ERA Glonass Service Manager Library for Open AT Framework

Development Guide

SIERRA WIRELESS

4114918
5.0
February 13, 2015

Important Notice

Due to the nature of wireless communications, transmission and reception of data can never be guaranteed. Data may be delayed, corrupted (i.e., have errors) or be totally lost. Although significant delays or losses of data are rare when wireless devices such as the Sierra Wireless modem are used in a normal manner with a well-constructed network, the Sierra Wireless modem should not be used in situations where failure to transmit or receive data could result in damage of any kind to the user or any other party, including but not limited to personal injury, death, or loss of property. Sierra Wireless accepts no responsibility for damages of any kind resulting from delays or errors in data transmitted or received using the Sierra Wireless modem, or for failure of the Sierra Wireless modem to transmit or receive such data.

Safety and Hazards

Do not operate the Sierra Wireless modem in areas where cellular modems are not advised without proper device certifications. These areas include environments where cellular radio can interfere such as explosive atmospheres, medical equipment, or any other equipment which may be susceptible to any form of radio interference. The Sierra Wireless modem can transmit signals that could interfere with this equipment. Do not operate the Sierra Wireless modem in any aircraft, whether the aircraft is on the ground or in flight. In aircraft, the Sierra Wireless modem **MUST BE POWERED OFF**. When operating, the Sierra Wireless modem can transmit signals that could interfere with various onboard systems.

Note: *Some airlines may permit the use of cellular phones while the aircraft is on the ground and the door is open. Sierra Wireless modems may be used at this time.*

The driver or operator of any vehicle should not operate the Sierra Wireless modem while in control of a vehicle. Doing so will detract from the driver or operator's control and operation of that vehicle. In some states and provinces, operating such communications devices while in control of a vehicle is an offence.

Limitations of Liability

This manual is provided "as is". Sierra Wireless makes no warranties of any kind, either expressed or implied, including any implied warranties of merchantability, fitness for a particular purpose, or noninfringement. The recipient of the manual shall endorse all risks arising from its use.

The information in this manual is subject to change without notice and does not represent a commitment on the part of Sierra Wireless. SIERRA WIRELESS AND ITS AFFILIATES SPECIFICALLY DISCLAIM LIABILITY FOR ANY AND ALL DIRECT, INDIRECT, SPECIAL, GENERAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES INCLUDING, BUT NOT LIMITED TO, LOSS OF PROFITS OR REVENUE OR ANTICIPATED PROFITS OR REVENUE ARISING OUT OF THE USE OR INABILITY TO USE ANY SIERRA WIRELESS PRODUCT, EVEN IF SIERRA WIRELESS AND/OR ITS AFFILIATES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR THEY ARE FORESEEABLE OR FOR CLAIMS BY ANY THIRD PARTY.

Notwithstanding the foregoing, in no event shall Sierra Wireless and/or its affiliates aggregate liability arising under or in connection with the Sierra Wireless product, regardless of the number of events, occurrences, or claims giving rise to liability, be in excess of the price paid by the purchaser for the Sierra Wireless product.

Customer understands that Sierra Wireless is not providing cellular or GPS (including A-GPS) services. These services are provided by a third party and should be purchased directly by the Customer.

SPECIFIC DISCLAIMERS OF LIABILITY: CUSTOMER RECOGNIZES AND ACKNOWLEDGES SIERRA WIRELESS IS NOT RESPONSIBLE FOR AND SHALL NOT BE HELD LIABLE FOR ANY DEFECT OR DEFICIENCY OF ANY KIND OF CELLULAR OR GPS (INCLUDING A-GPS) SERVICES.

Patents

This product may contain technology developed by or for Sierra Wireless Inc.

This product includes technology licensed from QUALCOMM®.

This product is manufactured or sold by Sierra Wireless Inc. or its affiliates under one or more patents licensed from InterDigital Group and MMP Portfolio Licensing.

Copyright

Trademarks

Sierra Wireless®, AirPrime®, AirLink®, AirVantage®, WISMO®, ALEOS®, and the Sierra Wireless and Open AT logos are registered trademarks of Sierra Wireless, Inc. or one of its subsidiaries.

Watcher® is a registered trademark of Netgear, Inc., used under license.

Windows® and Windows Vista® are registered trademarks of Microsoft Corporation.

Macintosh® and Mac OS X® are registered trademarks of Apple Inc., registered in the U.S. and other countries.

QUALCOMM® is a registered trademark of QUALCOMM Incorporated. Used under license.

Other trademarks are the property of their respective owners.

Contact Information

Sales Desk:	Phone:	1-604-232-1488
	Hours:	8:00 AM to 5:00 PM Pacific Time
	Contact:	http://www.sierrawireless.com/sales
Post:	Sierra Wireless 13811 Wireless Way Richmond, BC Canada V6V 3A4	
Technical Support:	support@sierrawireless.com	
RMA Support:	repairs@sierrawireless.com	
Fax:	1-604-231-1109	
Web:	http://www.sierrawireless.com/	

Consult our website for up-to-date product descriptions, documentation, application notes, firmware upgrades, troubleshooting tips, and press releases: www.sierrawireless.com

Document History

Version	Date	Updates
1.0	December 16, 2013	Creation
2.0	July 17, 2014	Added: ERAG_CERR_PARAM in erag_error_e as an error code for bad parameter. Call duration T2 timer event ERAG_EVENT_TIMER_EXPIRED_T2_CALL_DURATION. Support for redial if call is drop in state SENDING_MSD and VOICE_CALL. Add MSD V2 compliance with otherStorage parameter Updated: Legal boilerplate content. Transition in state diagram CALL_CONNECTED to go from Idle to VoiceCall. struct erag_msd_dynamic_param_t member passengerNumber from u16 to u8 to reflect the limits 0..255. Timer T5 and T6 default values.
3.0	November 9, 2014	Added: erag_callCancelRegardless() erag_setTimeBetweenDialAttempts erag_getTimeBetweenDialAttempts Redial Mechanism chapter Clarified MSD value ranges.
4.0	November 26, 2014	Aligned ECALL_DIAL_DURATION with ERA GLONASS Specification.
5.0	February 13, 2015	Updated legal boilerplate content. Updated figure and note in Section 4.2. Updated erag_setMsd() entries in Table 1.

Contents

1. Introduction

The following subsections present introductory information regarding the ERA Glonass Service Manager Library.

1.1. Overview

The purpose of this document is to provide Sierra Wireless customers with a full description of the APIs associated with the ERA Glonass Service Manager Library.

1.2. Related Documents

[1] Open AT Framework AT Commands Interface Guide for Firmware
4111703; 4111843

[2] Open AT Framework OS ADL User Guide
4111704; 4111844

[3] ASN1 for eCall In-Band Modem
4114948

[4] eCall In-Band Modem Library - Development Guide
41123560

1.3. Abbreviations

Abbreviation	Definition
ERA	Emergency Roadside Assistance
GLONASS	Global Orbiting Navigation Satellite System; Russia
GNSS	Global Navigation Satellite System
GPS	Global Positioning System; USA
HL-ACK	High Level Acknowledgment
HW	Hardware
ID	Identifier
IVS	In Vehicle System
LL-ACK	Low Level Acknowledgment
mas	milliarcsecond
MSD	Minimum Set of Data
NV	Non-Volatile
OS	Operating System
PSAP	Public-Safety Answering Point
RTC	Real Time Clock
UTC	Coordinated Universal Time
VIN	Vehicle Identification Number

1.4. Glossary

1.4.1. Embedded Module

Sierra Wireless' AirPrime Intelligent Embedded Module running OS, on which ERA Glonass Service Manager is executed.

2. System Overview

The system consists of the In Vehicle System (IVS), the ERA GLONASS Service Manager, In-Band Modem Plugin, Firmware and the Location Plugin.

There are two basic different hardware setups with either the GNSS is on the Modem side or on the Host side as displayed below. Since the Host side in both cases has a need to periodically know the position, our solution will be identic for both setups.

Please note that we will require the Host to initialize, enable and the handle the GNSS in both cases, for the internal GNSS through the interface for Location plugin.

The host is also responsible for making sure that the SIM card is in a good state with PIN code set.

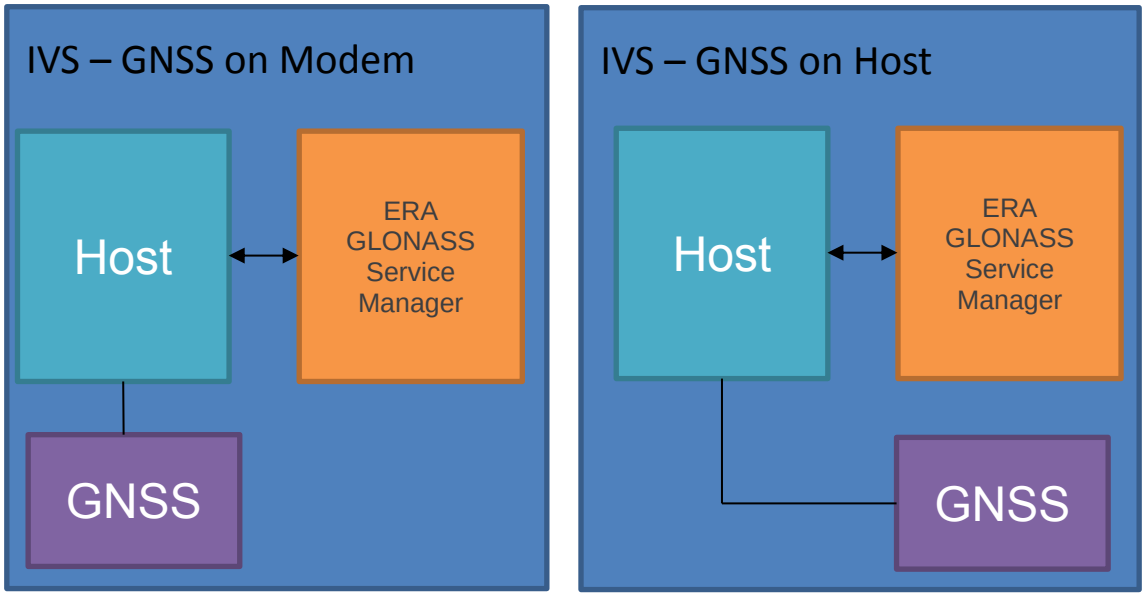


Figure 1. ERA Glonass Service Manager Library Architecture

3. Library Architecture

The following figure shows the architecture of the ERA Glonass Service Manager Library. Note that regardless if the GNSS is handled by the Host application.

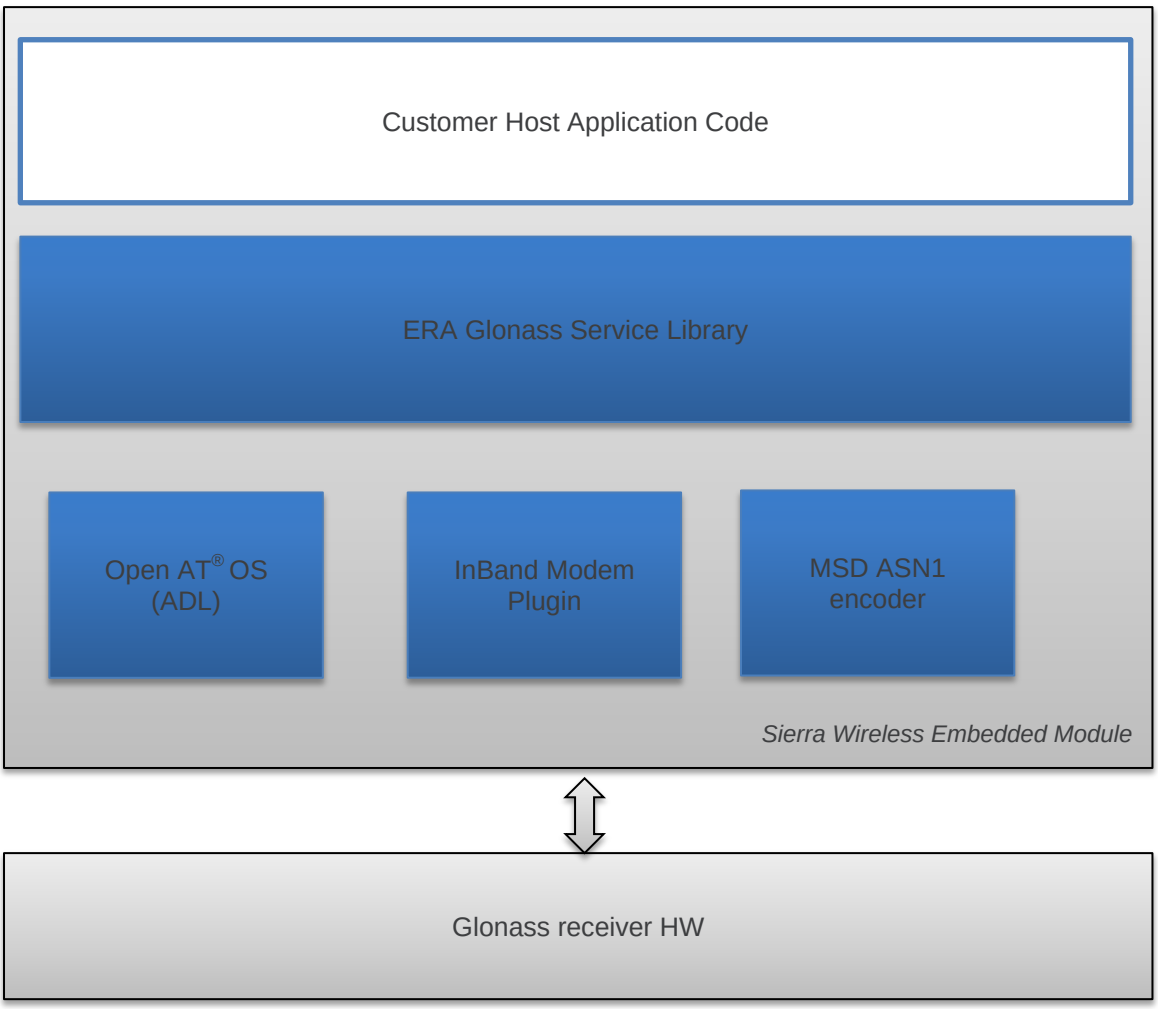


Figure 2. ERA Glonass Service Manager Library Architecture

4. Library Description

The following subsections provide high level description of the ERA Glonass Service Manager Library features and requirements.

4.1. Feature Description

The ERA Glonass Service Manager Library provides functionality to

- Setup and manage sending MSD to PSAP over a Voice Call.
- Manage behavior in accordance to ERA Glonass.
- Automatic configuration and management of the underlying In-Band Modem

4.2. Library and APIs

The ERA Glonass Service Manager Library APIs provides a way to setup and manage MSD sending over a voice channel in accordance to the ERA Glonass specification. The following figure shows the Library state machine.

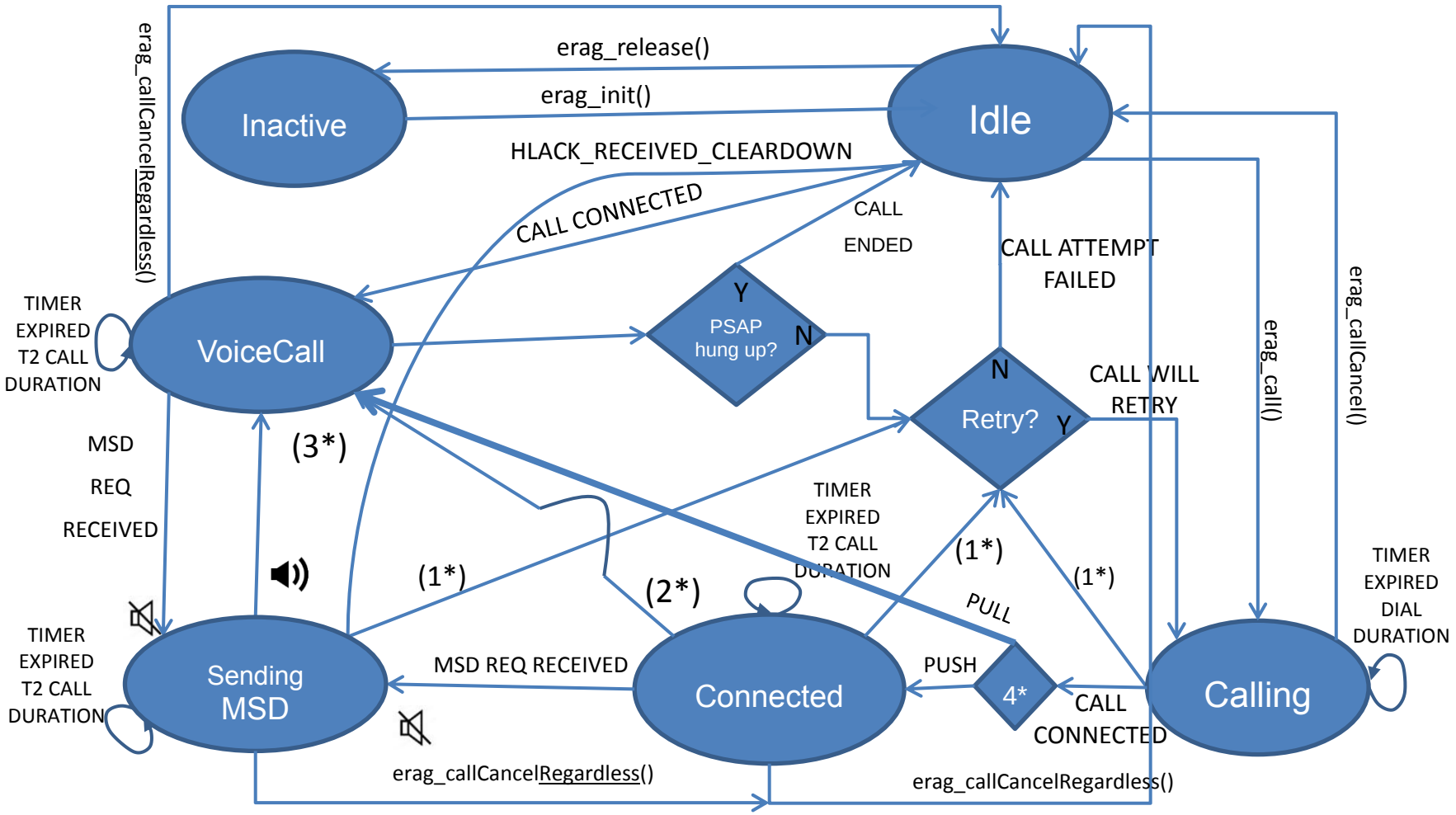


Figure 3. State Machine for ERA Glonass Service Manager

(*) Note:.
1. The CALL_ENDED event in this case is not given in call back, instead CALL_ATTEMPT_FAILED or CALL_WILL_RETRY will be given.
2. The following events will trigger this transition

ERAG_EVENT_INBM_MSD_REQ_NOT_RECEIVED

ERAG_EVENT_INBM_ABORT_RECEIVED

3. The following events will trigger this transition

ERAG_EVENT_INBM_HLACK_RECEIVED

ERAG_EVENT_INBM_HLACK_NOT_RECEIVED

ERAG_EVENT_INBM_LLACK_NOT_RECEIVED

ERAG_EVENT_INBM_ABORT_RECEIVED

4. Was the current call/redial was started in Push or Pull mode?

5. ERA Glonass Control API

The Control APIs allow the control and configuration of ERA Glonass Service Manager.

5.1. Required Header File

The header file for the ERA Glonass Service Manager to use is *erag.h*.

5.2. Types

5.2.1. The erag_states_e type

The erag_states_e enum corresponds to the states described in the State Machine chapter with the following implementation.

5.2.1.1. Definition

```
typedef enum erag_states_e
{
  ERAG_STATE_INACTIVE          = 0,
  ERAG_STATE_IDLE              = 1,
  ERAG_STATE_CALLING           = 2,
  ERAG_STATE_CONNECTED         = 3,
  ERAG_STATE_SENDING_MSD       = 4,
  ERAG_STATE_VOICE_CALL        = 5
}erag_states_e;
```

5.2.1.2. Values

Value	Definition
ERAG_STATE_INACTIVE	Enum value for the INACTIVE state. This is the start state for the automaton.
ERAG_STATE_IDLE	Enum value for the IDLE state.
ERAG_STATE_CALLING	Enum value for the CALLING state. The call is not yet connected at this time.
ERAG_STATE_CONNECTED	Enum value for the CONNECTED state. The call is connected.
ERAG_STATE_SENDING_MSD	Enum value for the SENDING MSD state. The PSAP requested a MSD.
ERAG_STATE_VOICE_CALL	Enum value for the VOICE CALL state. Sending of MSD is done.

5.2.2. The erag_cb_event_type_e type

The erag_cb_event_type_e enum corresponds to the call back events with the following implementation. This value is used to know how to interpret the data part of the erag_event_cb_t.

5.2.2.1. Definition

```
typedef enum erag_cb_event_type_e
{
  ERAG_CB_EVENT_INFO,
  ERAG_CB_EVENT_ERROR,
  ERAG_CB_EVENT_MSD
}erag_cb_event_type_e;
```

5.2.2.2. Values

Value	Definition
ERAG_CB_EVENT_INFO	The current call back event is of informative type.
ERAG_CB_EVENT_ERROR	The current call back event is of error type.
ERAG_CB_EVENT_MSD	The MSD call back event contains the current MSD encoded in ASN.1. It is received the MSD is successfully updated via erag_setMsd or erag_call.

5.2.3. The erag_info_events_e type

When the callback receives an event with the *id* field equal to ERAG_CB_EVENT_INFO, then the type of event is defined by this enum in the *info* part of the union in erag_event_cb_t.

The *erag_info_events_e* corresponds to type of informational event that was passed the callback function. Each event is visible in the State Machine chapter.

Please note that the InBand Modem events correspond are mirrored exactly below.

5.2.3.1. Definition

```
typedef enum erag_info_events_e
{
  /* InBand Modem Events */
  ERAG_EVENT_INBM_MSD_REQ_RECEIVED        = 0,
  ERAG_EVENT_INBM_MSD_REQ_NOT_RECEIVED,
  ERAG_EVENT_INBM_LLACK_RECEIVED,
  ERAG_EVENT_INBM_HLACK_RECEIVED,
  ERAG_EVENT_INBM_LLACK_NOT_RECEIVED,
  ERAG_EVENT_INBM_HLACK_NOT_RECEIVED,
  ERAG_EVENT_INBM_ABORT_RECEIVED,
  ERAG_EVENT_INBM_HLACK_RECEIVED_CLEARDOWN,

  /* CALL Handling events */
  ERAG_EVENT_CALL_CONNECTED    = 20,
  ERAG_EVENT_CALL_NOT_CONNECTED,
  ERAG_EVENT_CALL_ENDED,

  /* retry attempt will result in either of the two events*/
  ERAG_EVENT_CALL_WILL_RETRY,
  ERAG_EVENT_CALL_ATTEMPT_FAILED,

  ERAG_EVENT_CALL_INCOMING,

  /* COMMANDS events  */
  ERAG_EVENT_ERAGINIT         = 40,
  ERAG_EVENT_ERAGRELEASE,
  ERAG_EVENT_ERAGCALL,
  ERAG_EVENT_ERAGCALLCANCEL,
  ERAG_EVENT_ERAGSETMSD,
  ERAG_EVENT_ERAGSETERA,
```

```
  /* Timer events  */
  ERAG_EVENT_TIMER_EXPIRED_DIAL_DURATION = 50,
  ERAG_EVENT_TIMER_EXPIRED_T2_CALL_DURATION,
}erag_info_events_e;
```

5.2.3.2. Values

Value	Definition
ERAG_EVENT_INBM_MSD_REQ_RECEIVED	SEND MSD request received from PSAP server
ERAG_EVENT_INBM_MSD_REQ_NOT_RECEIVED	SEND MSD request not received from PSAP server within 2 seconds from the first SEND MSD REQUEST message sending
ERAG_EVENT_INBM_LLACK_RECEIVED	LLACK received from PSAP server for the MSD transmission
ERAG_EVENT_INBM_HLACK_RECEIVED	LLACK not received within 20 seconds from MSD transmission
ERAG_EVENT_INBM_LLACK_NOT_RECEIVED	HLACK received from PSAP server for the MSD transmission
ERAG_EVENT_INBM_HLACK_NOT_RECEIVED	HLACK not received within 5 seconds from the time LLACK received
ERAG_EVENT_INBM_ABORT_RECEIVED	Abort during data transmission
ERAG_EVENT_INBM_HLACK_RECEIVED_CLEARDOWN	Cleardown indication received from PSAP after successful MSD transfer
ERAG_EVENT_CALL_CONNECTED	Call is now connected.
ERAG_EVENT_CALL_NOT_CONNECTED	Call was not connected. The service manager will hang up the line to prepare for a potential retry.
ERAG_EVENT_CALL_ENDED	Call was hung up. Possible retry possible. Please note the (*)in the statemachine diagram. In retry scenario, this event is not received in the Callback, instead ERAG_EVENT_CALL_WILL_RETRY or ERAG_EVENT_CALL_ATTEMPT_FAILED is received.
ERAG_EVENT_CALL_WILL_RETRY	The just previously ended call will be retried.
ERAG_EVENT_CALL_ATTEMPT_FAILED	The just previously ended call will not be retried. End of retry attempts reached
ERAG_EVENT_CALL_INCOMING	Incoming call. It is up to the Host Application to answer the incoming call.
ERAG_EVENT_ERAGINIT	erag_init() was executed.

Value	Definition
ERAG_EVENT_ERAGRELEASE	erag_release() was executed.
ERAG_EVENT_ERAGCALL	erag_call() was executed.
ERAG_EVENT_ERAGCALLCANCEL	erag_callCancel() or erag_callCancelRegardless() was executed.
ERAG_EVENT_ERAGSETMSD	erag_setMsd() was executed.
ERAG_EVENT_ERAGSETERA	erag_setEra() was executed.
ERAG_EVENT_TIMER_EXPIRED_DIAL_DURATION	Dial Duration timer has expired. Current call will be hung up. This will cause the event ERAG_EVENT_CALL_ENDED to follow which will trigger eventual redial.
ERAG_EVENT_TIMER_EXPIRED_T2_CALL_DURATION	The Call duration timer T2 has expired. Current call will be hung up. See ccft_t2 in structure erag_settings_t in 5.2.8.1.

5.2.4. The erag_error_e type

The *erag_error_e* provides an error code for errors that might occur and tries to provide a hint to what the error is.

5.2.4.1. Definition

```
typedef enum erag_error_e {
  ERAG_OK = OK,
  ERAG_CERR_BAD_STATE        = -1,
  ERAG_CERR_UNKNOWN_STATE    = -2,
  ERAG_CERR_UNHANDLED_EVENT  = -3,
  ERAG_CERR_MSD_VALUE_RANGE  = -4,
  ERAG_CERR_NULL_VALUE       = -5,
  ERAG_CERR_INBND_MDM_ERROR  = -6,
  ERAG_CERR_UNKNOWN_ERROR    = -7,
  ERAG_CERR_PARAM            = -8,
  ERAG_CERR_LAST
} erag_error_e ;
```

5.2.4.2. Values

Value	Definition
ERAG_OK	The result of the calling function was OK
ERAG_CERR_BAD_STATE	Function was called when the state when the function cannot be executed.
ERAG_CERR_UNKNOWN_STATE	State machine has reached an unknown state.
ERAG_CERR_UNHANDLED_EVENT	Unhandled event in this state.
ERAG_CERR_MSD_VALUE_RANGE	Updating the MSD values failed. While trying to encode the MSD with the new values, the ASN1 encoding failed. New values are not taken into account.
ERAG_CERR_NULL_VALUE	Null value found when NULL is not allowed.
ERAG_CERR_INBND_MDM_ERROR	Error from InBand Modem.
ERAG_CERR_UNKNOWN_ERROR	Unspecified error occurred.
ERAG_CERR_PARAM	Bad parameter value. If not a value for MSD then see ERAG_CERR_MSD_VALUE_RANGE or NULL, then see ERAG_CERR_NULL_VALUE
ERAG_CERR_LAST	Last enum member value

5.2.5. The erag_error_event_e type

The *erag_error_event_e* is the type of error for an error event of type *ERAG_CB_EVENT_ERROR*.

5.2.5.1. Definition

```
typedef enum erag_error_event_e {
  ERAG_EV_ERROR_NO_SIM,
  ERAG_EV_ERROR_PIN_NOT_READY,
  ERAG_EV_ERROR_NO_CARRIER,
  ERAG_EV_ERROR_SETUP_ERROR,
  ERAG_EV_ERROR_NO_ANSWER,
  ERAG_EV_ERROR_BUSY
} erag_error_event_e ;
```

5.2.5.2. Values

Value	Definition
ERAG_EV_ERROR_NO_SIM	Call failed because there is no SIM detected
ERAG_EV_ERROR_PIN_NOT_READY	Call failed because SIM is not ready.
ERAG_EV_ERROR_NO_CARRIER	Call failed because there is no no Carrier.
ERAG_EV_ERROR_SETUP_ERROR	Call failed because there was a Setup Error during the call.
ERAG_EV_ERROR_NO_ANSWER	Call failed because there was no answer
ERAG_EV_ERROR_BUSY	Call failed because it's busy

5.2.6. The erag_msd_static_param_t type

The *erag_msd_static_param_t* provides structure contains all the parts of the MSD that are static will not change.

5.2.6.1. Definition

```
typedef struct erag_msd_static_param_t
{
    e_VehicleType    vehicleType;
    char    isowmi[3];
    char    isovds[6];
    char    isovisModelyear[1];
    char    isovisSeqPlant[7];
    struct {
        bool gasolineTankPresent;
        bool dieselTankPresent;
        bool compressedNaturalGas;
        bool liquidPropaneGas;
        bool electricEnergyStorage;
        bool hydrogenStorage;
        bool otherStorage;
   }vehiclePropulsionStorageType;
}erag_msd_static_param_t;
```

5.2.6.2. Values

Value	Definition
vehicleType	The type of the vehicle. Please refer to [4] for values since the type is from that interface.
Isowmi	A string containing the isowmi.
Isovds	A string containing the isovds
isovisModelyear	A string containing the isovisModelyear
isovisSeqPlant	A string containing the isovisSeqPlant
vehiclePropulsionStorageType	A structure containing the type of propulsion storages that are present.
vehiclePropulsionStorageType. gasolineTankPresent	True if Gasoline tank present.
vehiclePropulsionStorageType. dieselTankPresent	True if Diesel tank present.
vehiclePropulsionStorageType. compressedNaturalGas	True if Natural Gas tank present.
vehiclePropulsionStorageType. liquidPropaneGas	True if Propane Gas tank present.
vehiclePropulsionStorageType. electricEnergyStorage	True if Electric Energy Storage present.
vehiclePropulsionStorageType. hydrogenStorage	True if Hydrogen tank present.

Value	Definition
vehiclePropulsionStorageType. otherStorage	True if other storage present. Only for MSD V2

Please note that the above mentioned fields accepts only a limited set of characters. Here are the possible values as stated by ISO 3779 standard:

```
-- isowmi: World Manufacturer Index (WMI)
-- isovds: Vehicle Type Descriptor (VDS)
-- Vehicle Identifier Section (VIS) consisting of
-- isovisModelyear: Modelyear from Vehicle Identifier Section (VIS)
-- isovisSeqPlant: Plant code + sequential number
-- from Vehicle Identifier Section (VIS)
VIN ::= SEQUENCE {
isowmi PrintableString (SIZE(3))
(FROM("A".."H"|"J".."N"|"P"|"R".."Z"|"0".."9")),
isovds PrintableString (SIZE(6))
(FROM("A".."H"|"J".."N"|"P"|"R".."Z"|"0".."9")),
isovisModelyear PrintableString (SIZE(1))
(FROM("A".."H"|"J".."N"|"P"|"R".."Z"|"0".."9")),
isovisSeqPlant PrintableString (SIZE(7))
(FROM("A".."H"|"J".."N"|"P"|"R".."Z"|"0".."9"))

}
```

5.2.7. The erag_msd_dynamic_param_t type

The *erag_msd_dynamic_param_t* provides a structure that contains all the parts of the MSD that are dynamic.

5.2.7.1. Definition

```
typedef struct erag_msd_dynamic_param_t
{
   u8  ID;
   u8  msdID;
   struct {
      bool automaticActivation;
      bool testCall;
      bool positionCanBeTrusted;
   } control;
   u32  timestamp;
   s32  positionLatitude;
   s32  positionLongitude;
   u8   vehicleDirection;
   bool present_posN1;
   s32  positionLatitudeN1;
   s32  positionLongitudeN1;
   bool present_posN2;
   s32  positionLatitudeN2;
   s32  positionLongitudeN2;
   bool present_passNbr;
   u8  passengerNumber;
   bool present_OID;
   u32  OID_len;
   u8*  OID_p;
   u32  data_len;
   u8*  data_p;
} erag_msd_dynamic_param_t;
```

5.2.7.2. Values

Value	Definition	Type	Min	Max	Values with special meaning
ID	Parameter for id value	u8	1	2	-
msdID;	This is the MSD Id. The Host must set this accordingly.	u8	0	255	-
automaticActivation	TRUE if automatic activation. FALSE if manual.	bool	-	-	-
testCall	TRUE if test call. FALSE if emergency call.	bool	-	-	-
positionCanBeTrusted	TRUE if the position given can be trusted.	bool	-	-	-
timestamp	Parameter for timestamp value. Seconds since 1970:00:00:00	u32	0	4294967295 = 0xFFFFFFFF	0: failed to get time
positionLatitude	The latitude of the current position in unit *mas*.	s32	-324000000 (-90°)	+324000000 (+90°)	0x7FFFFFFF = 2147483647: latitude is invalid or unknown
positionLongitude	The longitude of the current position in unit *mas*	s32	-648000000 (-180°)	+648000000 (+180°)	0x7FFFFFFF = 2147483647: longitude is invalid or unknown
vehicleDirection	The direction of the vehicle. Degrees from magnetic North, 2-deg steps.	u8	0	179	0xFF: direction is invalid or unknown
present_posN1	If the optional parameter Position N1 is present.	bool	-	-	-
positionLatitudeN1	The previous positions latitude delta. Optional parameter. Unit is 100-mas steps (latitude delta)	s32	-512 (51.2"S)	+511 (51.1"N)	-
positionLongitudeN1	The previous positions longitude delta. Optional parameter Unit is 100-mas steps (longitude delta).	s32	-648000000 (-180°)	+648000000 (+180°)	-
present_posN2	If the optional parameter Position N2 is present.	bool			
positionLatitudeN2	The position N1 previous positions latitude delta. Optional parameter Unit is 100-mas steps (latitude delta).	s32	-512 (51.2"S)	+511 (51.1"N)	-

Value	Definition	Type	Min	Max	Values with special meaning
positionLongitudeN2	The position N1 previous positions longitude delta. Optional parameter. Unit is 100-mas steps (longitude delta).	s32	-648000000 (-180°)	+648000000 (+180°)	-
present_passNbr	If TRUE, then passengerNumber variable will be used.	bool	-	-	-
passengerNumber	The number of passengers in the vehicle. Optional parameter	u8	0	255	255: invalid or unknown
present_OID	If TRUE, then the following OID* and data* parameters should be used. They contain data which is already ASN1 encoded. Optional parameter.	bool	-	-	-
OID_len	The length of the OID_p. Optional parameter.	u32	-	-	-
OID_p	The OID data part. Optional parameter.	u8*	-	-	-
data_len	The length of the data_p. Optional parameter.	u32	-	-	-
data_p	The data part of the OID. Optional parameter.	u8*	-	-	-

5.2.8. The erag_settings_t type

The *erag_settings_t* provides structure which contains all the ERA Glonass parameters that are available . Note that all time values are given in seconds.

5.2.8.1. Definition

```
typedef struct erag_settings_t{
   u32 ccft_t2;
   u32 invitation_signal_duration_t3;
   u32 send_msg_period_t5;
   u32 al_ack_period_t6;
   u32 msd_max_transmission_time_t7;
   u32 ecall_dial_duration;
   u32 ecall_auto_dial_attempts;
   u32 ecall_manual_dial_attempts;
}erag_settings_t;
```

5.2.8.2. Values

Value	Definition	Default value
ccft_t2	The value of CCFT in seconds.	3600
invitation_signal_duration_t3	The value of INVITATION_SIGNAL_ DURATION in seconds.	2
send_msg_period_t5	The value of SEND_MSG_PERIOD in seconds.	5
al_ack_period_t6	The value of AL_ACK_PERIOD in seconds.	5
msd_max_transmission_time_t7	The value of MSD_MAX_ TRANSMISSION_TIME in seconds.	20
ecall_dial_duration	The value of ECALL_DIAL_DURATION in seconds.	300
ecall_auto_dial_attempts	The value of ECALL_AUTO_DIAL_ ATTEMPTS.	10
ecall_manual_dial_attempts	The value of ECALL_MANUAL_DIAL_ATTEMPTS	10
ecall_time_between_dial_attempts	The time in seconds between the end of a failed call and the next redial attempt.	1

5.2.9. The erag_event_cb_t type

The *erag_event_cb_t* provides structure that contains the data for all callback event data that is received through the callback function.
Please note that there is a id number that specifies which kind of type of event data it is, and then a union of which the type member should be accessed depending on the this id.

5.2.9.1. Definition

```
typedef struct erag_event_cb_t {
  erag_cb_event_type_e id;

  union erag_cbevent_content_t {
    struct msd{ /* if id == ERAG_CB_EVENT_MSD use this */
      u32   bin_length;
      /* ASN1 binary encoded*/
      u8* MSD_asn1_encoded;
    } msd;

    struct info{/* if id == ERAG_CB_EVENT_INFO use this */
      erag_info_events_e infocode; /* @see erag_events_types_e */
      erag_states_e state_before;/* The state before the event arrived*/
      erag_states_e state_after; /* The state after the event was processed*/
    } info;

    struct error{/* if id == ERAG_CB_EVENT_ERROR use this */
       erag_error_event_e errcode;
    } error;

  } content;
} erag_event_cb_t;
```

5.2.9.2. Values

Value	Definition
id	The type of event Information/ Error/ MSD. The union after contains three corresponding members.
content.msd	When id == ERAG_CB_EVENT_MSD, this union member should be accessed.
content.msd. bin_length	The length of the data in the pointer MSD_asn1_encoded.
content.msd.MSD_asn1_encoded	The ASN1 binary encoded MSD of length bin_length

Value	Definition
content.info	When id == ERAG_CB_EVENT_INFO, this union member should be accessed.
content.info.infocode	The type of event. See the erag_info_events_e.
content.info. state_before	The state before this event was processed.
content.info. state_after	The state after this event was processed.
content.error	When id == ERAG_CB_EVENT_ERROR, this union member data should be accessed.
content.error.errcode	The error code. Please see erag_error_event_e.

5.2.10. Event handler erag_eventHandler_f type

The *erag_eventHandler_f* is the callback function type used to accept all events from the ERA Glonass Service Manager.

5.2.10.1. Definition

```
typedef void (*erag_eventHandler_f)( erag_event_cb_t *ev, void * ctx );
```

5.2.10.2. Parameter

Value	Definition
ev	This event contains the data related to the event.
ctx	This parameter has the same value as the ctx parameter given in erag_init()

5.3. Functions

5.3.1. The erag_init function

The *erag_init* initializes the services in the Service Manager.
The parameters provide the callback, the static data of the vehicle; these will be used later for the MSD encoding and a possibility to pass a ctx pointer.

5.3.1.1. Definition

```
erag_error_e  erag_init ( erag_msd_static_param_t * msd_static,
        erag_eventHandler_f event_handler_cb ,
        void * ctx )
```

5.3.1.2. Parameter

Value	Definition
msd_static	A pointer to a structure that contains the static parts of the MSD.
event_handler_cb	This is the callback function that will receive the events.
ctx	This void pointer is given back in the callback handler. It is up to the client how this parameter is used. May be set to NULL.

5.3.1.3. Return value

The function returns

- ERAG_OK on success
- In case of error a negative error code as described below.

Value	Definition
ERAG_CERR_BAD_STATE	Function was not called in the state Inactive.
ERAG_CERR_NULL_VALUE	NULL pointer error.

5.3.2. The erag_call function

The *erag_call* function assembles the MSD with the last missing parameters and then calls the number provided.

An event of type ERAG_CB_EVENT_MSD will be received in the callback via the ANS.1 encoded MSD.

5.3.2.1. Definition

```
erag_error_e  erag_call(
        char *phoneNumber,
        erag_msd_dynamic_param_t * erag_msd_dynamic
        );
```

5.3.2.2. Parameter

Value	Definition
phoneNumber	A string containing the telephone number. It may also contain a category field. May not be NULL.
erag_msd_dynamic	The structure containing all the static parameters of the MSD. See erag_msd_dynamic_param_t.

5.3.2.3. Return value

The function returns

- ERAG_OK on success
- In case of error a negative error code as described below.

Value	Definition
ERAG_CERR_BAD_STATE	Function was called when the state when the function cannot be executed.
ERAG_CERR_NULL_VALUE	Null pointer was found where no Null pointer is allowed.
ERAG_CERR_MSD_VALUE_RANGE	Some values provided for the MSD encoding resulted bad encoding of ASN1.

5.3.3. The erag_callCancel function

The function *erag_callCancel* allows a call that has been requested through *erag_call* to be cancelled before it has been connected. If the call has already been connected the *erag_callCancel* will fail.

5.3.3.1. Definition

```
erag_error_e  erag_callCancel( void )
```

5.3.3.2. Parameter

No parameters.

5.3.3.3. Return value

The function returns

- ERAG_OK on success
- In case of error a negative error code as described below.

Value	Definition
ERAG_CERR_BAD_STATE	Function was called when the state when the function cannot be executed.

5.3.4. The erag_callCancelRegardless function

The function *erag_callCancelRegardless* allows a call that has been requested through *erag_call* to be cancelled regardless if it has been connected or not.

5.3.4.1. Definition

```
erag_error_e  erag_callCancelRegardless( void )
```

5.3.4.2. Parameter

No parameters.

5.3.4.3. Return value

The function returns

- ERAG_OK on success
- In case of error a negative error code as described below.

Value	Definition
ERAG_CERR_BAD_STATE	Function was called when the state when the function cannot be executed.

5.3.5. The erag_release function

The *erag_release* function is used when going from Idle to Inactive state. It disables all activities with InBand Modem and callhandling listening. It may only be called in Idle State.

5.3.5.1. Definition

```
erag_error_e  erag_release( void )
```

5.3.5.2. Parameter

No parameters.

5.3.5.3. Return value

The function returns

- ERAG_OK on success
- In case of error a negative error code as described below.

Value	Definition
ERAG_CERR_BAD_STATE	Function was called when the state when the function cannot be executed.

5.3.6. The erag_setEra function

The *erag_setEra* function enables the setting of ERA Glonass settings.
This function may only be called in IDLE state.
Please refer to the *erag_settings_t* documentation for details and default values.

5.3.6.1. Definition

```
erag_error_e  erag_setEra( erag_settings_t * pEraSettings)
```

5.3.6.2. Parameter

Value	Definition
pEraSettings	This structure contains all the settings needed for the ERA Glonass. May not be NULL.

5.3.6.3. Return value

The function returns

- ERAG_OK on success
- In case of error a negative error code as described below.

Value	Definition
ERAG_CERR_NULL_VALUE	Inparameter has NULL value.
ERAG_CERR_BAD_STATE	Function was called when the state when the function cannot be executed.

5.3.7. The erag_setMSD function

The *erag_setMSD* enables to update the dynamic values for the MSD in the same way the erag_call() does. The ANS.1 encoded MSD will be received in the callback via an event of type ERAG_CB_EVENT_MSD.

5.3.7.1. Definition

```
erag_error_e  erag_setMsd( erag_msd_dynamic_param_t * erag_msd_dynamic )
```

5.3.7.2. Parameter

Value	Definition
erag_msd_dynamic	The structure containing all the dynamic parameters of the MSD.

5.3.7.3. Return value

The function returns

- ERAG_OK on success
- In case of error a negative error code as described below.

Value	Definition
ERAG_CERR_BAD_STATE	Function was called when the state when the function cannot be executed.
ERAG_CERR_NULL_VALUE	Null pointer was found where no Null pointer is allowed.
ERAG_CERR_MSD_VALUE_RANGE	Some values provided for the MSD encoding resulted bad encoding of ASN1.

5.3.8. The erag_getState function

The function *erag_getState* returns the current state.

5.3.8.1. Definition

```
erag_states_e  erag_getState( void );
```

5.3.8.2. Parameter

No parameters.

5.3.8.3. Return value

The function returns the value as described in the enum *erag_states_e*.

5.3.9. The erag_getMSD function

This function returns the current MSD with all values. It is possible to get only one part the by letting the other parameter be NULL. Both must not be NULL.

5.3.9.1. Definition

```
erag_error_e erag_getMsd( erag_msd_static_param_t * out_msd_static ,
                          erag_msd_dynamic_param_t * out_msd_dynamic);
```

5.3.9.2. Parameter

Value	Definition
out_msd_static	A pointer to a buffer to a variable with static parameters provided by the calling function. Can be NULL if out_msd_dynamic is not.
out_msd_dynamic	A pointer to a buffer to a variable with dynamic parameters provided by the calling function. Can be NULL if out_msd_static is not.

5.3.9.3. Return Value

The function returns

- ERAG_OK on success
- In case of error a negative error code as described below.

Value	Definition
ERAG_CERR_NULL_VALUE	Both parameters were NULL.

5.3.10. The erag_getERAGSettings function

This function gets the current values of all ERA Glonass settings.

5.3.10.1. Definition

```
erag_error_e erag_getERAGSettings( erag_settings_t * pOut_erag_Settings  )
```

5.3.10.2. Parameter

Value	Definition
pOut_erag_Settings	The pointer to a buffer provided by the calling function. The result will be provided in this structure.

5.3.10.3. Return Value

The function returns

- ERAG_OK on success
- In case of error a negative error code as described below.

Value	Definition
ERAG_CERR_NULL_VALUE	Inparameter was NULL.

5.3.11. The erag_getVersion function

This function returns a string containing the ERA Glonass Library Version.

5.3.11.1. Definition

```
const signed char * erag_getVersion( void )
```

5.3.11.2. Parameter

No parameters.

5.3.11.3. Return value

The function returns a string containing the library version. The string is statically defined in the library.

5.3.13. The erag_setTimeBetweenDialAttempts function

The *erag_setTimeBetweenDialAttempts* function configures the period between the call fails, and the retry call is launched. Default value is 1 second. Note that the event ERAG_EVENT_CALL_WILL_RETRY is issued once the retry call is launched, not when the call fails. This function may only be called in IDLE state.

Please see chapter 8 about the redial mechanism.

5.3.13.1. Definition

```
erag_error_e  erag_setTimeBetweenDialAttempts( u32 time_between_dial_attempts );
```

5.3.13.2. Parameter

Value	Definition	Default Value	Min Value	Max Value
time_between_dial_attempts	Value in seconds. The period between when the call fails and the call is redialed.	1	1	3600

5.3.13.3. Return value

The function returns

- ERAG_OK on success
- In case of error a negative error code as described below.

Value	Definition
ERAG_CERR_PARAM	Inparameter is out of range.
ERAG_CERR_BAD_STATE	Function was called when the state when the function cannot be executed.

5.3.14. The erag_getTimeBetweenDialAttempts function

This function returns time in seconds between the dial attempts as described in *erag_setTimeBetweenDialAttempts()*

5.3.14.1. Definition

```
u32 erag_getTimeBetweenDialAttempts( void );
```

5.3.14.2. Parameter

No parameters.

5.3.14.3. Return value

This function returns time in seconds between the dial attempts as described in *erag_setTimeBetweenDialAttempts()*

6. Resource Usage

6.1. IRQ Usage

The ERA Glonass Service Manager uses the In-Band Modem Library.

The In-Band Modem Library internally uses both Low Level IRQ Handler and High Level IRQ Handlers. Please refer to the In-Band Modem[5] documentation for further more information. The important information to keep in mind is that both stack for Low Level IRQ Handler and High Level IRQ Handlers must be defined.

7. API Calls Requirements

7.1. Function Calls Requirements

The following table shows the prerequisites when calling functions.

'X' means the function is authorized in the corresponding state.

'-' means the function will NOT work in the corresponding state.

Table 1. ERA GLONASS commands prerequisites

Function	Inactive	Idle	Calling	Connected	Sending MSD	VoiceCall
erag_init()	X	-	-	-	-	-
erag_call()	-	X	-	-	-	-
erag_setEra()	-	X	-	-	-	-
erag_setMsd()	-	X	X	-	-	X
erag_callCancel()	-	-	X	-	-	-
erag_callCancelRegardless()	-	-	X	X	X	X
erag_release()	-	X	-	-	-	-
erag_getState	X	X	X	X	X	X
erag_getMSD	-	X	X	X	X	X
erag_getERAGSettings	-	X	X	X	X	X
erag_setTimeBetweenDialAttempts	-	X	-	-	-	-
erag_getTimeBetweenDialAttempts	-	X	X	X	X	X

Caution: *The function reentrancy is NOT allowed in the Library*

8. Redial Mechanism

8.1. Introduction

The ERA GLONASS Service Manager handles the redials in the following manner.

Retries are limited in both time and number, whichever runs out first.

ECALL_DIAL_DURATION defines the maximum time when dialing. That is to say from the call is launched, but not yet connected including all redial attempts.See 5.2.8 and the member of structure The erag_settings_t type.

The maximum number of redials that will be performed is either ECALL_AUTO_DIAL_ATTEMPS or ECALL_MANUAL_DIAL_ATTEMPS. See 5.2.8 and the member of structure The erag_settings_t type.

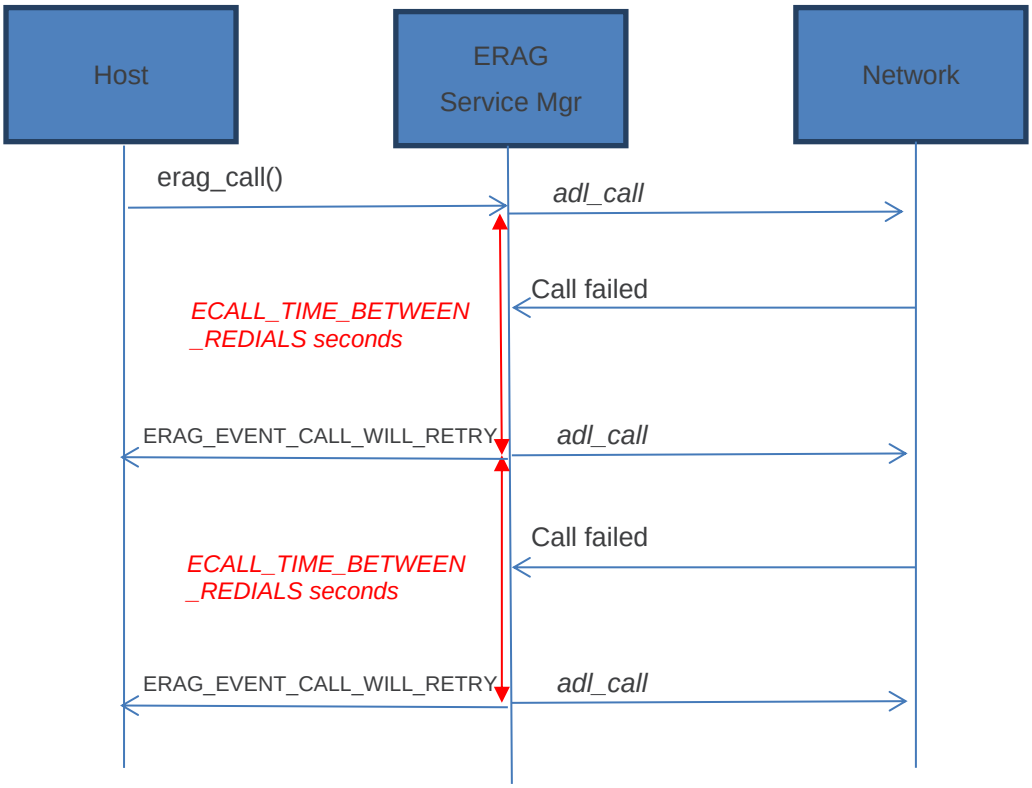
The minimum time between redials is controlled by ECALL_TIME_BETWEEN_REDIALS. This is from the moment the call is placed, until the new call is placed, under the assumption that the call not connected and has ended when the time is up. See chapter 5.3.12.

In the case where the time to do a redial is zero or negative, a one second timer will be used, to allow the stack to properly hang up, before redialing.

Below are examples showing how they interact.

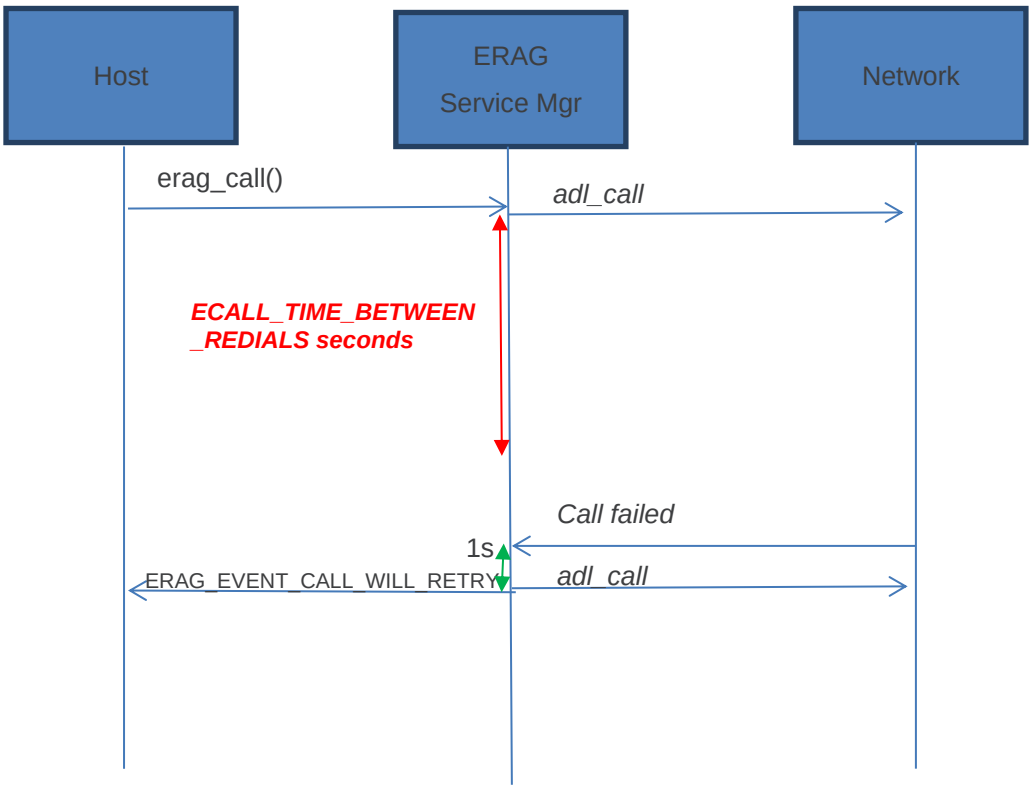
8.2. Example: No Answer and Call Cut

The following figure shows the case when there is no answer from the PSAP and the call fails before ECALL_TIME_BETWEEN_REDIALS seconds has passed.



8.3. Example: Call Still Dialing when Period Passed

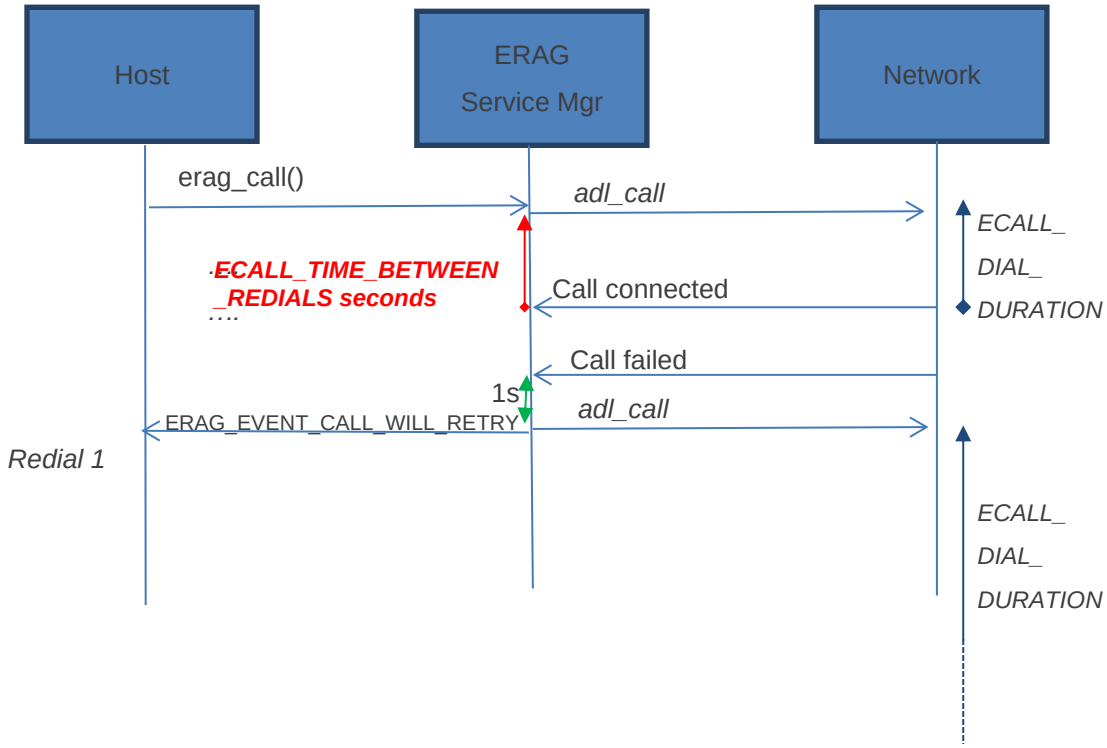
If the call is still dialing when the time ECALL_TIME_BETWEEN_REDIALS seconds has passed, it will not cut the call. Instead it will redial directly if the dial attempt is cut or not answered as shown in the following figure. Please note the 1 second timer to allow hang up and then redial.



8.4. Example: Call Connected then Cut

Note that once the call is connected the timers will stop. If the call is then cut, the redial starts immediately, after a 1 second before redialing.

Note that all both counter for number of redials and ECALL_DIAL_DURATION restart directly

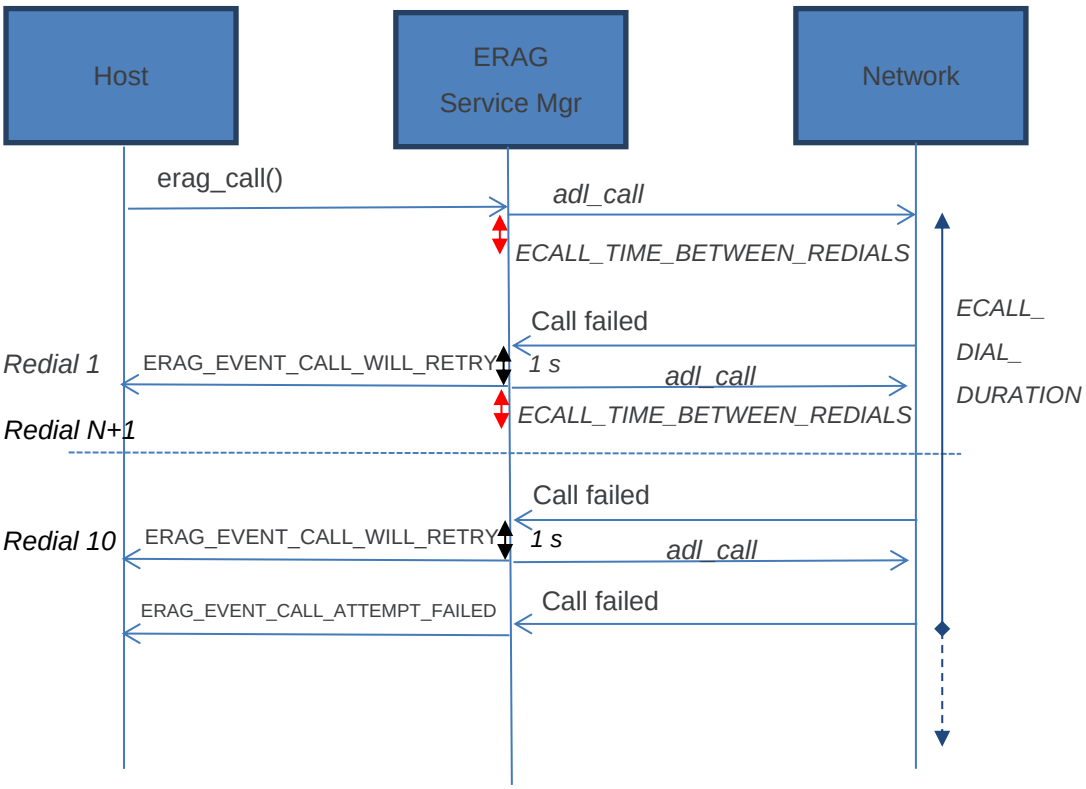


8.5. Example: Running Out of Retries

The following example shows the behavior with default values, 10 retries are done in a shorter period than ECALL_DIAL_DURATION.

ECALL_TIME_BETWEEN_REDIALS is by default 1 second.

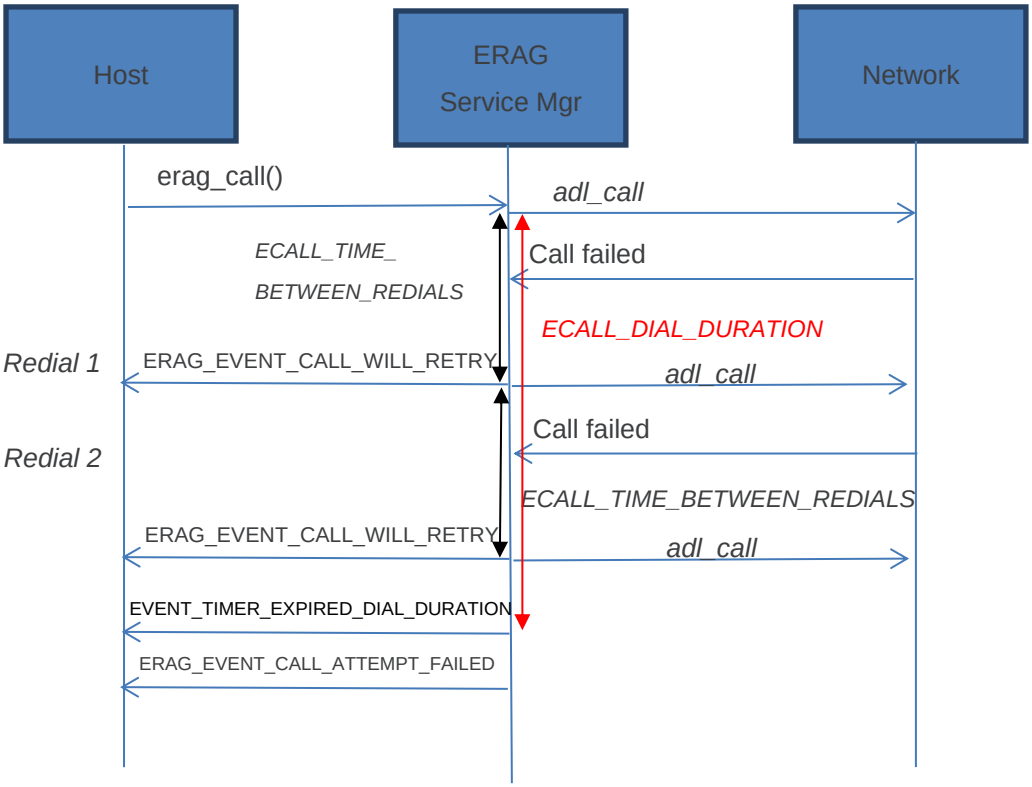
Note that if the ECALL_TIME_BETWEEN_REDIALS has passed, there will be an extra 1 second delay between call failure and call retry to ensure proper hang up.



8.6. Example: Running Out of Time

The following example shows when the ECALL_DIAL_DURATION expires before the number of redial attempts.

For this example imagine ECALL_TIME_BETWEEN_REDIALS is set about half of ECALL_DIAL_DURATION seconds.



Index

SIERRA
WIRELESS®