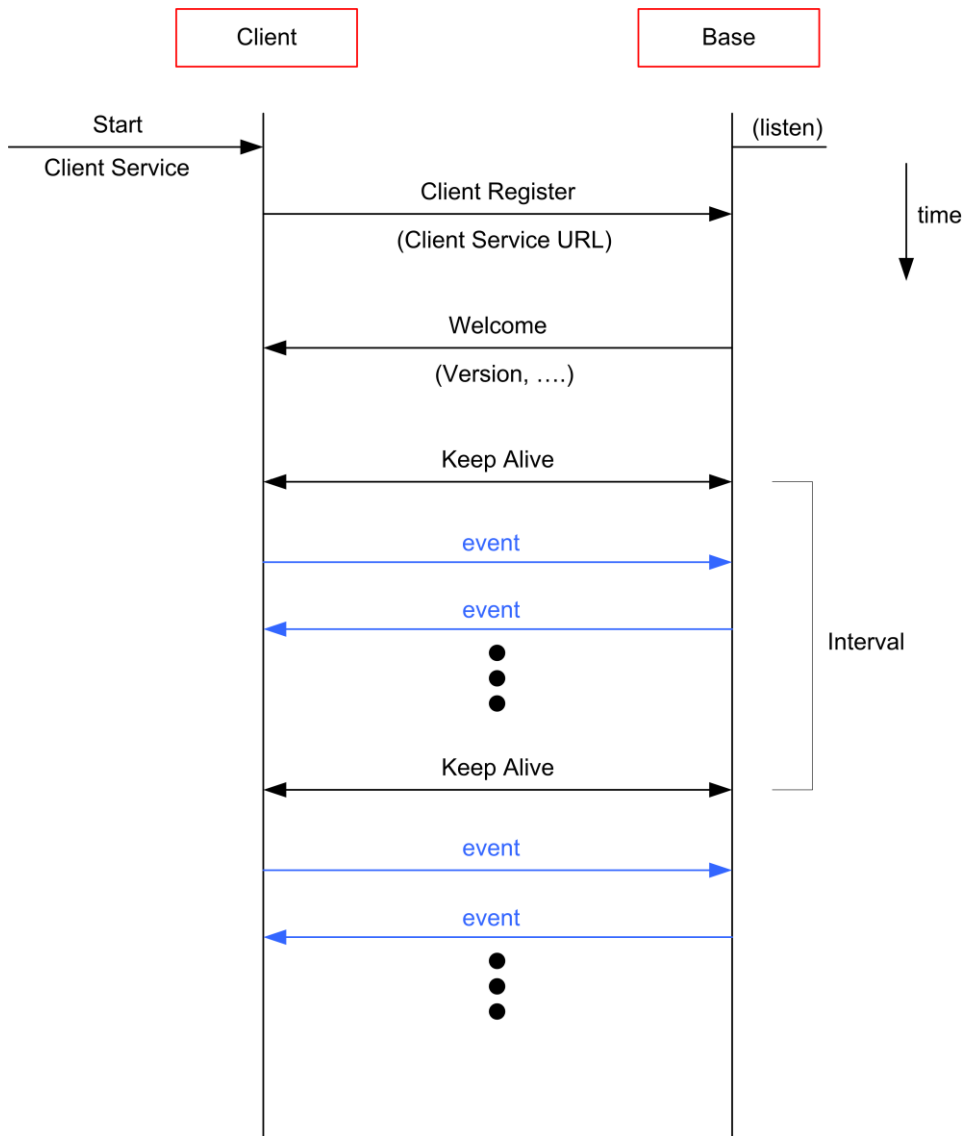


Figure 25: Basic communications in SOAP

1. The Base is in idle status and listens to incoming events
2. First the 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 already connected");

    // First clear the incoming event queue
    clearIncomingEventQueues();

    // 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 26: 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 client 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 handleIncomingEvent(EventBase eventData)
{
    try
    {
        Console.WriteLine("Rec: " + eventData.GetType() + " ");
        // Every event we receive, we just put into the queue
        incomingEventQueue.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 27: Incoming events are just added to the queue

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

## A.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 processed:

```
const int MAX_BACKLOG_SIZE = 100;

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

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)
{
    return waitForAll(timeout, eventTypeToWaitFor)[0];
}
```



```

public EventBase[] waitForAll(TimeSpan timeout, params Type[] eventTypesToWaitFor)
{
    if (eventTypesToWaitFor == null || eventTypesToWaitFor.Length == 0)
        return new EventBase[0];

    sortSpecificToGeneric(eventTypesToWaitFor);

    var matchesFromBacklog = allFromBacklog(out eventTypesToWaitFor, eventTypesToWaitFor);
    var matches = new List<EventBase>(matchesFromBacklog);
    var unmatchedTypes = eventTypesToWaitFor.ToList();

    while (unmatchedTypes.Count > 0)
    {
        if (isAnyEventTypeInteresting(unmatchedTypes))
            Console.WriteLine("waitForAll([" + string.Join(", ", unmatchedTypes.Select(t => t.Name)) + "] ...)");

        var newEvent = remote.receiveNextEvent(timeout);
        var bestMatch = unmatchedTypes.FirstOrDefault(t => t.IsAssignableFrom(newEvent.GetType()));

        if (bestMatch != null)
        {
            // that was one of the events we were waiting for
            matches.Add(newEvent);
            unmatchedTypes.Remove(bestMatch);
        }
        else
        {
            // we got an event, but it was not among the types we are waiting for ...
            // ... so put into backlog for later processing
            appendToBacklog(newEvent);
        }
    }

    return matches.ToArray();
}

```

Figure 28: 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 29: The generic wrapper 'waitFor'

## B Glossary

Abbreviation	Description
AL-ACK	Acknowledgement on the application layer
BPPM	Bipolar pulse position modulation
CRC	Cyclic redundancy check
CSCF	Call session control function
DL	Downlink
eCall	Emergency call
GLONASS	Globalnaja nawigazionnaja sputnikowaja sistema (engl.: Global navigation satellite system)
GNSS	Global navigation satellite system
GPS	Global positioning system
GSM	Global system for mobile communications
HSS	Home subscriber server
IMS	IP multimedia subsystem
IP	Internet protocol
IP-CAN	IP connectivity access network
IVS	In-vehicle system
LL-ACK	ACKnowledgement on the link-layer
LTE	Long term evolution
MSD	Minimum set of data
NG eCall	Next generation eCall
PSAP	Public safety answering point
RTCP	Real-time control protocol
RTP	Real-time transport protocol
RV	Redundancy version
SDP	Session description protocol
SIP	Session initiation protocol
TCP	Transmission control protocol
UDP	User datagram protocol
UE	User equipment
UL	Uplink
VIN	Vehicle identification number
VoLTE	Voice over LTE
WCDMA	Wideband code division multiple access