

Feature Specification- Software

Document information

Project	Beatus
Epic	Text message and alarm key
Status	Draft
Document revision	0016
Last updated Date	20 April 2018

This document analyzes and describes the implementation of Text message and alarm key especially the FP communication to the Message Server/Client. The design presented herein is at the time of writing a prototype that is not governed by controlling specifications or product requirements. It is anticipated that the design presented here will be a base for further specification and standardization work. Nevertheless it is anticipated that the design will have to be refactored and updated during the life cycle of the project.

Contents

DOCUMENT INFORMATION	1
CONTENTS	1
HISTORY	3
REFERENCES	3
TERMS & ABBREVIATIONS	3
FEATURE HIGHLIGHTS AND PROBLEM STATEMENTS	4
INTRODUCTION	4
MESSAGE FORMAT	5
<i>XML interface</i>	5
XML Schema Description	6
Schema Description Explication	8
SETUP AND SYNCHRONIZATION BETWEEN FP AND MS	12
USING THE ALARM SYSTEM IN A MULTI CELL SYSTEM	17
ALARMS SENT TO THE FP/HANDSET FROM MS	19
Priority	19
Confirmation type	20
Reference Number – substituting an alarm	20
Reference Number – deleting an alarm	20
Response handling	20
EXAMPLES	21
Alarm with user confirmation	21
Delete an Alarm	23
SMS SENT TO THE FP/HANDSET FROM MS	25
Priority	25
Confirmation types	25
SMS SEND FROM FP/HANDSET TO ANOTHER FP/HANDSET VIA A MESSAGES SERVER	26
Priority	26
ALARM SEND FROM FP/HANDSET TO A MS	32



BEACON MESSAGE SEND FROM FP/HANDSET TO A MS	34
HANDSET USER INTERFACE FOR ALARM RECEPTION	37
Receive first alarm	37
Receive More than One Alarm	37
Handling the size of the Alarm List	38
Enter the Alarm List	39
Receiving an Incoming	39
PROVISIONING OF BEACON SETTINGS	39
HANDSET SETUP FOR BEACON	39
RTX8200 SETUP	42
BEACON TEST INTERFACE	43
INTERFACE SPECIFICATION FOR TRANSFER DATA FROM HANDSET TO BASE VIA CISS.....	45
OPEN ISSUES TO DISCUSS	48
AUTO TEST	49
ALARM TEST 1 – CONFIRMATIONTYPE 0.....	49
ALARM TEST 2 – CONFIRMATIONTYPE 1.....	49
ALARM TEST 3 – CONFIRMATIONTYPE 2 - OK	49
ALARM TEST 4 – CONFIRMATIONTYPE 2 - REJECT	50
ALARM TEST 5 – CONFIRMATIONTYPE 2 – SUBSTITUTE ALARM	50
ALARM TEST 6 – CONFIRMATION TYPE 1 -DELETE	50
ALARM TEST 7 – CONFIRMATION TYPE 2 -DELETE	51
ALARM TEST 8 – 50 ALARMS CONFIRMATION TYPE 2	51
SMS TEST 1 – CONFIRMATION TYPE 0	52
SMS TEST 2 – CONFIRMATION TYPE 1 - OK	52
SMS TEST 3 – CONFIRMATION TYPE 1 - REJECT	52
SMS TEST 4 – 50 SMS CONFIRMATION TYPE 0	53

History

Revision	Author	Date	Comment
0001	LH	2014-02-27	Initial Revision
0002	LH	2014-03-18	Updated according to final specification from MobiCall
0003	LH	2014-05-05	Updated according to new specification from MobiCall
0004	LH	2016-05-24	Clarify the XML header needed in the XML signaling and persondata and senderdata use corrected
0005	LH	2016-09-29	AlarmDataDef corrected
0006	BFE	2017-06-20	Document updated with latest information
0007	BFE	2017-07-04	Number of rings has been updated
0008	BFE	2017-07-09	Added auto test cases, updated chapter "Handling the size of the Alarm List"
0009	BFE		Added beacon messages
0010	BFE	2018-01-26	Added documentation for CISS interface between handset and base Updated "send alarm from Handset" with beacon info.
0011	BFE	2018-02-6	Updated Beacon and general information
0012			Beacon field has been updated
0013	BFE	2018-02-19	Updated interface for beacon
0014	BFE	2018-02-20	Added information about multi cell handling
0015	BFE	2018-02-21	Added extension info for all handsets to the system info packet Working from release 410 and forward
0016	BFE	2018-02-27	Changes interface for beacon

References

There are no sources in the current document.

NVINTF3036_XML_Out_EN_V0503214.pdf

Ref	Description
[1]	NVINTF3036_XML_Out_EN_V15042014.pdf

Terms & Abbreviations

Term	Description
DECT	Digital Enhanced Cordless Telecommunications
PP	Portable Part (Handset)
FP	Fixed Part (Base station)
PI	Protocol Interface
MM	Mobility Management
CC	Call Control
DLC	Data Link Control
LCE	Link Control Entity
HLD	High Level Design
Alarm	Alarm messages that will be placed under the exclamation mark icon in the handset.
SMS	Simple message that will be placed under the envelope icon in the handset.
MS	Message server. Is a server that is connected to a basestation that can sent and received messages from a basestation.
Message	Information in XML format send between the FP and MS

Feature highlights and Problem Statements

- Send and receive SMS and alarms with different priority.
- Support up to 50 alarms that can wait in the handset for user confirmation/rejection
- Delete alarms and SMS in the handset
- Delete and substitute unacknowledged Alarm from a message server (MS)
- Receive and see more than one alarm at the same time in a list mode

Introduction

This document describes the functionality of an Alarm/SMS/Beacon system between a RTX basestation (FP) and a message server (MS).

The system is used to the following:

- Send/receive SMS on the handset
- Send alarms from the handset to a MS
- Receive alarms on the handset from a MS
- Send Beacon information to a MS

The system is based on sending UDP packets and send an acknowledge that the UDP packet has been received. In Figure 1 is shown an example of sending beacon information from a beacon to a MS through the handset and basestation. In Figure 2 is shown an example of sending an alarm from a messages server to a handset.

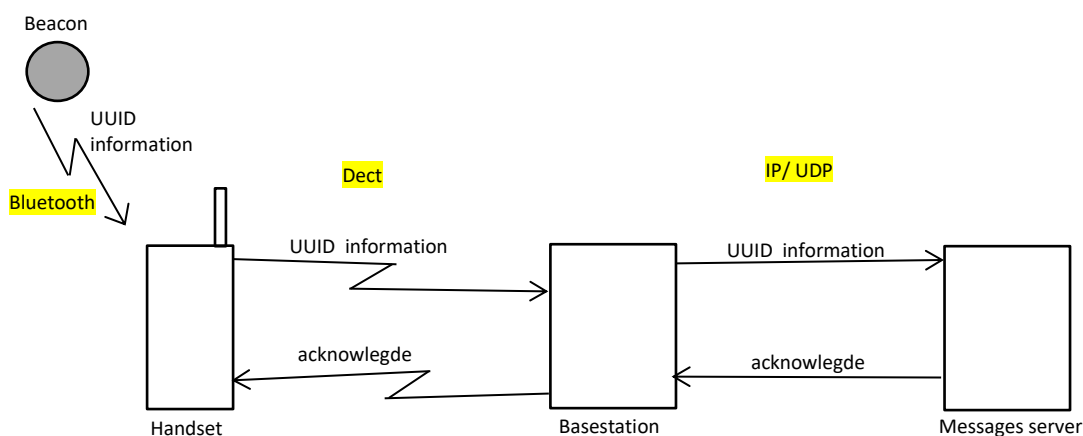


Figure 1: Example of sending Beacon information from a beacon via handset and basestation to a messages server.

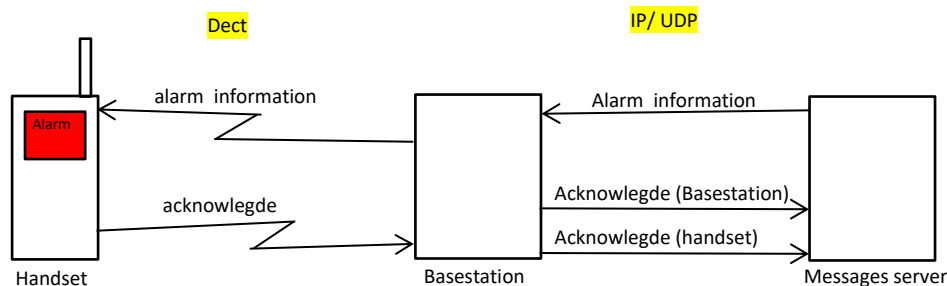


Figure 2: Example of sending an alarm from a message server to a handset.

There are 5 different message types that can be used between a MS and a FP.

XML tag	Description	Direction
Systeminfo	Messages that handles synchronization between FP and MS	MS -> FP FP -> MS
Login	Messages that informs the MS about which handset that is located at the FP.	FP->MS
Job	Alarms send from the MS to the FP and will pop up at the handset and be placed under the alarm icon.	MS->FP
	SMS send from the MS to the FP and will pop up at the handset and be placed under the SMS icon.	MS->FP
	SMS send from FP/handset to another FP/handset via a messages server	FP->MS
alarm	Alarm send from FP/handset to a messages server	FP->MS
beacon	Beacon information from FP/handset to a messages server	FP->MS

In the following the messages format between the MS and FP is describes followed by the different messages types. Finally, the handset UI is briefly described for alarms.

System info and Login are only used when the messages server need to send alarms/SMS to the basestation/handset.

For messages servers where only receiving beacons or alarms from the basestation/handset the Systeminfo and Login procedures can be skipped in the messages server.

Message format

Below the XML format is described and a brief description of the different parameters are described.

XML interface.

Total length of XML packet is 4096 bytes.

The interface follows the XML standards. Here are some basic rules:

- All Xml packets need to start with this XML header:
`<?xml version="1.0" encoding="UTF-8"?>`
- No spaces in the <> elements except for separating attributes. Examples:
`<element>` is valid
`< element >` is not valid (spaces before and after "element")
`<element >` is not valid (space after "element")

<element attribute="attributel"> is valid
 <element attribute = "attributel" > is not valid
 (spaces before and after = and after the second "
 <element /> is not valid (space before /)

- All elements have to be properly closed.

At the moment UTF-8 is not supported. Only the below extended ASCII table is supported:

RTX Character editor [T:\bfe\10_messages_Alarm\Charedit\PixelHeight10_reg.c]

File Edit Font Help

Currently displaying ascii char values: 0x0000 - 0x00FF

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	`	p	€	í		°	À	Ð	à	ð
1			!	1	A	Q	a	q	ı	'	ı	±	Á	Ñ	á	ñ
2			"	2	B	R	b	r	,	'	ø	Č	Â	Ò	â	ò
3			#	3	C	S	c	s	f	"	£	č	Ã	Ó	ã	ó
4			\$	4	D	T	d	t	„	"	¥	'	Ä	Ô	ä	ô
5			%	5	E	U	e	u	...	•	¥	µ	Å	Õ	å	õ
6			&	6	F	V	f	v	†	—	ı	¶	Æ	Ö	æ	ö
7			'	7	G	W	g	w	‡	—	§	·	Ç	×	ç	÷
8			(8	H	X	h	x	^	~	ˆ	ˆ	È	Ø	è	ø
9)	9	I	Y	i	y	Ř	ř	Û	Ď	É	Ù	é	ù
A			*	:	J	Z	j	z	Š	š	û	d'	Ê	Ú	ê	ú
B			+	;	K	[k	{	<	>	«	»	Ë	Û	ë	ü
C			,	<	L	\	l		Œ	œ	Ë	ť	Ì	Ü	ì	ü
D			-	=	M]	m	}	Š	š	ě	t'	Í	Ý	í	ý
E			.	>	N	^	n	~	Ž	ž	Ň	ň	Î	Þ	î	þ
F			/	?	O	_	o	Ğ	ğ	Ÿ	—	ı	İ	ß	ï	ÿ

XML Schema Description

This is the XML Schema Description for the generic outgoing XML-based interface from the messages server.

The FP shall ignore unknown fields and packages received by the MS, to make sure that upgrading the MS to a newer version with new features does not cause the FP to crash.

```
<xs:element name="request" type="FrameDef" />
```

```
<xs:element name="response" type="FrameDef" />
```

```
<xs:complexType name="FrameDef">
```

```
<xs:attribute name="version" type="xs:string" />
```

```
<xs:attribute name="type" type="xs:string" />
```

```
<xs:sequence>
```

```
<xs:element name="externalid" type="xs:string" minOccurs="1"/>
```

```
<xs:element name="status" type="xs:decimal" minOccurs="0"/>
```

```
<xs:element name="statusinfo" type="xs:string" minOccurs="0"/>
```

```
<xs:element name="systemdata" type="SystemDataDef" minOccurs="1"/>
```

```
<xs:element name="jobdata" type="JobDataDef" minOccurs="0"/>
```

```
<xs:element name="alarmdata" type="AlarmDataDef" minOccurs="0"/>
```

```
<xs:element name="beacondata" type="BeaconDataDef" minOccurs="0"/>
```

```
<xs:element name="logindata" type="LoginDataDef" minOccurs="0"/>
```

```
<xs:element name="persondata" type="PersonDataDef" minOccurs="0"/>
```

```

    <xs:element name="senderdata" type="PersonDataDef" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="SystemDataDef">
  <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="1"/>
    <xs:element name="datetime" type="xs:dateTime" minOccurs="1"/>
    <xs:element name="timestamp" type="xs:string" minOccurs="1"/>
    <xs:element name="status" type="xs:decimal" minOccurs="0"/>
    <xs:element name="statusinfo" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="JobDataDef">
  <xs:sequence>
    <xs:element name="alarmnumber" type="xs:decimal" minOccurs="0"/>
    <xs:element name="referencenumber" type="xs:string" minOccurs="0"/>
    <xs:element name="priority" type="xs:decimal" minOccurs="0"/>
    <xs:element name="flash" type="xs:decimal" minOccurs="0"/>
    <xs:element name="rings" type="xs:decimal" minOccurs="0"/>
    <xs:element name="confirmationtype" type="xs:decimal" minOccurs="0"/>
    <xs:element name="messages" type="MessagesDef" minOccurs="0"/>
    <xs:element name="status" type="xs:decimal" minOccurs="0"/>
    <xs:element name="statusinfo" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="MessagesDef">
  <xs:sequence>
    <xs:element name="message1" type="xs:string" minOccurs="0"/>
    <xs:element name="message2" type="xs:string" minOccurs="0"/>
    <xs:element name="messageui" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="AlarmDataDef">
  <xs:sequence>
    <xs:element name="type" type="xs:decimal" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="BeaconDataDef">
  <xs:sequence>
    <xs:element name="type" type="xs:decimal" minOccurs="0"/>
    <xs:element name="uuid" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="LoginDataDef">
  <xs:sequence>
    <xs:element name="status" type="xs:decimal" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="PersonDataDef">
  <xs:sequence>
    <xs:element name="address" type="xs:string" minOccurs="0"/>
    <xs:element name="name" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

```



```
<xs:element name="location" type="xs:string" minOccurs="0"/>
<xs:element name="status" type="xs:decimal" minOccurs="0"/>
<xs:element name="statusinfo" type="xs:string" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
```

Schema Description Explication

Child description for (request / response)

- XML child "version" (request / response)
 - only "1" supported
- XML child "type" (request / response)
 - "systeminfo": Watchdog frame / system error information (MUST be contained in all frames)
 - "job": alarm job sent from messages server.
 - "alarm": 3rd party alarms to the messages server (e.g. redbutton)
 - "beacon": beacon message with uuid from a beacon
 - "login": Login / Logout of persons
- XML child "externalid" (request / response)
 - unique notification external ID
- XML child "status", "statusinfo" (request / response)

Note! Not used in request.

 - "status" is the decimal value the following values.
 - "statusinfo" is the string values the following values:
 - 0: Not accepted by external system
 - 1: Accepted by external system
- XML child "systemdata" (request / response)

Must be contained in all frames.

 - See **"Error! Reference source not found."**
- XML child "jobdata" (request / response)
- Only used when frame type is "job", then this sequence must be contained.
 - See **"Error! Reference source not found."**
- XML child "alarmdata" (request / response)

Only used when frame type is "alarm", then this sequence must be contained.

This sequence will only be relevant in direction to MS system.

 - See **"Error! Reference source not found."**
- XML child "beacondata" (request / response)

Only used when frame type is "beacon", then this sequence must be contained.

This sequence will only be relevant in direction to MS system.

 - See **"Error! Reference source not found."**
- XML child "logindata" (request / response)

Only used when frame type is "alarm", then this sequence must be contained.

This sequence will only be relevant in direction to MS system.

 - See **"Error! Reference source not found."**
- XML child "persondata", "senderdata" (request / response)



“persondata” is used with frame types: “job”, “alarm”, “login”.

If “senderdata” is not present in frame, the “persondata” is related to the handset which has logged in to MS.

If “senderdata” is present, then “senderdata” will contain calling party information and “persondata” will contain called party information.

- See **“Error! Reference source not found.”**



Child description for (SystemDataDef)

This sequence MUST be present in all frames.

- XML child "name" (SystemDataDef)
 - System name eg. base name or messages server system name.
- XML child "datetime" (SystemDataDef)
 - "2014-03-04T10:16:20" is local date and time for log-reason and can be individual formatted between the systems. Notice: this timestamp is not used when presenting the alarm in the handset.
- XML child "timestamp" (SystemDataDef)
 - UTC time in HEX format.
- XML child "status", "statusinfo" (SystemDataDef)
 - "status" is the decimal value the following values.
 - "statusinfo" is the string values the following values:
 - 0: System not running
 - 1: System running

Child description for (JobDataDef)

- ~~XML child "alarmnumber" (JobDataDef)~~
~~decimal value from 10 to 9999~~
 Not used at the moment.
- XML child "referencenumber" (JobDataDef)

When the string referencenumber is present in the message from a message server the message is interpreted as an alarm in the handset.

If the referencenumber string is not present, the message is interpreted as a SMS.

Please read further description of the referencenumber in the chapters below.
- XML child "priority" (JobDataDef)

It is possible to set an alarm with different priority. Further description is placed in the chapters below.
- XML child "flash" (JobDataDef)
 - values 0 or 1, popup (like flash-SMS).

This is only supported for SMS not Alarms.
- XML child "rings" (JobDataDef)
 - values 0..9, number of ring tone alerts, where 0 means endless ringing.

Only valid for Alarms
- ~~XML child "beepcode" (JobDataDef)~~
~~values 0..19, type of alert tone. Decimal values are individual depending on external system.~~
 Not used at the moment. If used, then the string is ignored.
- XML child "confirmationtype" (JobDataDef)

Please see the sections "Alarms sent to the FP/handset from MS" and "SMS sent to the FP/handset from MS" for further description.
- XML child "messages" (JobDataDef)

Only used in request and MUST always be present in frame type "job".

 - See **"Error! Reference source not found."**

- XML child "status", "statusinfo" (JobDataDef)
 - "status" is the decimal value the following values.
 - "statusinfo" is the string values the following values:

0: No answer	Comment: The end user has not answered
1: Answer	Comment: The end user has answered -> Delivered to handset
2: Busy	Comment: The end user was busy
3: Error	Comment: The end user an error
4: Confirmed	Comment: The end user has confirmed the alarm positive
5: Rejected	Comment: The end user has rejected the alarm
6: Dialing	Comment: The remote system is dialing the number -> Base is paging handset
10: Canceled	Comment: The remote system cancelled the job
11: Not reached	Comment: The end user was not reachable
21: Read	Comment: The end user read the job

Child description for (MessageDef)

- ~~XML child "message1" (MessageDef)~~
 - ~~"message1" can be shown as pre message when device is receiving a message.~~
 - Not used at the moment. If used, then the string is ignored.
- ~~XML child "message2" (MessageDef)~~
 - ~~"message2" can be used as used as message title when user opens a message.~~
 - Not used at the moment. If used, then the string is ignored.
- XML child "messageui" (MessageDef)
 - "messageui" is allways the bodytext of the message.

Child description for (AlarmDataDef)

- XML child "type" (AlarmDataDef)
 - values 0..9
 - 0: Man Down
 - 1: No Movement
 - 2: Running
 - 3: Pull Cord
 - 4: Red Key
 - 5-9 Reserved

Child description for (BeaconDataDef)

- XML child "type" (BeaconDataDef)
 - values 0..1
 - 0: entering proximity of the beacon
 - 1: leaving proximity of the beacon
- XML child "uuid" (BeaconDataDef)
 - string holding the uuid of the beacon

Child description for (LoginDataDef)

- XML child "status" (LoginDataDef)
 - "status" is the decimal value the following values.
 - 0: Logout
 - 1: Login

Child description for (PersonDataDef)

- XML child “address” (PersonDataDef)
- Called party number (handset SIP account).
- XML child “name” (PersonDataDef)
- Called party display name (handset SIP account display name).
- XML child “location” (PersonDataDef)
- Can be handset current location if avail (Base name).
This is the name of the base station where the handset first time was registered after startup (SIP registration), so the location cannot be used located where the handset actually has the DECT connection, when walking around between different basestations.

Setup and Synchronization between FP and MS

On the FP webpage, the text messaging is configured. In Figure 3 the webpage setup is shown.

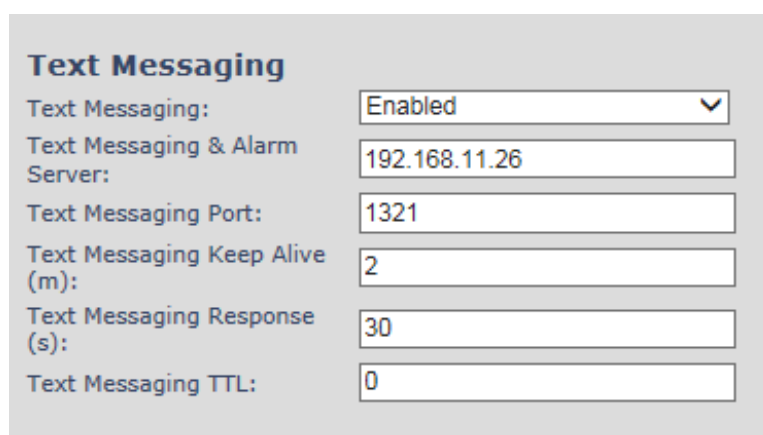


Figure 3: Text messaging setup.

Text Messaging:

Disabled: SMS and alarm cannot be used

Enabled without server: Used to send SMS between handsets that is located on the same FP or multicell system, but without using a MS. This feature is not described in this document.

Enabled: The FP is connected to a MS and the MS is handling all SMS/Alarms.

Text messaging & Alarm server: This is the MS the FP is connected to.

Text messaging port: This is the IP port that is used by the MS and FP.

Text Messaging Keep Alive: Minutes between the FP will send a keep alive signal to the MS

Text messaging response: If the FP does not receive a response from the MS within 30 seconds, when the FP/handset has sent a SMS to the MS, then the FP will inform the handset that the messages was not delivered.

When the FP is started with the configuration set in Figure 3 there is sent a sync messages followed by a sync message for each phone that is locating to the FP.

FP to MS sync:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="17.6.15.1526" type="systeminfo">
  <externalid>0352675351</externalid>
  <systemdata>
    <name>SME VoIP</name>
    <datetime>1970-01-01 00:00:09</datetime>
    <timestamp>00000009</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
</request>
```

MS confirm response to FP:

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="1.0" type="systeminfo">
  <systemdata>
    <name>Micromedia-Alert</name>
    <datetime>2017-06-15T16:07:10</datetime>
    <timestamp>5942948e</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <externalid>0352675351</externalid>
  <status>1</status>
  <statusinfo>Accepted by external system</statusinfo>
</response>
```

FP information to MS about registered handset (login) – notice the extension number:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="17.6.15.1526" type="login">
  <externalid>3294079664</externalid>
  <systemdata>
    <name>SME VoIP</name>
    <datetime>2017-06-15 09:07:51</datetime>
    <timestamp>594294b7</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <logindata>
    <status>1</status>
  </logindata>
  <senderdata>
    <address>991</address>
    <name>991</name>
    <location>SME VoIP</location>
  </senderdata>
</request>
```

MS confirm response to FP:

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="1.0" type="login">
  <systemdata>
    <name>Micromedia-Alert</name>
    <datetime>2017-06-15T16:07:54</datetime>
    <timestamp>594294ba</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <externalid>3294079664</externalid>
  <status>1</status>
  <statusinfo>Accepted by external system</statusinfo>
</response>
```

The FP and MS has now informed each other that they exist.

Now both the MS and FP can send a “keep alive” signal to each other.

In the “keep Alive” messages from the FP there is also added all the handset located to the FP. So the MS can always request this info in case it is lost for example if the MS has been restarted

For the FP it can be set at the homepage how often there is send a keep alive signal to the MS. In Figure 3 it is set to 2 minutes.

Below is shown the “keep alive” signals. Notice that the response signal always use the same external Id as the request.

FP send “keep alive” to MS: Notice that the name field can be empty.

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="18.2.21.1359" type="systeminfo">
  <externalid>1528744172</externalid>
  <systemdata>
    <name>SME VoIP</name>
    <datetime>2018-02-21 07:34:45</datetime>
    <timestamp>5a8d7575</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <senderdata>
    <address>400</address>
    <name></name>
    <address>401</address>
    <name>Bob Andersen</name>
    <address>402</address>
    <name>Hansi</name>
    <address>403</address>
    <name>George Lucas</name>
    <address>404</address>
    <name>Michael Jensen</name>
    <address>406</address>
    <name>406</name>
    <address>407</address>
    <name>Fenger</name>
    <address>408</address>
    <name>408</name>
    <address>410</address>
    <name>410</name>
  </senderdata>
</request>
```

MS confirm response to FP:

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="1.0" type="systeminfo">
  <systemdata>
    <name>Micromedia-Alert</name>
    <datetime>2017-06-15T16:13:10</datetime>
    <timestamp>594295f6</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <externalid>5a8d7575</externalid>
  <status>1</status>
  <statusinfo>Accepted by external system</statusinfo>
</response>
```

MS send "keep alive" to FP:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.0" type="systeminfo">
  <systemdata>
    <name>Micromedia-Alert</name>
    <datetime>2017-06-15T16:17:59</datetime>
    <timestamp>59429717</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <externalid>mmi59427f9a-0</externalid>
</request>
```

FP confirm response to MS:

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="18.2.21.1359" type="systeminfo">
  <externalid>mmi59427f9a-0</externalid>
  <systemdata>
    <name>SME VoIP</name>
    <datetime>2018-02-21 07:39:03</datetime>
    <timestamp>5a8d7677</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <senderdata>
    <address>400</address>
    <name>hej400</name>
    <address>401</address>
    <name>Bob Andersen</name>
    <address>402</address>
    <name>Hansi</name>
    <address>403</address>
    <name>George Lucas</name>
    <address>404</address>
    <name>Michael Jensen</name>
    <address>406</address>
    <name>406</name>
    <address>407</address>
    <name>Fenger</name>
    <address>408</address>
    <name>408</name>
    <address>410</address>
    <name>410</name>
  </senderdata>
</response>
```

If a handset is turned off there is sent a logout signal to the MS

FP information to MS about deregistered handset (logout)


```
<?xml version="1.0" encoding="UTF-8"?>
<request version="17.6.15.1526" type="login">
  <externalid>0587227135</externalid>
  <systemdata>
    <name>SME VoIP</name>
    <datetime>2017-06-16 02:07:02</datetime>
    <timestamp>59438396</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <logindata>
    <status>0</status>
  </logindata>
  <senderdata>
    <address>991</address>
    <name>991</name>
    <location>SME VoIP</location>
  </senderdata>
</request>
```

MS confirm response to FP:

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="1.0" type="login">
  <systemdata>
    <name>Micromedia-Alert</name>
    <datetime>2017-06-16T09:07:05</datetime>
    <timestamp>59438399</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <externalid>0587227135</externalid>
  <status>1</status>
  <statusinfo>Accepted by external system</statusinfo>
</response>
```

Using the alarm system in a multi Cell system

In Figure 4 there is shown a multicell system and how the login information is send to the messages server. After the login information has been received the MS knows which BS to send an alarm to if for example the MS need to alert extension 202.

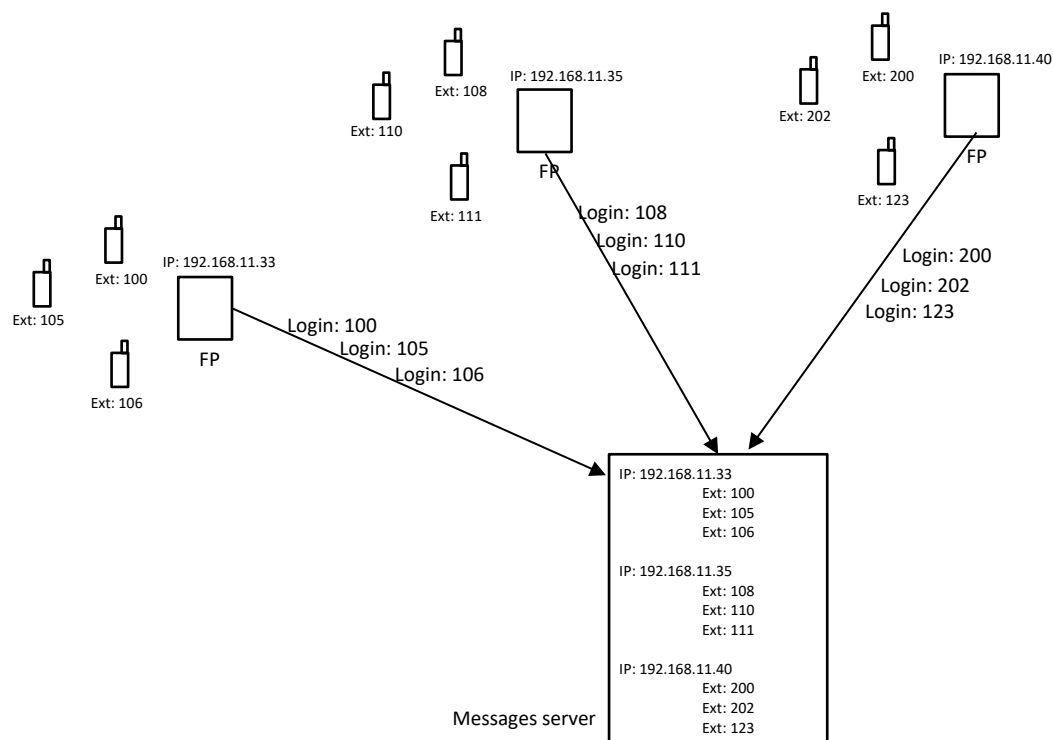


Figure 4: Login information send to MS when handsets are located to a BS.

If a handset is turned off there is send a logout messages to the MS. This is shown in Figure 5.

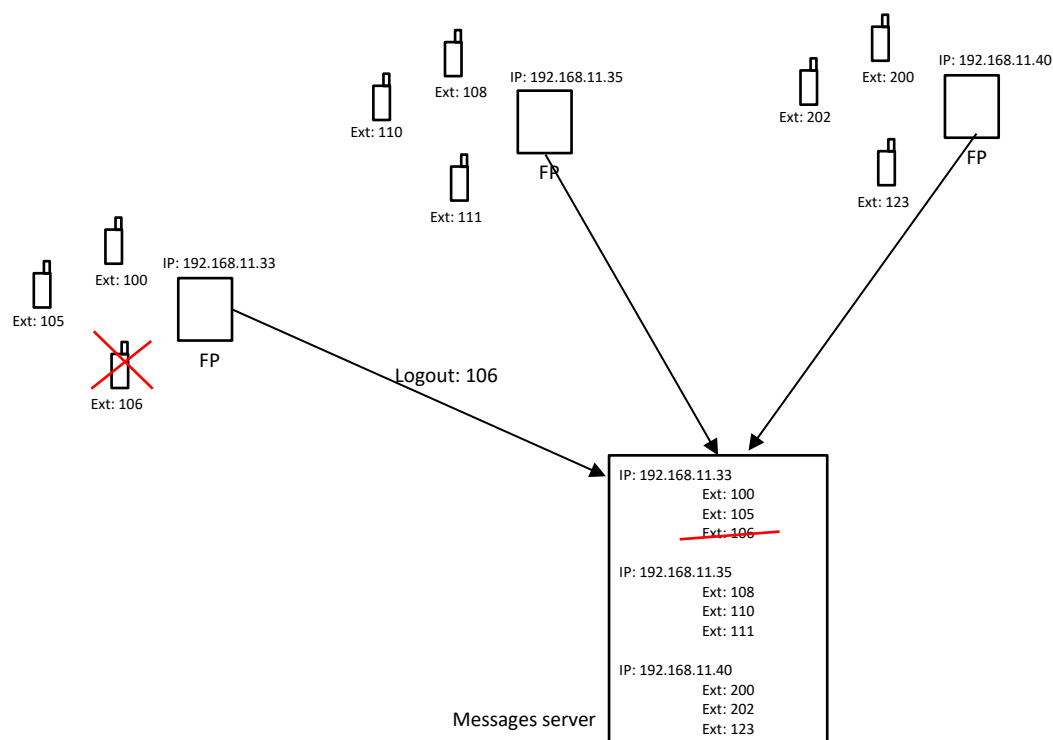


Figure 5: Logout is send to the MS when a handset is turned off.

Attention:

- 1) If the battery is removed from the handset there is not send a logout to the MS. So, if the handset is turned ON and locates on a new BS there is send a new login to the MS. The MS should then remove the extension from the old IP in the list and place the handset extension with the new IP address.
- 2) If the battery is removed from the handset there is not send a logout to the MS. When the MS is sending an alarm to the handset there will only be send back one response. There will not be send a response that the handset has received the alarm. The MS now knows that there is something wrong with the handset.
- 3) If the MS is in doubt where the handset extension is located it is also possible just to send the alarm to all FP. All FP will respond with status "11: Not reached" except the FP where the handset is located. This will of cause create some traffic on the network and load the FP with unnecessary traffic – so it is not recommended to use this method.
- 4) If the UDP packet with handset login information is lost in the network the MS will not know that the handset has been located to a FP, but the MS will also get the information about registered handsets in the "keep alive" message. The MS can also send "keep alive" to the FP and the FP will then respond with information about registered handsets.
- 5) If the MS is rebooted and the IP/extension list is lost the MS does not know which extension belongs to which FP IP address. In this case the MS can just request the information by sending the "keep alive" request.

Alarms sent to the FP/handset from MS

A messages server (MS) can send a UDP packet to a basestation (FP) with XML information about the alarm. The FP will send the alarm to the handset and a response is sent back to the MS.

In Figure 6 is shown how an alarm is shown in the handset.



Figure 6: Alarm displayed in the handset.

Priority

It is possible to set an alarm with different priority, below is shown the different colors that is used in the handset depending on the priority.

- 1: red
- 2: red
- 3: red
- 4: Yellow
- 5: Yellow
- 6: Yellow
- 7: Green
- 8: Green
- 9: Green
- All other values: Green

Confirmation type

XML child "confirmationtype" (JobDataDef)

- 0: No User confirmation needed.
The FP will send back a confirmation to MS that the message has been received in the FP.
- 1: User Received Job confirmation needed.
The FP will send back a confirmation to MS that the message has been received in the FP.
The FP will send back a confirmation to MS that the message has been received in the handset.
- 2: User Confirm/Reject Job confirmation needed.
The FP will send back a confirmation to MS that the message has been received in the FP.
The FP will send back a confirmation to MS that the message has been received in the handset.
The FP will send back a confirmation to MS if the user confirm/reject the message

Reference Number – substituting an alarm

The reference number is unique number per alarm.

When sending an alarm, the reference number always need to be present otherwise the message is interpreted as an SMS.

If there is sent an alarm with the same referencenumber as a previous alarm that has NOT YET been confirmed/rejected (confirmation type 2) the first alarm will be deleted and only the new alarm will be saved and displayed in the handset.

If there is sent an alarm with the same referencenumber as a previous alarm that has been confirmed/rejected then both alarms is shown.

If there is sent an alarm with the same referencenumber as a previous alarm that has confirmation type 0 or 1 then both alarms are shown in the handset.

Reference Number – deleting an alarm

It is possible to delete an Alarm that has NOT YET been confirmed/rejected in the handset. This means that it is only possible to delete an alarm with confirmation type 2.

For deleting an existing alarm the same reference number should be used and the status should be set to 10.

Response handling

For confirmation type 1 and 2 there is always send a response to the MS that the FP has received the messages and a response that the handset has received the message.

The MS need both responses to be sure that the messages has been received in the handset.

The MS should not send a new message to the same handset before both responses has been received.

In case the MS does not receive the response from the handset, the MS should wait a minimum of 10 sec. before trying to send a message again to the same handset. This is needed so the FP has enough time to timeout the first messages.

When using confirmation type 0 the MS will not know if the SMS/alarm has been lost from the FP to the handset. Here the MS should always wait 10 sec. between each message to the same handset.

Examples

Alarm with user confirmation

Below example shown an alarm send from the MS with confirmation type 2

MS send alarm to FP:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.0" type="job">
  <systemdata>
    <name>Micromedia-Alert</name>
    <datetime>2017-06-20T09:25:24</datetime>
    <timestamp>5948cde4</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <jobdata>
    <alarmnumber>5</alarmnumber>
    <referencenumber>5</referencenumber>
    <priority>1</priority>
    <flash>0</flash>
    <rings>0</rings>
    <confirmationtype>2</confirmationtype>
  </jobdata>
  <messages>
    <message1></message1>
    <message2></message2>
    <messageuui>Alarm1 text</messageuui>
  </messages>
  <status>0</status>
  <statusinfo/>
  </jobdata>
  <persondata>
    <address>991</address>
    <status>0</status>
    <statusinfo/>
  </persondata>
  <externalid>mmi5948c09b-5</externalid>
</request>
```

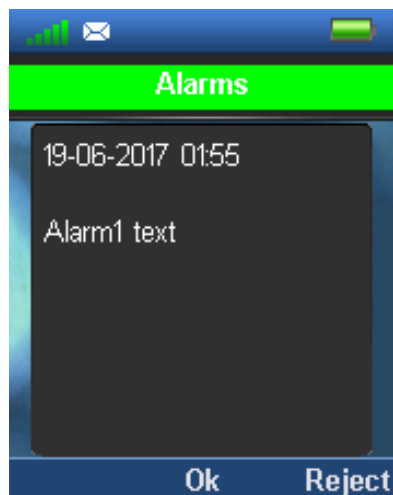
FP response to MS, that FP has received the alarm:

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="17.6.15.1526" type="job">
<externalid>mmi5948c09b-5</externalid>
<status>1</status>
<systemdata>
<name>SME VoIP</name>
<datetime>2017-06-20 02:25:22</datetime>
<timestamp>5948cde2</timestamp>
<status>1</status>
<statusinfo>System running</statusinfo>
</systemdata>
<senderdata>
<address>991</address>
<name>991</name>
<location>SME VoIP</location>
</senderdata>
</response>
```

FP response to MS, that handset has received the alarm:

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="17.6.15.1526" type="job">
<externalid>mmi5948c09b-5</externalid>
<systemdata>
<name>SME VoIP</name>
<datetime>2017-06-20 02:25:23</datetime>
<timestamp>5948cde3</timestamp>
<status>1</status>
<statusinfo>System running</statusinfo>
</systemdata>
<jobdata>
<priority>1</priority>
<messages>
<message1></message1>
<message2></message2>
<messageuui></messageuui>
</messages>
<status>1</status>
<statusinfo></statusinfo>
</jobdata>
<senderdata>
<address>991</address>
<name>991</name>
<location>SME VoIP</location>
</senderdata>
</response>
```

The handset MMI will now look like below:



The user can now press OK or reject

User has pressed OK and FP send response to MS:

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="17.6.15.1526" type="job">
<externalid>mmi5948c09b-5</externalid>
<systemdata>
<name>SME VoIP</name>
<datetime>2017-06-20 02:25:38</datetime>
<timestamp>5948cdf2</timestamp>
<status>1</status>
<statusinfo>System running</statusinfo>
</systemdata>
<jobdata>
<priority>1</priority>
<messages>
<message1></message1>
<message2></message2>
<messageuui></messageuui>
</messages>
<status>4</status>
<statusinfo></statusinfo>
</jobdata>
<senderdata>
<address>991</address>
<name>991</name>
<location>SME VoIP</location>
</senderdata>
</response>
```

Delete an Alarm

MS send alarm to FP:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.0" type="job">
  <systemdata>
    <name>Micromedia-Alert</name>
    <datetime>2017-06-20T09:58:25</datetime>
    <timestamp>5948d5a1</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <jobdata>
    <alarmnumber>5</alarmnumber>
    <referencenumber>5</referencenumber>
    <priority>1</priority>
    <flash>0</flash>
    <rings>0</rings>
    <confirmationtype>2</confirmationtype>
    <messages>
      <message1></message1>
      <message2></message2>
      <messageuui>Alarm1 text</messageuui>
    </messages>
    <status>0</status>
    <statusinfo/>
  </jobdata>
  <persondata>
    <address>991</address>
    <status>0</status>
    <statusinfo/>
  </persondata>
  <externalid>mmi5948c0a6-5</externalid>
</request>
```

MS request the FP to delete the alarm – notice that the same reference number is used and status is 10


```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.0" type="job">
  <systemdata>
    <name>Micromedia-Alert</name>
    <datetime>2017-06-20T09:58:25</datetime>
    <timestamp>5948d5a1</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <jobdata>
    <alarmnumber>5</alarmnumber>
    <referencenumber>5</referencenumber>
    <priority>1</priority>
    <flash>0</flash>
    <rings>0</rings>
    <confirmationtype>2</confirmationtype>
    <messages>
      <message1></message1>
      <message2></message2>
      <messageuui>Alarm1 text</messageuui>
    </messages>
    <status>10</status>
    <statusinfo/>
  </jobdata>
  <persondata>
    <address>991</address>
    <status>0</status>
    <statusinfo/>
  </persondata>
  <externalid>mmi5948c0a6-5</externalid>
</request>
```

SMS sent to the FP/handset from MS

When sending a SMS the reference number string should be removed from the XML message. When this is done, the messages will be interpreted as an SMS instead of an alarm.

It is not possible to send a delete request from a MS to the FP/handset for a SMS.

It is not possible to substitute a SMS with a new SMS.

Priority

An SMS can be sent to a handset with 2 priorities:

0: Normal

1: Urgent

See Figure 11 for screen UI

Confirmation types

There are 2 confirmations types for SMS

- 0: No User confirmation needed.

The FP will send back a confirmation to MS that the message has been received in the FP.

The FP will send back a confirmation to MS that the message has been received in the handset.

- 1: User Confirm/Reject Job confirmation needed.

The FP will send back a confirmation to MS that the message has been received in the FP.

The FP will send back a confirmation to MS that the message has been received in the handset.

The FP will send back a confirmation to MS if the user confirm/reject the message

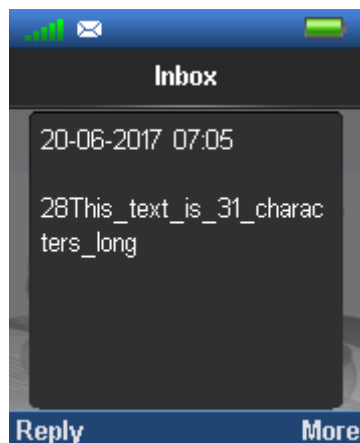


Figure 7: Confirmation type 0

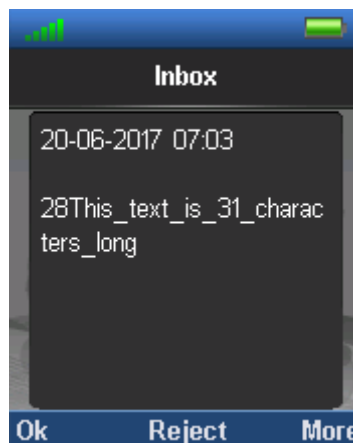


Figure 8: Confirmation type 1

If the pop up screen is left (using on hook), before pressing ok/reject it is no longer possible to the send an ok/reject back to the MS.

SMS send from FP/handset to another FP/handset via a messages server

It is possible to send a SMS from one handset to another handset via a MS. The MS need to support this.

In Figure 9 the flow for sending a SMS between 2 handsets is shown.

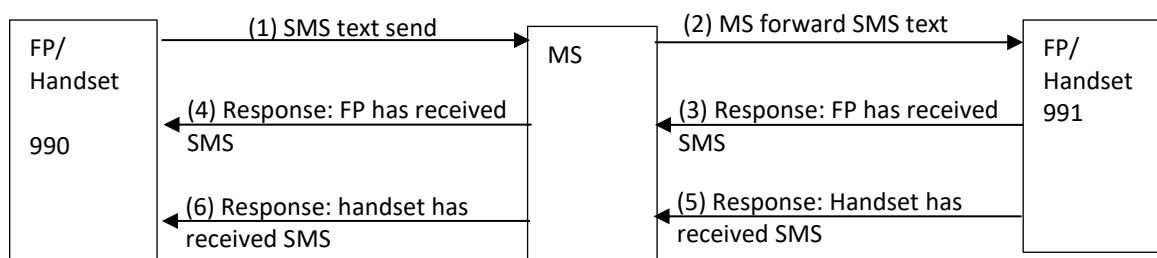


Figure 9: Flow for sending a SMS via a MS

After handset 990 has received the last response from handset 991 the text "Message sent" will be shown in handset 990. In case handset 991 does not respond, handset 990 will after 30 sec. show "message not set". See section "Setup and Synchronization between FP and MS" for further information about timeout.

Priority

An SMS can be sent from a handset with 2 priorities:

0: Normal

1: Urgent

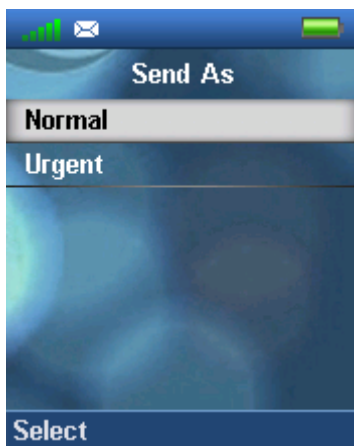


Figure 10: Send options for priority.

Urgent SMS is shown with an exclamation mark as shown below:

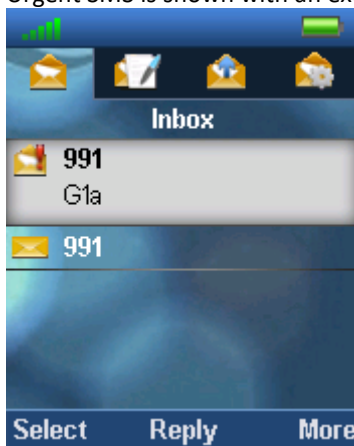


Figure 11: Sms with different priority.

Below is shown the 6 messages that is sent between the FP's and MS

(1) SMS text send

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="17.6.15.1526" type="job">
  <externalid>0649396368</externalid>
  <systemdata>
    <name>SME VoIP</name>
    <datetime>2017-06-20 04:12:16</datetime>
    <timestamp>5948e6f0</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <jobdata>
    <priority>1</priority>
    <messages>
      <message1></message1>
      <message2></message2>
      <messageuui>Tw</messageuui>
    </messages>
    <status>0</status>
    <statusinfo></statusinfo>
  </jobdata>
  <senderdata>
    <address>990</address>
    <name>990</name>
    <location>SME VoIP</location>
  </senderdata>
  <persondata>
    <address>991</address>
  </persondata>
</request>
```

(2) MS forward SMS text

```

<?xml version="1.0" encoding="UTF-8"?>
<request version="17.6.15.1526" type="job">
<externalid>0649396368</externalid>
<systemdata>
<name>SME VoIP</name>
<datetime>2017-06-20 04:12:16</datetime>
<timestamp>5948e6f0</timestamp>
<status>1</status>
<statusinfo>System running</statusinfo>
</systemdata>
<jobdata>
<priority>1</priority>
<messages>
<message1></message1>
<message2></message2>
<messageuui>Tw</messageuui>
</messages>
<status>0</status>
<statusinfo></statusinfo>
</jobdata>
<senderdata>
<address>990</address>
<name>990</name>
<location>SME VoIP</location>
</senderdata>
<persondata>
<address>991</address>
</persondata>
</request>

```

(3) Response: FP has received SMS

```

<?xml version="1.0" encoding="UTF-8"?>
<response version="17.6.15.1526" type="job">
<externalid>0649396368</externalid>
<status>1</status>
<systemdata>
<name>SME VoIP</name>
<datetime>2017-06-20 04:12:16</datetime>
<timestamp>5948e6f0</timestamp>
<status>1</status>
<statusinfo>System running</statusinfo>
</systemdata>
<senderdata>
<address>991</address>
<name>991</name>
<location>SME VoIP</location>
</senderdata>
<persondata>
<address>990</address>
<name>990</name>
<location>SME VoIP</location>
</persondata>
</response>

```

(4) Response: FP has received SMS

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="17.6.15.1526" type="job">
  <externalid>0649396368</externalid>
  <status>1</status>
  <systemdata>
    <name>SME VoIP</name>
    <datetime>2017-06-20 04:12:16</datetime>
    <timestamp>5948e6f0</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <senderdata>
    <address>991</address>
    <name>991</name>
    <location>SME VoIP</location>
  </senderdata>
  <persondata>
    <address>990</address>
    <name>990</name>
    <location>SME VoIP</location>
  </persondata>
</response>
```

(5) Response: Handset has received SMS

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="17.6.15.1526" type="job">
  <externalid>0649396368</externalid>
  <systemdata>
    <name>SME VoIP</name>
    <datetime>2017-06-20 04:12:17</datetime>
    <timestamp>5948e6f1</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <jobdata>
    <priority>1</priority>
    <messages>
      <message1></message1>
      <message2></message2>
      <messageuui></messageuui>
    </messages>
    <status>1</status>
    <statusinfo></statusinfo>
  </jobdata>
  <senderdata>
    <address>991</address>
    <name>991</name>
    <location>SME VoIP</location>
  </senderdata>
  <persondata>
    <address>990</address>
    <name>990</name>
    <location>SME VoIP</location>
  </persondata>
</response>
```

(6) Response: handset has received SMS

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="17.6.15.1526" type="job">
<externalid>0649396368</externalid>
<systemdata>
<name>SME VoIP</name>
<datetime>2017-06-20 04:12:17</datetime>
<timestamp>5948e6f1</timestamp>
<status>1</status>
<statusinfo>System running</statusinfo>
</systemdata>
<jobdata>
<priority>1</priority>
<messages>
<message1></message1>
<message2></message2>
<messageuui></messageuui>
</messages>
<status>1</status>
<statusinfo></statusinfo>
</jobdata>
<senderdata>
<address>991</address>
<name>991</name>
<location>SME VoIP</location>
</senderdata>
<persondata>
<address>990</address>
<name>990</name>
<location>SME VoIP</location>
</persondata>
</response>
```

Alarm send from FP/handset to a MS

A handset can be configured so it sends an alarm to the MS for the following cases:

- 0: Man Down
- 1: No Movement
- 2: Running
- 3: Pull Cord
- 4: Red Key
- 5-9 Reserved

Furthermore the latest available beacon data is added to the alarm messages.

If no beacon UUID is available the field is empty.

beacontype can be:

- 0: Unknown
- 1: iBeacon
- 2: AltBeacon
- 3: Eddystone



broadcastdata: HEX encoded raw beacon BLE broadcast payload. Maximum 64 characters (32 bytes). In Figure 12, Figure 13 and Figure 14 are shown an overview of the broadcastdata for the 3 supported beacon types.

bdaddr; HEX encoded permanent Bluetooth address for the received beacon. Maximum 12 characters (6 bytes).

Below is shown the XML messages when a handset is sending an alarm for "man down". Notice the message type is alarm.

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="17.6.15.1526" type="alarm">
<externalid>0623295349</externalid>
<systemdata>
<name>SME VoIP</name>
<datetime>2017-06-20 05:55:51</datetime>
<timestamp>5948ff37</timestamp>
<status>1</status>
<statusinfo>System running</statusinfo>
</systemdata>
<alarmdata>
<type>0</type>
<beacontype>X</beacontype>
<broadcastdata>54235663af54235663af54235663af54235663af54235663af54235663af</broadcastdata>
<bdaddr>12AFCE98BEDE</bdaddr>
</alarmdata>
<senderdata>
<address>991</address>
<name>991</name>
<location>SME VoIP</location>
</senderdata>
</request>
```

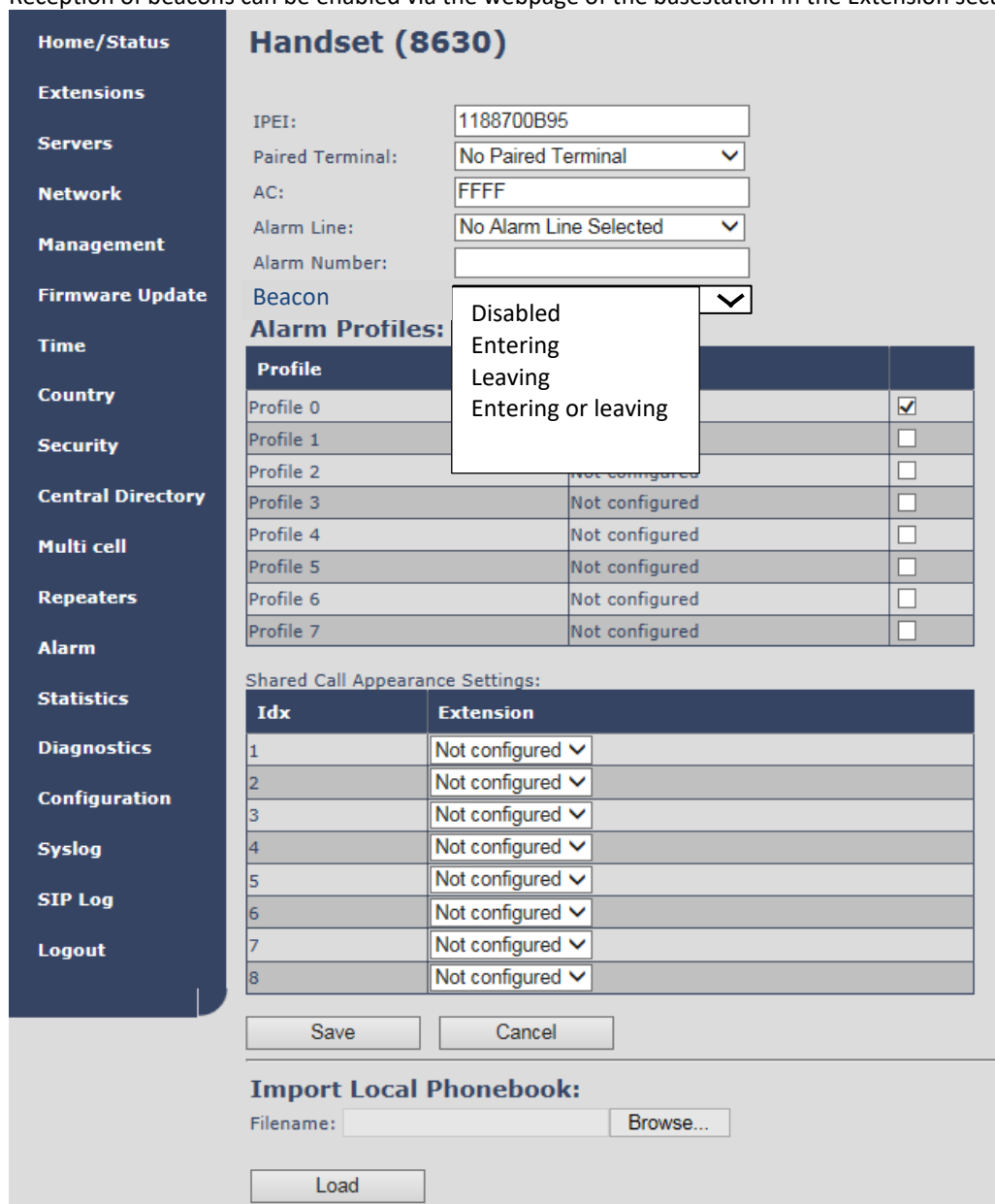
Response from the MS to the FP

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="v.3.4.7.1047" type="alarm">
<externalid>0623295349</externalid>
<status>1</status>
<statusinfo>Accepted by New Voice XML outgoing interface</statusinfo>
<systemdata>
<name>Mobical1</name>
<datetime>2017-06-20T12:55:55</datetime>
<timestamp>5948ff3b</timestamp>
<status>1</status>
<statusinfo>System running</statusinfo>
</systemdata>
</response>
```

Beacon message send from FP/handset to a MS

A handset can receive beacons messages from a beacon and send a message to the MS via the base with the information received.

Reception of beacons can be enabled via the webpage of the basestation in the Extension section:



Handset (8630)

IPEI:

Paired Terminal:

AC:

Alarm Line:

Alarm Number:

Beacon:

Alarm Profiles:

Profile		
Profile 0		<input checked="" type="checkbox"/>
Profile 1		<input type="checkbox"/>
Profile 2	Not configured	<input type="checkbox"/>
Profile 3	Not configured	<input type="checkbox"/>
Profile 4	Not configured	<input type="checkbox"/>
Profile 5	Not configured	<input type="checkbox"/>
Profile 6	Not configured	<input type="checkbox"/>
Profile 7	Not configured	<input type="checkbox"/>

Shared Call Appearance Settings:

Idx	Extension
1	Not configured
2	Not configured
3	Not configured
4	Not configured
5	Not configured
6	Not configured
7	Not configured
8	Not configured

Import Local Phonebook:

Filename:

The beacon alarm can be configured in 4 different ways when sending beacon alarms to the MS:

- When entering the proximity of a beacon
- When leaving the proximity of a beacon
- When entering, or leaving the proximity of a beacon
- Or disabled



Please read the specification: "BTLE smart Beacon alarms" for detailed information about when there is sent entering/leave messages, default thresholds etc.

Below is shown the XML messages when a handset/base is sending a beacon alarms.

The eventtype can be:

0: entering proximity of the beacon

1: leaving proximity of the beacon

beacontype can be:

0: Unknown

1: iBeacon

2: AltBeacon

3: Eddystone

broadcastdata: HEX encoded raw beacon BLE broadcast payload. Maximum 64 characters (32 bytes). In Figure 12, Figure 13 and Figure 14 are shown an overview of the broadcastdata for the 3 supported beacon types.

bdaddr; HEX encoded permanent Bluetooth address for the received beacon. Maximum 12 characters (6 bytes).

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="17.6.15.1526" type="beacon">
<externalid>0623295349</externalid>
<systemdata>
<name>SME VoIP</name>
<datetime>2017-06-20 05:55:51</datetime>
<timestamp>5948ff37</timestamp>
<status>1</status>
<statusinfo>System running</statusinfo>
</systemdata>
<beacondata>
<eventtype>0</eventtype>
<beacontype>X</beacontype>
<broadcastdata>54235663af54235663af54235663af54235663af54235663af54235663af</broadcastdata>
<bdaddr>12AFCE98BEDE</bdaddr>
</beacondata>
<senderdata>
<address>991</address>
<name>991</name>
<location>SME VoIP</location>
</senderdata>
</request>
```

Response from the MS to the FP

```

<?xml version="1.0" encoding="UTF-8"?>
<response version="v.3.4.7.1047" type="beacon">
<externalid>0623295349</externalid>
<status>1</status>
<statusinfo>Accepted by New Voice XML outgoing interface</statusinfo>
<systemdata>
<name>MessagesServerName</name>
<datetime>2017-06-20T12:55:55</datetime>
<timestamp>5948ff3b</timestamp>
<status>1</status>
<statusinfo>System running</statusinfo>
</systemdata>
</response>

```

BLE Advertising PDU Payload

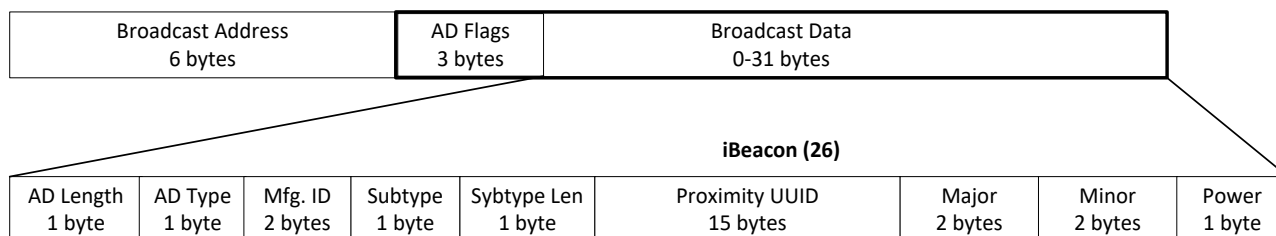


Figure 12: iBeacon

BLE Advertising PDU Payload

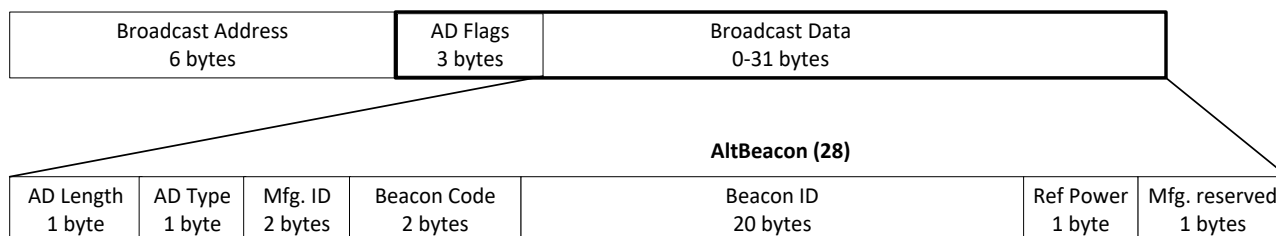


Figure 13: AltBeacon

BLE Advertising PDU Payload

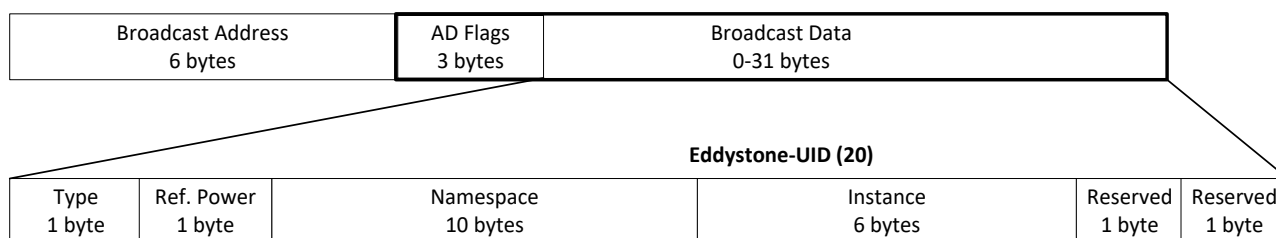


Figure 14: Eddystone beacon

Handset user interface for alarm reception

Receive first alarm

When the first alarm is received, the display will be shown as in Figure 15 if confirmation type is 2.

The middle softkey is 'OK' (Accept the alarm and moving the alarm to the list of alarms) and the right softkey is 'Reject' (Reject the alarm and delete the alarm).



Figure 15: Displaying a single alarm.

If the Alarm pop up is closed without pressing OK / Reject (e.g. if the hook on key is pressed) then the alarm is neither accepted or rejected.

For the one-alarm view the softkeys will be like this:

Alarm state	Left softkey	Center softkey	Right Softkey
Accepted	OK	-	Delete
None accepted / rejected		OK	Reject

For confirmation type 0 and 1 the softkeys will be like this:

Alarm state	Left softkey	Center softkey	Right Softkey
-	OK	-	Delete

Receive More than One Alarm

When more than one alarm has been received, then it is important that the user can see all pending alarms at the same time. This gives the user a chance to prioritize between all the pending alarms. In this case the UI is switched to the list the sorting will be such that none accepted alarms will be displayed first and then the priority and time is used.

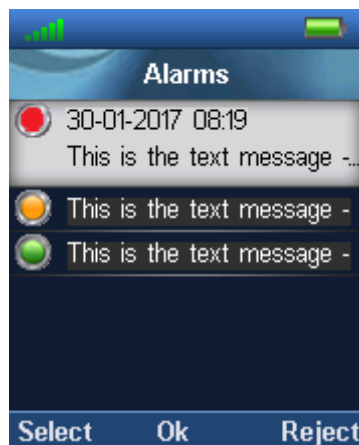


Figure 16: Displaying multiple alarms.

This list will display all pending and accepted alarms.

This list two types of items in the list:

- Alarms that has not been accepted or rejected.
- Alarm that has been accepted.

Alarms that has been rejected is not visible in any list, but are deleted immediately when they are rejected.

During display of the alarms, the window can be closed by using the hook on key.

When an item is highlighted, then the priority of the alarm (an icon that is either red, yellow or green) + timestamp + the first part of the alarm text is displayed. When a none-accepted / none-rejected alarm is displayed, then the softkeys will be 'Select', 'Ok' and 'Reject'. When an accepted alarm is displayed, then the softkeys will be 'Select' (left softkey) and 'Delete' (right softkey).

When an item is not highlighted, then priority of the alarm (an icon that is either red, yellow or green) + timestamp is displayed. Alarms that has not been read will be in bold.

From this list of alarms, then the user can directly Accept (pressing the middle softkey 'OK') or Reject (pressing the right softkey 'Reject') the highlighted alarm.

If the user wants to see the entire alarm text, then the left softkey 'Select' can be pressed. Then a UI component identical to the one used if only one alarm was present, will be displayed. If the alarm has been accepted, then the right softkey will 'Delete'. Otherwise the middle / right softkey will be / 'OK' / 'Reject'.

For the multiple-alarm view the softkeys will be like this:

Alarm state	Left softkey	Center softkey	Right Softkey
Accepted	Select	-	Delete
None accepted / rejected	Select	OK	Reject

While the alarm list is displayed, then the number of items can change. If a new alarm is received, then the alarm will be added to the list and if an alarm is Rejected, then the alarm will be deleted from the list. An item can also be deleted from the list, if the MS of the alarm cancels the alarm.

Handling the size of the Alarm List

The alarm list can contain up to 50 items.

When an alarm items is to be put into a full alarm list, then the user is presented with a pop up message with the text “memory is full”. A response is sent to the MS that the memory is full (status 3).

If the setting “overwrite old” is enabled (menu->envelope->settings) alarms that has read or confirmed (confirmation type2) will be overwritten when the alarm list is full. Oldest will be overwritten first.

Enter the Alarm List

The alarm list can at all time be displayed by pressing “Right” on the navigation key from idle or select the alarm symbol in the menu.

Receiving an Incoming

If an incoming call is received while displaying one of the two alarms views, then the alarm view will be hidden by the incoming call UE component. When the call is ended then the hidden alarm view will appear again.

Provisioning of beacon settings

Handset setup for beacon

In this chapter it is described which parameters that is transferred from the FP to the beacon handset application.

The handset has 2 modes for beacon, active and passive.

Active mode: Here is the handset acting as a beacon itself and transmits beacon information to the surroundings.

Passive mode: Here the handset is listening for beacon and report the received beacons to the messages server.

In Figure 17 and Figure 18 are the 2 modes shown.

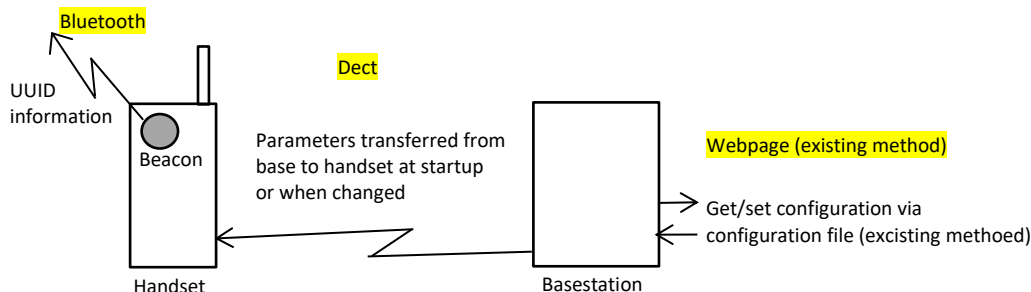


Figure 17: Active mode: handset is acting as a beacon itself.

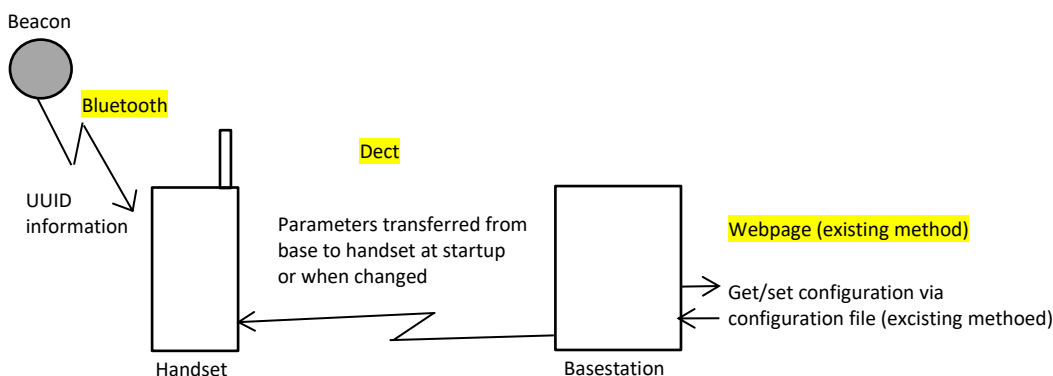


Figure 18: Passive mode. Handset is listening for beacons.

Parameters are stored in the base and transferred to the handset when the handset is located to the base (existing method).

Parameters are transferred from the base to the handset when they are changed on the base (existing method).

It is not possible to change the parameters in the handset.

Some parameters are only relevant for active mode and some only for passive mode. Below is listed the different parameters:

Table 1: Parameters transferred from base to the handset.

Name	Setting	Comment
Handset mode (2 bit)	Receive beacon, Transmit beacon, receive and Transmit beacon	1 selectable (drop down)
Passive mode		
Receive mode (2 bit)	report beacon information: Disabled, enter proximity, leave proximity, enter/leave proximity.	1 selectable (drop down)
Receive Sensitivity (2 bit)	Full, +8 or +16 dBm receiver sensitivity.	1 selectable (drop down)
Receive selection (4 bit) 1 extra bit reserved	Select which beacons to receive. iBeacon, AltBeacon, Eddystone.	All 3 can be selected
Active mode		
Transmit power (6 bit)	0 to -46dBm in 2 dB steps	1 selectable (drop down)
Transmit selection (2 bit)	Select which beacon type to transmit: iBeacon, AltBeacon, Eddystone.	1 selectable (drop down)
Transmit interval (2 bit)	Interval between each beacon transmit. 100ms 300ms 500ms	1 selectable (drop down)

In the multicell Voip System there is space for 1000 handsets. It is therefore important to use as little space as possible for the parameters.

It is therefore not recommended to store the broadcast for each handset as this will fill 32kByte in the base.

We should use the IPEI number as part of the UUID number together with a customer selectable value that is the same for all handsets.

Transmit beacon content (32 bytes)	Broadcast data of the transmitted beacon (space reserved 32 bytes) iBeacon: 26 bytes AltBeacon: 28 bytes EddyStone: 20 bytes	Input of up to 32 bytes
------------------------------------	---	-------------------------

The parameters are set following place in the webpage (same place as other parameters that are send to the handset at startup)

Home/Status
Extensions
Servers
Network
Management
Firmware Update
Time
Country
Security
Central Directory
Multi cell
Repeaters
Alarm
Statistics
Diagnostics
Configuration
Syslog
SIP Log
Logout

SME VoIP

Handset (8830)

IPEI:
Paired Terminal:
AC:
Alarm Line:
Alarm Number:

Beacon settings:

Setting	Selection
Handset mode	<input type="text" value="Receive beacon"/>
Receive mode	<input type="text" value="Leave proximity"/>
Receive sensivity	<input type="text" value="Full"/>
Receive selection	<input type="checkbox"/> iBeacon <input type="checkbox"/> Altbeacon <input type="checkbox"/> Eddystone
Transmit power	<input type="text" value="-24dBm"/>
Transmit interval	<input type="text" value="300ms"/>
Transmit selection	<input type="text" value="AltBeacon"/>
Transmit content	<input type="text" value="BE4578913FE779911347"/>

Alarm Profiles:

Profile	Alarm Type	
Profile 0	Alarm Button	<input checked="" type="checkbox"/>
Profile 1	Man Down	<input checked="" type="checkbox"/>
Profile 2	Not configured	<input type="checkbox"/>
Profile 3	Not configured	<input type="checkbox"/>
Profile 4	Not configured	<input type="checkbox"/>
Profile 5	Not configured	<input type="checkbox"/>
Profile 6	Not configured	<input type="checkbox"/>
Profile 7	Not configured	<input type="checkbox"/>

Shared Call Appearance Settings:

Idx	Extension
1	<input type="text" value="Not configured"/>
2	<input type="text" value="Not configured"/>
3	<input type="text" value="Not configured"/>
4	<input type="text" value="Not configured"/>
5	<input type="text" value="Not configured"/>
6	<input type="text" value="Not configured"/>
7	<input type="text" value="Not configured"/>
8	<input type="text" value="Not configured"/>

Import Local Phonebook:

Filename: No file chosen

Home/Status

Extensions

Servers

Network

Management

Firmware Update

Time

Country

Security

Central Directory

Multi cell

Repeaters

Alarm

Statistics

Diagnostics

Configuration

Syslog

SIP Log

Logout

SME VoIP

Handset (8830)

IPEI:

Paired Terminal:

AC:

Alarm Line:

Alarm Number:

Alarm Profiles:

Profile	Alarm Type	
Profile 0	Alarm Button	<input checked="" type="checkbox"/>
Profile 1	Man Down	<input checked="" type="checkbox"/>
Profile 2	Not configured	<input type="checkbox"/>
Profile 3	Not configured	<input type="checkbox"/>
Profile 4	Not configured	<input type="checkbox"/>
Profile 5	Not configured	<input type="checkbox"/>
Profile 6	Not configured	<input type="checkbox"/>
Profile 7	Not configured	<input type="checkbox"/>

Shared Call Appearance Settings:

Idx	Extension
1	<input type="text" value="Not configured"/>
2	<input type="text" value="Not configured"/>
3	<input type="text" value="Not configured"/>
4	<input type="text" value="Not configured"/>
5	<input type="text" value="Not configured"/>
6	<input type="text" value="Not configured"/>
7	<input type="text" value="Not configured"/>
8	<input type="text" value="Not configured"/>

Import Local Phonebook:

Filename: No file chosen

Home/Status

Extensions

Servers

Network

Management

Firmware Update

Time

Country

Security

Central Directory

Multi cell

Handset (8630)

IPEI:

Paired Terminal:

AC:

Alarm Line:

Alarm Number:

Beacon

Alarm Profiles:

Profile		
Profile 0		<input checked="" type="checkbox"/>
Profile 1		<input type="checkbox"/>
Profile 2	Not configured	<input type="checkbox"/>
Profile 3	Not configured	<input type="checkbox"/>
Profile 4	Not configured	<input type="checkbox"/>
Profile 5	Not configured	<input type="checkbox"/>

RTX8200 setup

The RTX8200 has also as the handset 2 mode, active or passive.

The setup is almost done the same way for the RTX8200 as for the handset except for the following differences.

The setting for the RTX8200 can be setup either via the base or via a Bluetooth interface. In Figure 19 and Figure 20 the 2 ways are shown.

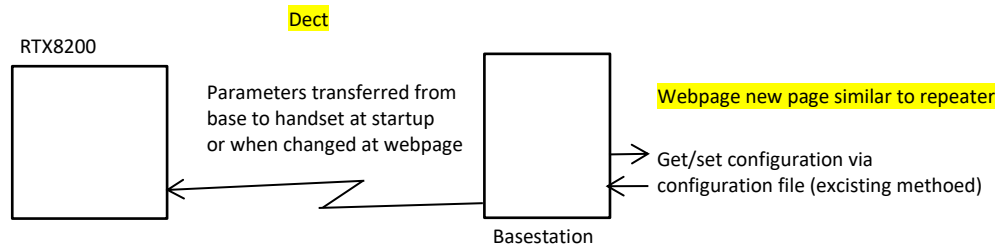


Figure 19: Setting parameters in the RTX8200 from the base.

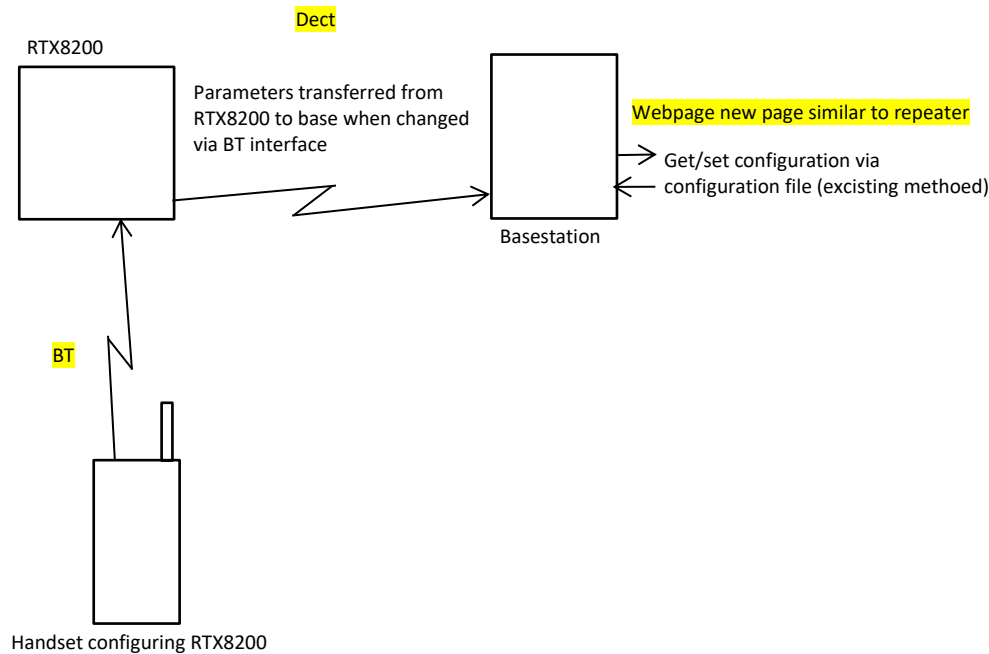


Figure 20: Setting parameters in the RTX8200 via a BT interface from a handset.

When the RTX8200 are updated via the BT interface the parameters are pushed back to the base and stored in the base NVS, so they are used next time the RTX8200 is restarted.

The RTX8200 has the same parameters as the handset (see Table 1) except that the antenna beamform can be set.

Name	Setting	
Antenna beamform	3 choises: left, right, cross	

Beacon test interface

There has been developed a AT interface for testing interface between the handset and base/message server

This test interface is emulating the application that will normally send the beacon request to the sms module in the handset.

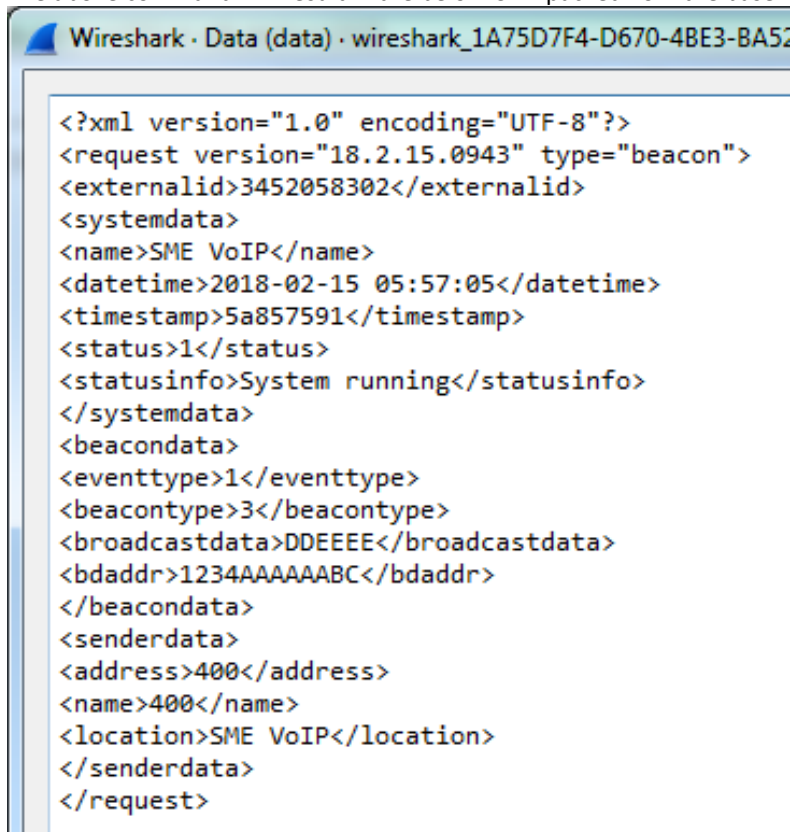
Parameters: - remember that all input is in HEX

rsbool **ContinuousMode** - True: there will be send a beacon at each timeinterval. False: there is only send 1 beacon.
 rsuint16 **TimeInterval** - Interval in ms between each beacon request.
 rsbool **KeepCissOpen** - True if the ciss connection should be left open.
 rsuint16 **BeaconRefNum** - Reference number for the beacon.
 rsuint8 **eventtype** - Eventtype: 0: enter. 1: leaving.
 rsuint8 **beaconType** - BeaconType: 0: Unknown, 1: iBeacon, 2:AltBeacon , 3:Eddystone
 rsuint8 **bdaddr[12]** - BT MAC address, always 12 characters.
 rsuint8 **broadcastLen** - Length of broadcastdata. Max 64 characters
 rsuint8 **broadcastdata[1]** - broadcastdata from the beacon.

Example of AT command:

BeaconAutotest.bat 0 1000 1 9 1 3 31 32 33 34 41 41 41 41 41 41 41 42 43 6 44 44 45 45 45 45

The above command will result in the below UDP packet from the base:

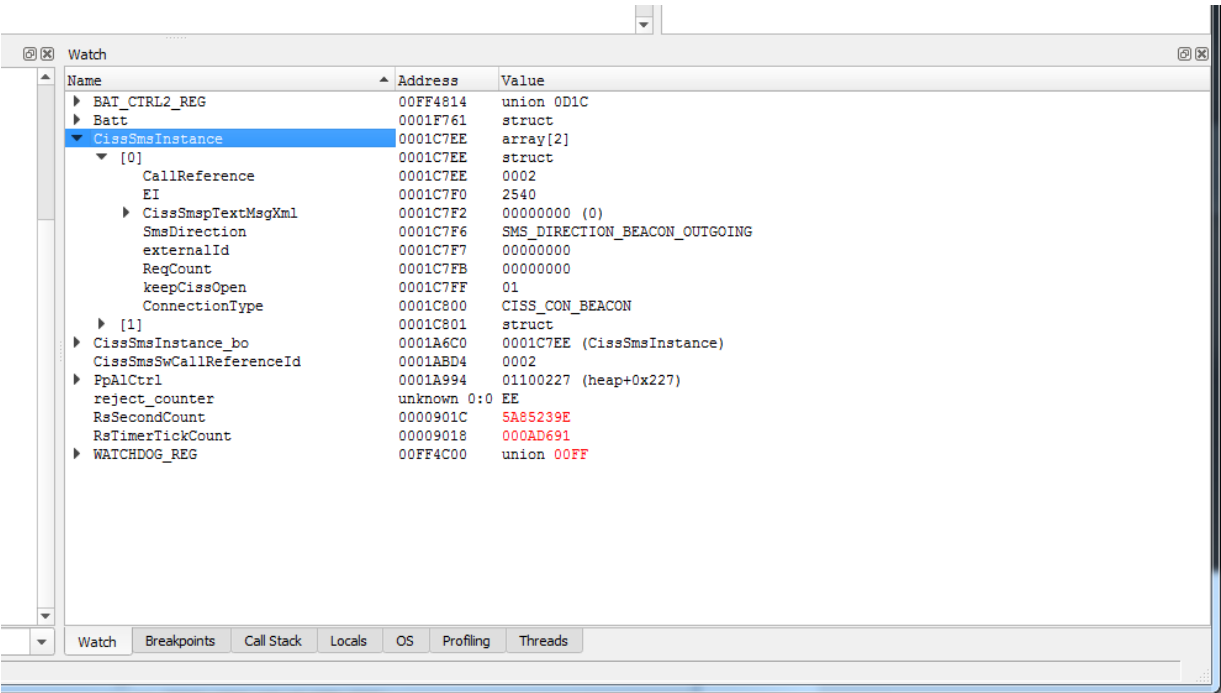


```

Wireshark · Data (data) · wireshark_1A75D7F4-D670-4BE3-BA52
<?xml version="1.0" encoding="UTF-8"?>
<request version="18.2.15.0943" type="beacon">
  <externalid>3452058302</externalid>
  <systemdata>
    <name>SME VoIP</name>
    <datetime>2018-02-15 05:57:05</datetime>
    <timestamp>5a857591</timestamp>
    <status>1</status>
    <statusinfo>System running</statusinfo>
  </systemdata>
  <beacondata>
    <eventtype>1</eventtype>
    <beacontype>3</beacontype>
    <broadcastdata>DDEEEE</broadcastdata>
    <bdaddr>1234AAAAAABC</bdaddr>
  </beacondata>
  <senderdata>
    <address>400</address>
    <name>400</name>
    <location>SME VoIP</location>
  </senderdata>
</request>

```

When testing with the test interface it is interesting to watch the CissSmsInstance, where the ReqCount is indicating how many outstanding requests there is currently in the handset.



Interface specification for transfer data from handset to base via CISS

Below is described the interface between the handset application handling Beacons to SMS module in the handset. In Figure 21 to Figure 25 is shown how the application in interfacing with the SMS module in the handset.

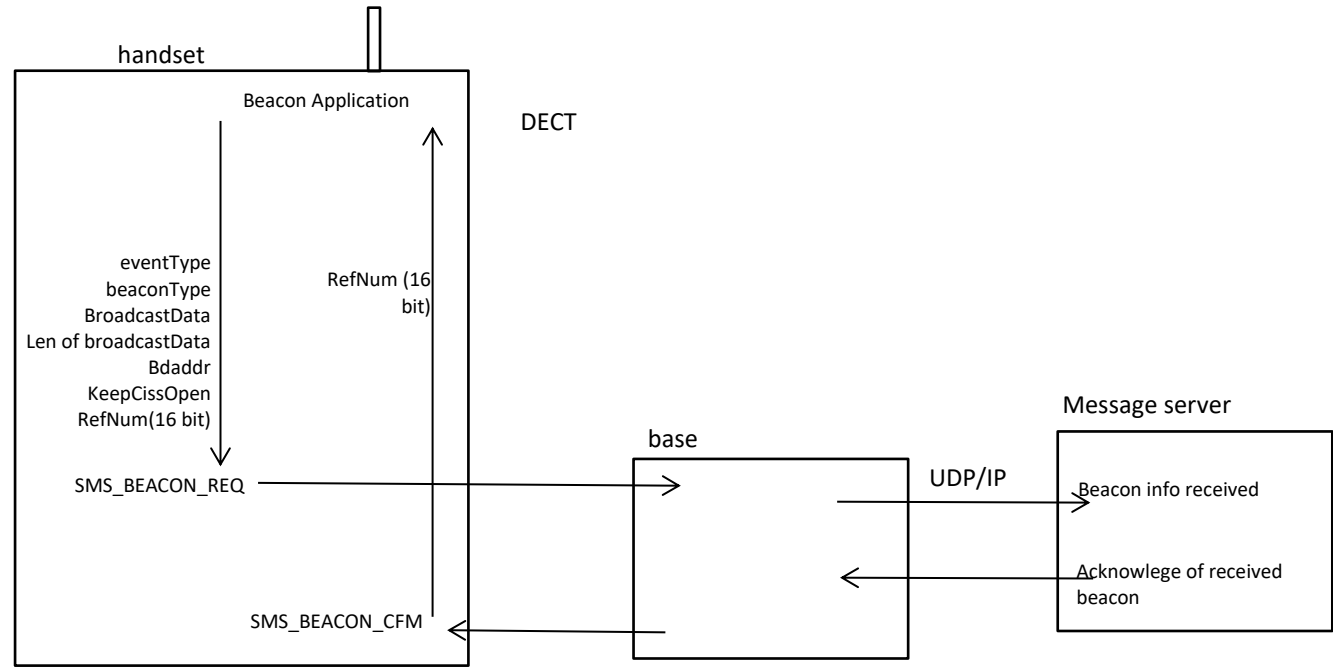


Figure 21: Interface between beacon application and the SMS module

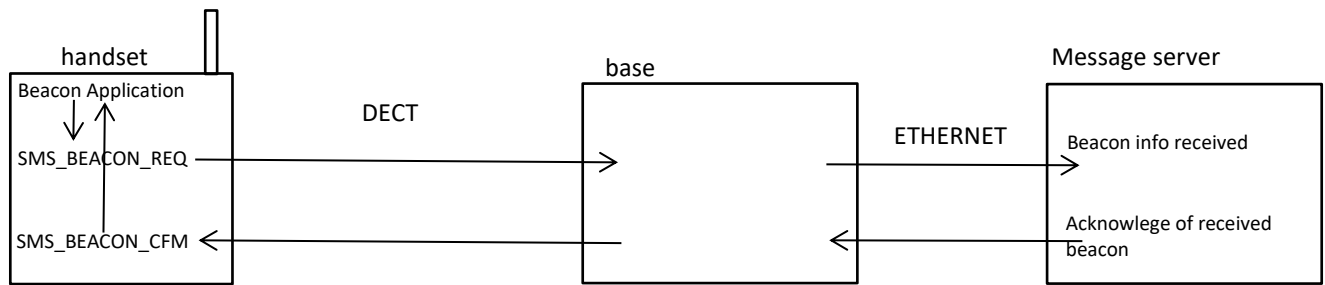


Figure 22: Successful beacon send to a message server.

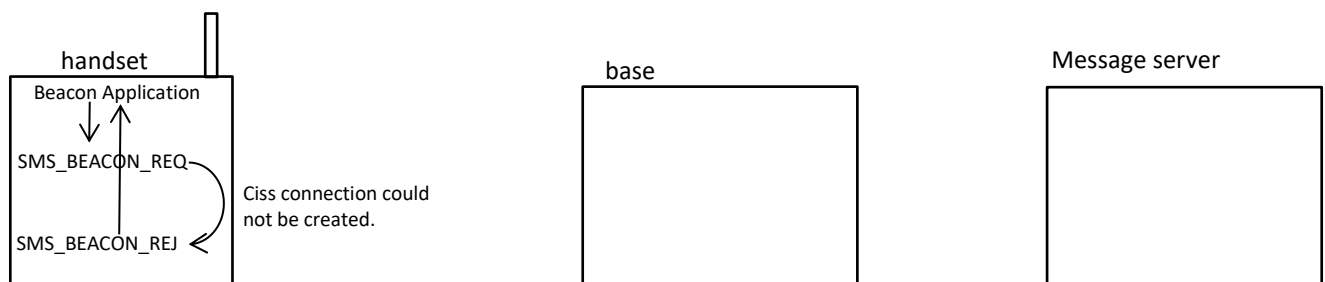


Figure 23: Creating Ciss connection from handset to base failed.

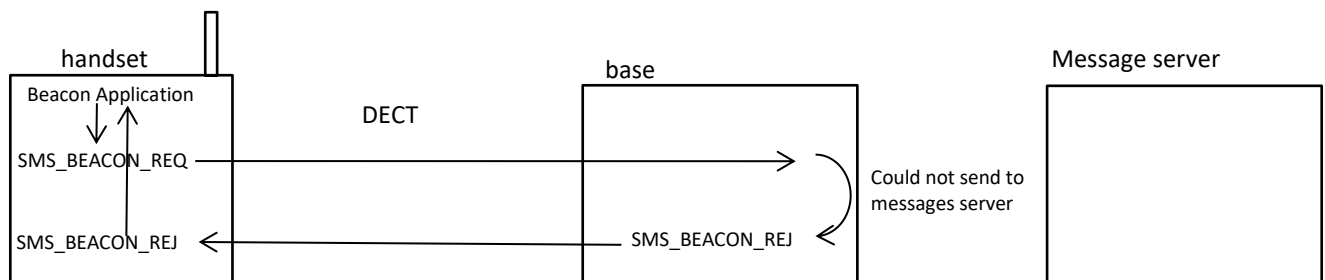


Figure 24: Sending beacon from base to messages server failed.

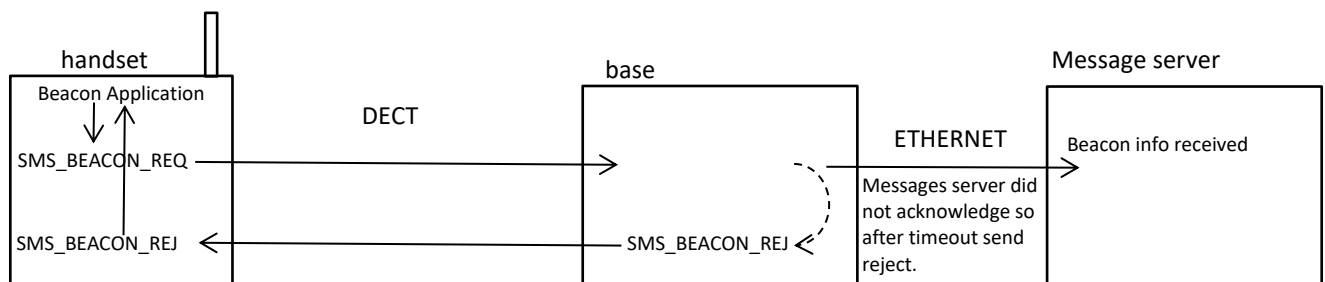


Figure 25: Messages server did not acknowledge the received beacon.

The interface header is placed in sms.h:

```
void SendSmsBeaconReq(RosTaskIdType Src, rsuint8 eventype, BeaconType beacontype, rsuint8 *broadcastData, rsuint8 broadcastDataLen, rsuint8 *bdaddr, rsbool KeepCissOpen, rsuint16 beaconRefNum);
```

```

* eventType:
* 0: entering proximity of the beacon
* 1: leaving proximity of the beacon

* beaconType:
* BEACON_TYPE_UNKNOWN    = 0x00,
* BEACON_TYPE_IBEAICON   = 0x01,
* BEACON_TYPE_ALTBEAICON = 0x02
* BEACON_TYPE_EDDYSTONE  = 0x03
*
* broadcastData:
* (max 32 bytes -will be converted to a 64 byte string in the base before send to the messages server)
*
*
* broadcastDataLen:
* length of broadcastData.
*
* bdaddr:
* always 6 bytes - -will be converted to a 12 byte string in the base before send to the messages server
*
* KeepCissOpen:
* TRUE: the ciss connection is keep open until informed by the application to be closed
* FALSE: the ciss connection is closed when all outstanding request has either been confirmed or rejected ( there is a counter for request and confirm/reject messages)

* beaconRefNum:
* reference number that will be returned in the confirm or reject mail
* it is not allowed to use reference number 0xFFFF as this is reserved, see more info in SendBeaconSendRej

```

The structure for confirmation SMS_BEACON_CFM is:

```

typedef struct SmsSendBeaconCfmType
{
    RosPrimitiveType Primitive;    /*!< SMS_BEACON_CFM */
    rsuint16 beaconRefNum;        /*!< Reference number of the beacon that has been sent from the application to the SMS module. Used for
identification in CFM and REJ messages */
} SmsSendBeaconCfmType;

```

The structure for rejection SMS_BEACON_REJ is:

```

typedef struct SmsSendBeaconRejType
{
    RosPrimitiveType Primitive;    /*!< SMS_BEACON_REJ */
    SmsRejectReasonType RejectReason; /*!< Reason that the message is rejected. */
    rsbool Resend;                /*!<Info if the information should be resend */
    rsuint16 beaconRefNum;        /*!< Reference number of the beacon that has been sent from the application to the SMS module. Used for
identification in CFM and REJ messages */
} SmsSendBeaconRejType;

```

Special case for beaconRefNum: 0xFFFF

If the handset is out of range of the basestation (in searching mode) or if all resources are used (8 ongoing calls on the basestation) the beaconRefNum is 0xFFFF in SMS_BEACON_REJ. When this happens all beacon requests that has yet not been confirmed or rejected should be resend. It is recommended to wait a little time before trying to resend.

Even though there have been several requests there will only be send one SendBeaconSendRej with reference number 0xFFFF.

Timing constrains

If the message server does not answer the request it takes some time for the base to timeout the beacon request from the handset. Every request that is not confirmed or rejected in the handset will fill the heap until it is rejected/confirmed. In case message server does not respond it takes some time before the base is sending a reject to the handset. In case the application continues to receive reject from the base the application should not send request at a higher interval than 100 ms between each other, otherwise the heap will run full at the handset. This has only been tested with Rocket/Beatus and could be differently for other combinations.

Furthermore, there has been created a new interface for sending alarms where the latest beacon broadcast data can be included in the messages:

```

void SendSmsAlarmAndBeaconReq(RosTaskIdType Src, EmergencyTypeOfAlarmType Alarm, BeaconType beaconType, rsuint8 *broadcastData,
rsuint8 broadcastDataLen, rsuint8 *bdaddr);

```

```

* beaconType:
* BEACON_TYPE_UNKNOWN    = 0x00,
* BEACON_TYPE_IBEAICON   = 0x01,

```



- * BEACON_TYPE_ALTBACON = 0x02
- * BEACON_TYPE_EDDYSTONE = 0x03
- *
- * broadcastData:
- * (max 32 bytes -will be converted to a 64 byte string in the base before send to the messages server)
- *
- *
- * broadcastDataLen:
- * length of broadcastData.
- *
- * bdaddr:
- * always 6 bytes -will be converted to a 12 byte string in the base before send to the messages server

Open issues to discuss

How should the handset react in case the beacon is not received in the messages server – pop up on the screen ?
When an alarm is triggered should we automatically turn on reception of beacons

Test:

when sending SMS_ALARM_REQ more often than every 64ms will slowly increase the heap on Rocket.

Auto test

The following is auto test that is used for Jenkins/robot framework.

Alarm test 1 – ConfirmationType 0

Inputs:

Priority: 1

ReferenceNo:1

Confirmationtype: 0

Text: Alarm test 1

Status: 0

ExternalId:1

Verify:

that only 1 response is received from the base

delete the alarm and verify that there are 0 alarms in the list

Alarm test 2 – ConfirmationType 1

Inputs:

Priority: 1

ReferenceNo:1

Confirmationtype: 1

Text: Alarm test 1

Status: 0

ExternalId:1

Verify:

that 2 responses are received from the base

delete the alarm and verify that there are 0 alarms in the list

Alarm test 3 – ConfirmationType 2 - OK

Inputs:

Priority: 1

ReferenceNo:1

Confirmationtype: 2

Text: Alarm test 1

Status: 0

ExternalId:1

Verify:

that 2 responses are received from the base

Inputs:

Press ok

Verify:

that 1 responses are received from the base

delete alarm and verify that the alarm list is empty

Alarm test 4 – ConfirmationType 2 - reject

Inputs:

Priority: 1

ReferenceNo:1

Confirmationtype: 2

Text: Alarm test 1

Status: 0

ExternalId:1

Verify:

that 2 responses are received from the base

Inputs:

Press reject

Verify:

that 1 responses are received from the base

verify that the alarm list is empty

Alarm test 5 – ConfirmationType 2 – substitute Alarm

Inputs:

Priority: 1

ReferenceNo:1

Confirmationtype: 2

Text: Alarm test 1

Status: 0

ExternalId:1

Verify:

that 2 responses are received from the base

the text received in the handset

Inputs:

Priority: 1

ReferenceNo:1

Confirmationtype: 2

Text: Alarm test 1 é NEW alarm

Status: 0

ExternalId:15

Verify:

that 2 responses are received from the base

that there is only 1 alarm in the list

delete alarm and verify that the alarm list is empty

Alarm test 6 – confirmation type 1 -delete

Inputs:

Priority: 1

ReferenceNo:1

Confirmationtype: 1

Text: Alarm test 1

Status: 0

ExternalId:1

Verify:

that 2 responses are received from the base

that there is 1 alarm in the list

Input (delete alarm):

Priority: 1

ReferenceNo:1

Confirmationtype: 1

Text: Alarm test 1

Status: 10

ExternalId:

Verify:

that 2 responses are received from the base

that there is 1 alarm in the list

delete alarm and verify that there are 0 alarms in the alarm list

Alarm test 7 – confirmation type 2 -delete

Inputs:

Priority: 1

ReferenceNo:1

Confirmationtype: 2

Text: Alarm test 1

Status: 0

ExternalId:1

Verify:

that 2 responses are received from the base

that there is 1 alarm in the list

Inputs2 (delete alarm):

Priority: 1

ReferenceNo:1

Confirmationtype: 2

Text: Alarm test 1

Status: 10

ExternalId:

Verify:

that 2 responses are received from the base

that there is 0 alarm in the list

Alarm test 8 – 50 alarms confirmation type 2

50 alarms with

Inputs:

Priority: 1
ReferenceNo:1
Confirmationtype: 2
Text: Alarm test 1
Status: 0
ExternalId:1

Verify:

that there are 50 alarms in the list
delete 49 alarms and verify that there is 1 alarm left
delete 1 alarm and verify that there is 1 alarm left

SMS test 1 – confirmation type 0

Inputs:

Priority: 0
Sender: 888
Confirmationtype: 0
Text: Sms test 1
Status: 0
ExternalId:1

Verify:

The received text
The received sender
That there is 1 SMS. Delete the SMS and verify that there is 0 SMS

SMS test 2 – confirmation type 1 - OK

Inputs:

Priority: 0
Sender: 888
Confirmationtype: 1
Text: Sms test 1
Status: 0
ExternalId:1

Input 2:

Press OK

Verify:

There is received a status 4 response
The received text
The received sender
That there is 1 SMS. Delete the SMS and verify that there is 0 SMS

SMS test 3 – confirmation type 1 - REJECT

Inputs:

Priority: 0
Sender: 888
Confirmationtype: 1
Text: Sms test 1



Status: 0
ExternalId:1

Input 2:
Press REJECT

Verify:
There is received a status 5 response
The received text
The received sender
That there is 1 SMS. Delete the SMS and verify that there is 0 SMS

SMS test 4 – 50 SMS confirmation type 0

Sent 50 SMS to the handset

Inputs:
Priority: 0
Sender: 888
Confirmationtype: 0
Text: Sms test 1
Status: 0
ExternalId:1

Verify:
that there are 50 SMS in the list
delete all 50 SMS with “delete all” and verify that there is 0 SMS left