AN994: BLUETOOTH PBAP AND MAP PROFILES

iWRAP APPLICATION NOTE

Monday, 18 March 2019

Version 2.6

VERSION HISTORY

Version	Comment
1.0	First version
1.5	Revision
1.8	Further revision
1.9	More addition and revision
2.0	Made changes to formatting, modified UCD
2.1	Added MAP documentation
2.2	Added Message Notification Service documentation
2.3	Added MAP filtering
2.4	A few style and typo fixes
2.5	Improvements to MAP and MSG commands
2.6	Updated references

TABLE OF CONTENTS

1 iWRAP firmware overview

iWRAP is an embedded firmware running entirely on the RISC processor of WT11i, WT12, WT32, WT32i and WT41 modules. It implements the full *Bluetooth* protocol stack and many *Bluetooth* profiles as well. All software layers, including application software, run on the internal RISC processor in a protected user software execution environment known as a Virtual Machine (VM).

The host system can interface to iWRAP firmware through one or more physical interfaces, which are also shown in the figure below. The most common interfacing is done through the UART interface by using the ASCII commands that iWRAP firmware supports. With these ASCII commands, the host can access *Bluetooth* functionality without paying any attention to the complexity, which lies in the *Bluetooth* protocol stack. GPIO interface can be used for event monitoring and command execution. PCM, SPDIF, I2S or analog interfaces are available for audio. The available interfaces depend on the used hardware.

The user can write application code to the host processor to control iWRAP firmware using ASCII commands or GPIO events. In this way, it is easy to develop *Bluetooth* enabled applications.

On WT32 and WT32i there is an extra DSP processor available for data/audio processing.

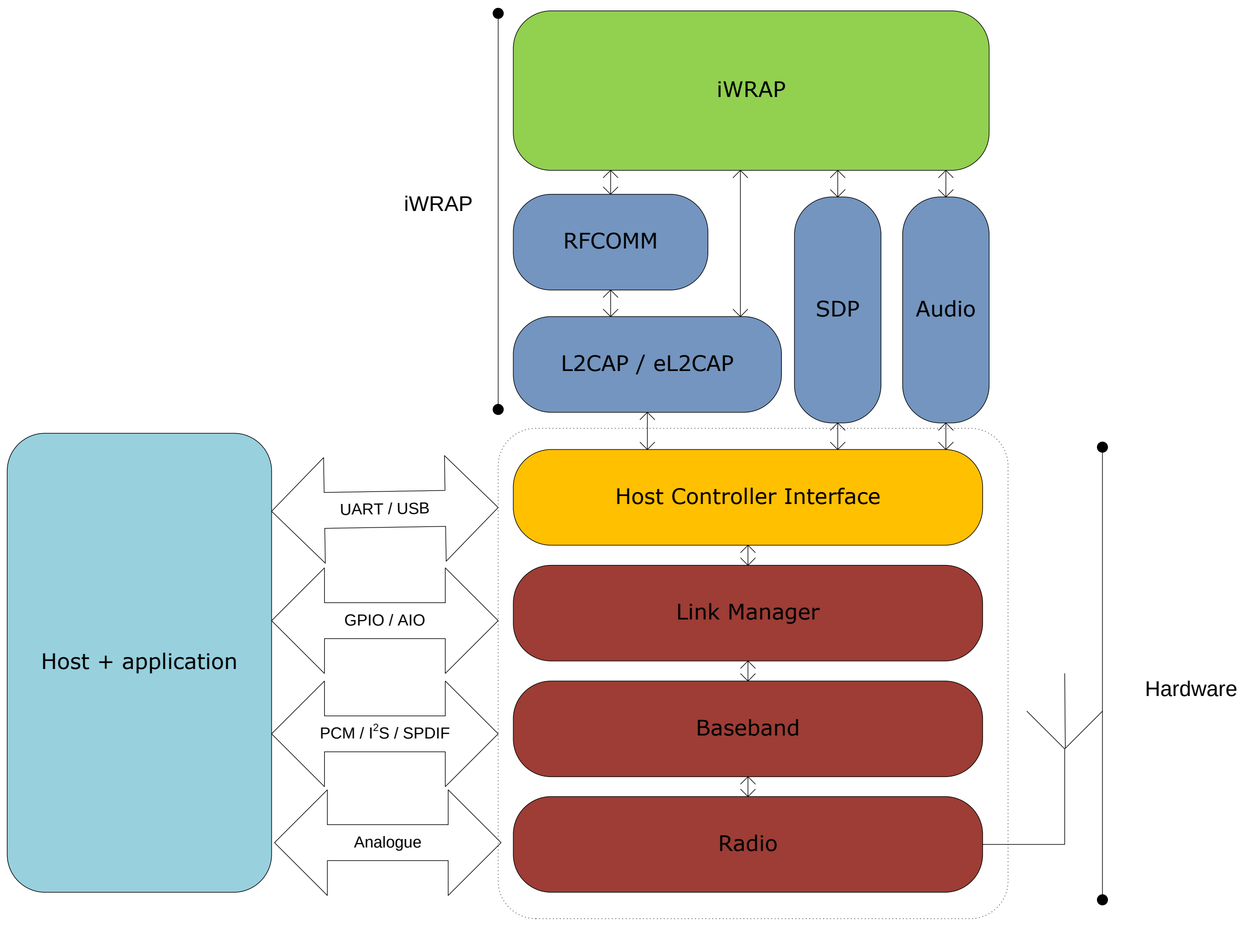


Figure 1: iWRAP Bluetooth stack In the figure above, a Bluegiga *Bluetooth* module with iWRAP firmware could be connected to a host system for example through the UART interface. The options are:

- If the host system has a processor, software can be used to control iWRAP by using ASCII based commands or GPIO events.
- If there is no need to control iWRAP, or the host system does not need a processor, iWRAP can be configured to be totally transparent and autonomous, in which case it only accepts connections or automatically opens them.
- GPIO lines that WRAP THOR modules offer can also be used together with iWRAP to achieve additional functionality, such as Carrier Detect or DTR signaling.

2 Introduction

This application note describes the usage of Phone Book Access Profile (PBAP) and Message Access Profile (MAP). Several practical use case examples are included for both profiles.

2.1 Phone Book Access Profile

The Phone Book Access Profile (PBAP) specification defines the procedures and protocols to exchange Phone Book objects between devices. It is especially tailored for the automotive Hands-Free use case where an onboard terminal device (typically a Car-Kit installed in the car) retrieves Phone Book objects from a mobile device (typically a mobile phone or an embedded phone). This profile may also be used by any client device that requires access to Phone Book object is stored in a server device.

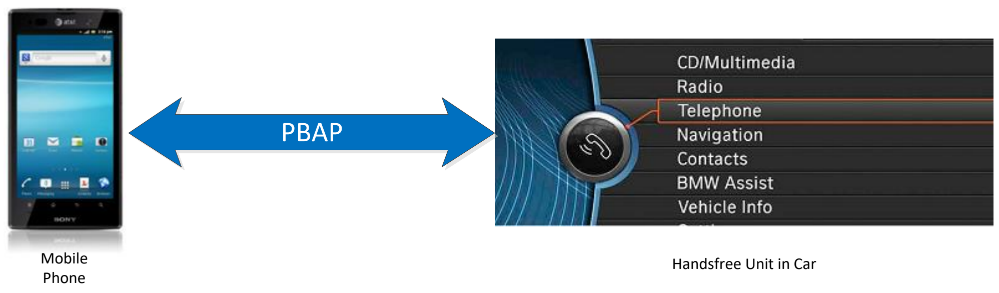


Figure 2: Typical PBAP Use Case

The PBAP is based on a Client-Server interaction model where the Client device pulls phone book objects from the Server device. In Figure 1 for example, the client is the hands-free unit in the car and the server is the mobile phone. In many of today's automobiles, this PBAP functionality is enabled and integrated within the factory car audio system.

Please note however that this profile only allows for the consultation of phone book objects (read-only). It is not possible to alter the content of the original phone book object (read/write). Phone Book Access Profile is dependent upon the Generic Object Exchange Profile, the OBEX Object Exchange protocol, RFCOMM protocol, and the Generic Access Profile.

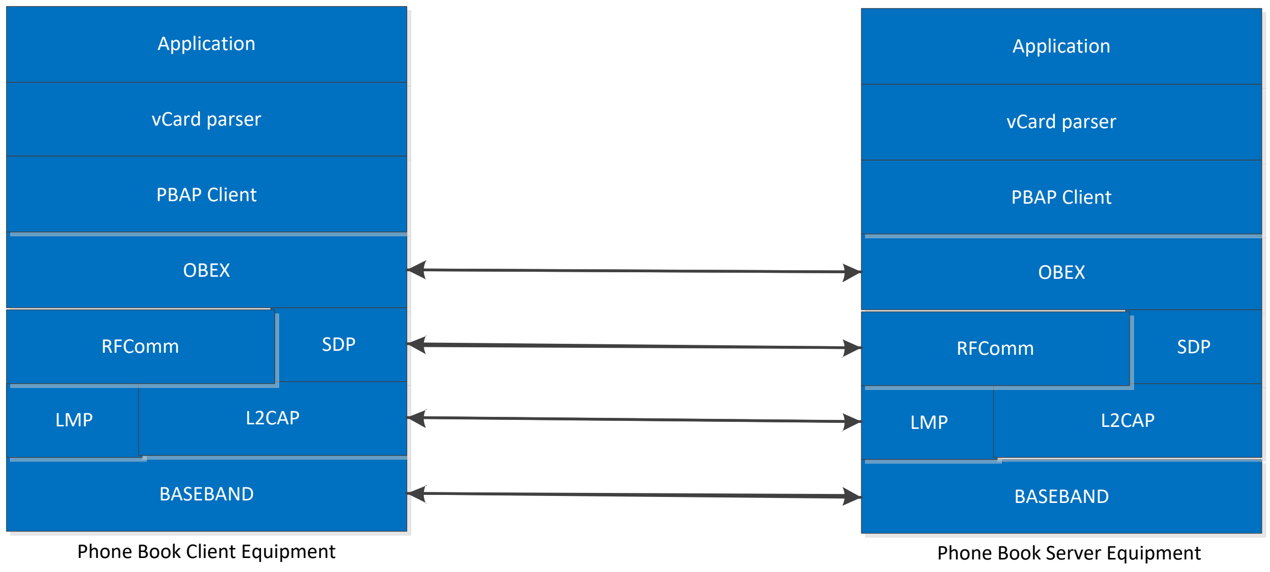


Figure 3: PBAP Profile Stack

2.2 Message Access Profile

Message Access Profile (MAP) is often used in similar use cases as PBAP. A typical scenario is a hands-free car kit, which in addition to controlling the telephony functions of the mobile phone, can also display incoming SMS or email messages. MAP can also be used to exchange received SMS or email messages, for example between a phone and PC.

MAP is based on a client-server-model. The client (Message Client Equipment or MCE) retrieves messages from the server (Message Server Equipment or MSE). Like PBAP, MAP works on top of the OBEX protocol. The MAP implementation of iWRAP can only list and retrieve messages from the MSE; delete and message push functionalities are not supported by iWRAP.

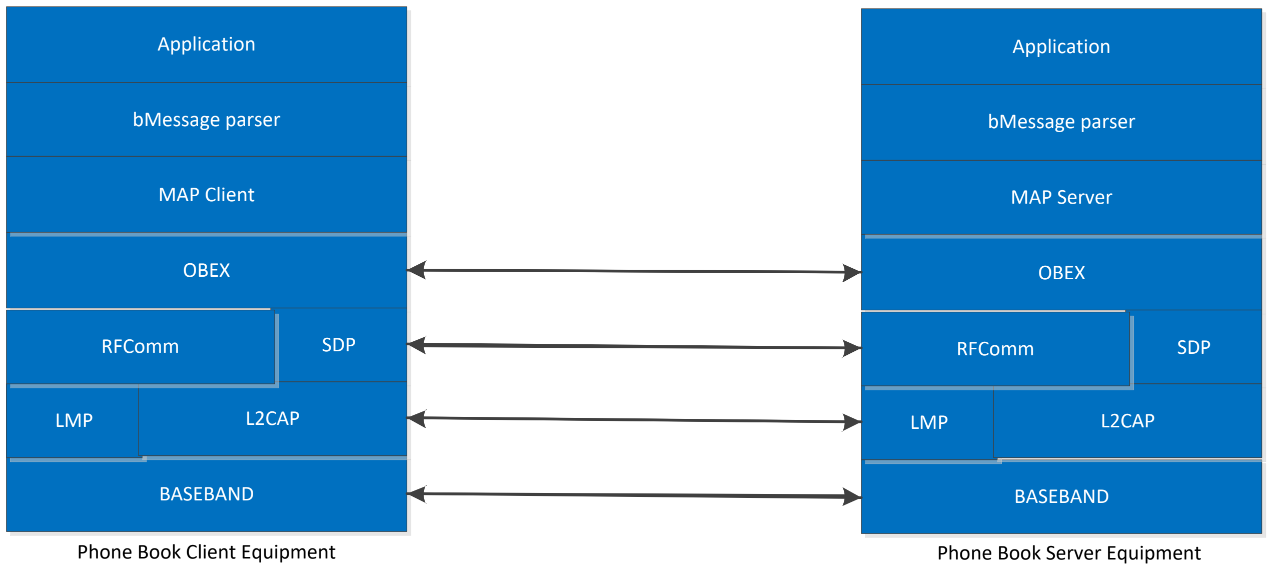


Figure 4: MAP Profile Stack

3 Using PBAP with iWRAP

This chapter instructs the PBAP usage and configuration with the iWRAP firmware.

3.1 Profile configuration

The PBAP profile must be turned on along with the HFP profile. In iWRAP, HFP profile is needed as it works hand in hand with PBAP profile to retrieve phone book objects from the Phone Server Equipment (PSE). For more info on the HFP profile, please refer to the HFP/HSP Application note. PBAP is turned on by issuing "**SET PROFILE PBAP ON**".

```
SET PROFILE PBAP ON
SET PROFILE HFP ON
SET BT CLASS 200408
SET BT NAME {Module Friendly Name}
SET BT AUTH * {0-16 DIGIT PIN}
RESET
```

As can be seen in the box above, a reset command is issued to make the profile(s) and configurations active.

Note:

The PIN code can be set from 0-16 characters. Also, starting from iWRAP5, Secure Simple Pairing (SSP) is enabled by default to fulfil Bluetooth 2.1 specification requirements. It is not possible to disable SSP in iWRAP5 and if the remote device does not support SSP, iWRAP will automatically fall back to legacy pairing through PIN code. For more info on SSP, please refer to the extensive documentation in iWRAP User Guide.

3.2 Service discovery

Bluetooth technology enables wireless service discovery, so you can find out the capabilities the remote device supports. Wireless service discovery uses the Bluetooth Service Discovery Profile (SDP).

With iWRAP the service discovery is performed with command: "**SDP {*bd_addr*} {*uuid*}**".

bd_addr	Bluetooth device address of the remote device.
uuid	Universally unique identifier, which refers to the Bluetooth profile or service to be discovered. For PBAP server the ***uuid*** is 112f. Please refer to the iWRAP User Guide for ***uuid*** of PBAP PCE and PSE.

Below is an example how to perform a service discovery for PBAP device.

```
SDP 00:07:80:aa:bb:cc 112F
SDP 00:07:80:aa:bb:cc < I SERVICENAME S "OBEX Phonebook Access Server" > < I PROTOCOLDESCRIPTORLIST < < U L2CAP > < U RFCOMM I 13 > < U OBEX > > >
SDP
```

OBEX Phonebook Access Server	= Service name
13	= RFCOMM channel for PBAP server

3.3 Connection establishment

The PCE is usually the device that initiates the connection. If connecting to a device for the first time, iWRAP will automatically pair with the device before starting the call procedure.

The PBAP connection is established with the following command:

"**CALL {*bd_addr*} 112F PBAP**"

bd_addr	Bluetooth device address of the remote device.

Below is an example how to set up a PBAP from iWRAP to a PSE device.

```
CALL 00:07:80:aa:bb:cc 112F PBAP
CALL 0
CONNECT 0 PBAP 5
OBEX 0 READY
```

Note:

If the devices have not been previously paired, iWRAP will do the pairing process during the connection establishment. A PAIR event may be received (depending on the SET CONTROL CONFIG bits set) prior to receiving the CONNECT event.

3.4 PBAP command

PBAP command is used to retrieve phone book entries or call history from a PBAP PSE device.

3.4.1 Syntax

Synopsis
PBAP {*path*} {*count*} [*offset*] [*filter*] [*format*]

Description		
path	**Left HEX** Store to retrieve data from. **0** Phone **1** SIM card	**Right HEX** Phone book or call history to read. **0** Phonebook **1** Incoming call history **2** Outgoing call history **3** Missed call history **4** Combined call history
count	Number of entries to be retrieved. **0** Returns phone book size **FFFF** Retrieves all entries	
offset	Offsets from which to start the retrieve from.	
filter	This is a bit mask to filter the response. If this is left to 0 all fields will be returned. Mandatory attributes for vCard 2.1 are VERSION ,N and TEL and they are returned always. Mandatory attributes for vCard 3.0 are VERSION, N, FN and TEL and they are also returned always.	

	bit 0 VERSION vCard Version **Bit 1** FN Formatted PBAP Name **bit 2** N Structured Presentation of Name **bit 3** PHOTO Associated Image or Photo **bit 4** BDAY Birthday **bit 5** ADR Delivery Address **bit 6** LABEL Delivery **bit 7** TEL Telephone Number **bit 8** EMAIL Electronic Mail Address **bit 9** MAILER Electronic Mail **bit 10** TZ Time Zone **bit 11** GEO Geographic Position **bit 12** TITLE Job **bit 13** ROLE Role within the Organization **bit 14** LOGO Organization Logo **bit 15** AGENT vCard of Person Representing

	bit 16 ORG Name of Organization **bit 17** NOTE Comments **bit 18** REV Revision **bit 19** SOUND Pronunciation of Name **bit 20** URL Uniform Resource Locator **bit 21** UID Unique ID **bit 22** KEY Public Encryption Key **bit 23** NICKNAME Nickname **bit 24** CATEGORIES Categories **bit 25** PROID Product ID **bit 26** CLASS Class information **bit 27** SORT-STRING String used for sorting operations **bit 28** X-IRMC-CALL-DATETIME Time stamp
format	**1** Return vCard 3.0 **0** Return vCard 2.1

3.4.2 Examples

```
PBAP 00 1
BEGIN:VCARD
VERSION:2.1
FN:My name
N:My name
TEL;CELL:1234567890
END:VCARD
```

Following the syntax structure, the above command returns the “first” entry in the “Phone’s PhoneBook”. The example below is another example, which shows the Phone’s incoming call history of the last two calls.

```
PBAP 01 2
BEGIN:VCARD
VERSION:2.1
FN:701-921-5750
N:701-921-5750
TEL;X-0: 701-921-5750
X-IRMC-CALL-DATETIME;RECEIVED:20120529T123716
END:VCARD
BEGIN:VCARD
VERSION:2.1
FN:678-318-3100
N:678-318-3100
TEL;X-0:678-318-3100
X-IRMC-CALL-DATETIME;RECEIVED:20120524T085933
END:VCARD
```

The responses from iWRAP are OBEX Frames. This is because the PBAP profile is based on top of OBEX OPP profile to transfer phone book content. The OBEX frame header format is shown in the table in the following page.

Response	
{OBEX header} {vCARD}	
OBEX header	OBEX header. See the header descriptions below.
vCARD	vCard data

Length	**Name**	**Description**	**Value**
8 bits	Begin	Start of OBEX frame.	0xFC (last frame) or 0xFB (more frames to follow)
8 bits	Length	Length of full frame	0x00 – 0xFF (most significant byte)
8 bits	Length	Length of full frame	0x00 – 0xFF (least significant byte)
8 bits	Body	Body or end-of-body.	0x49 (last frame) 0x48 (more frames to follow)
8 bits	Length of data	Length of data field	0x00 – 0xFF (most significant byte)
8 bits	Length of data	Length of data field	0x00 – 0xFF (least significant byte)

Table 1: OBEX Header

Events
None

3.5 Connection termination

The PBAP connection should be terminated on iWRAP using the “**DISCONNECT**” or “**CLOSE {link_id}**” command.

PBAP connection termination:

```
CLOSE 0
NO CARRIER 0 ERROR 0
```

3.6 Example connection diagram

An example of PBAP configuration and a simple PBAP connection setup is illustrated below.

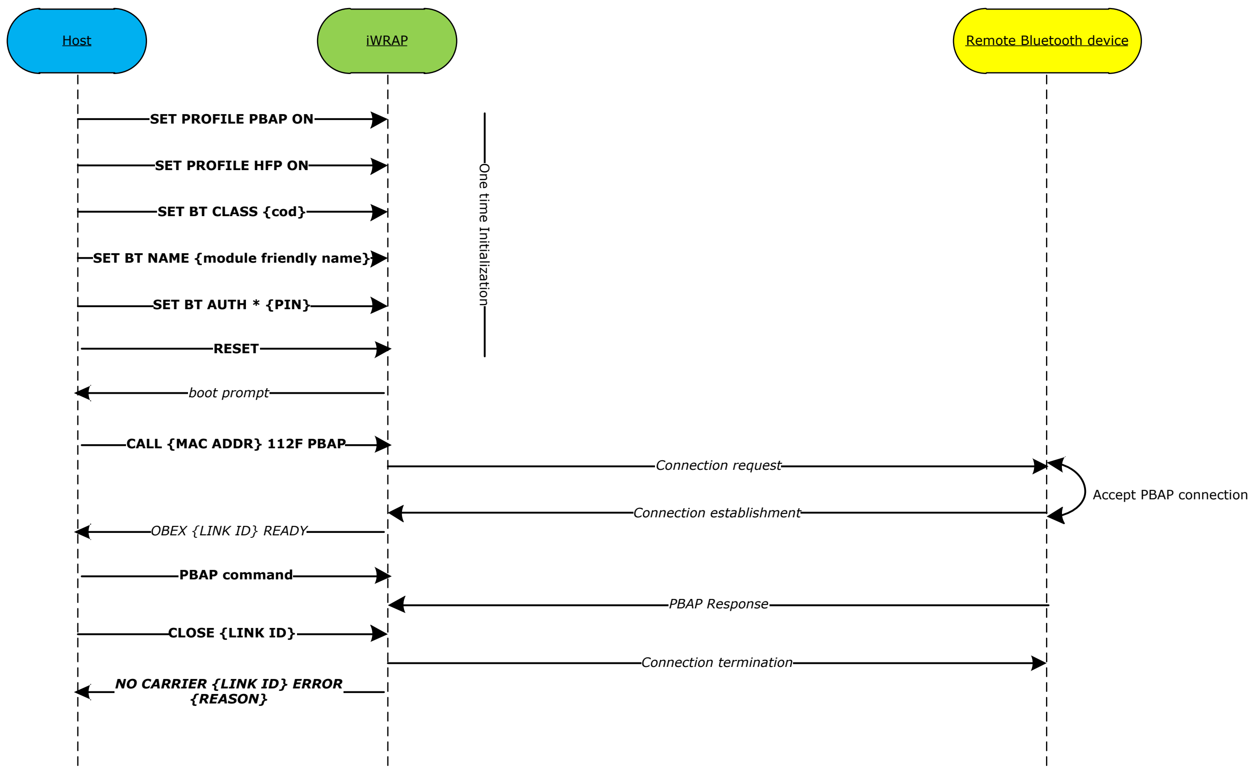


Figure 5: PBAP Connection Example

In the above example the **AUTH** event is only seen if a legacy pairing with pin code and interactive paring is made. If the legacy pairing does not take place and SSP pairing is used instead then **AUTH** event is not seen and there is no need to reply to it with the **AUTH** command. With SSP pairing, depending on the SSP mode, however SPP events may be displayed and then need to be responded with correct SSP pairing commands.

Please refer to iWRAP User Guide for more information about the iWRAP command and events.

4 Using MAP with iWRAP

This chapter instructs MAP usage and configuration with the iWRAP firmware.

4.1 Profile configuration

The MAP client will always connect the server, so the client does not need to present a service record. Therefore no initialization is required on the profile part. MAP imposes no requirements on the Class of Device field either, since many different types of devices may implement MAP. The Class of Device should therefore be based upon other profiles and functionalities of the device.

Generally, the Secure Simple Pairing Just Works mode should be used for maximum interoperability; that is, the second parameter of SET BT SSP should be set to zero. The first parameter should reflect the device's I/O capabilities. Please see the iWRAP User Guide for more information.

```
SET BT NAME {Module Friendly Name}

SET BT AUTH * {0-16 DIGIT PIN}

SET BT SSP 3 0

RESET
```

4.2 Service discovery

Bluetooth technology enables wireless service discovery, so you can find out the capabilities the remote device supports. Wireless service discovery uses the Bluetooth Service Discovery Profile (SDP).

With iWRAP the service discovery is performed with command: "**SDP {*bd_addr*} {*uuid*} [ALL]**".

bd_addr	Bluetooth device address of the remote device.
uuid	Universally unique identifier, which refers to the Bluetooth profile or service to be discovered. For Message Access Server service the ***uuid*** is 0x1132.
ALL	Optional flag to read all the SDP information from the remote device.

Below is an example how to perform a service discovery for MAP device.

```
SDP 00:07:80:aa:bb:cc 1132 ALL

SDP 00:07:80:aa:bb:cc < I 0 I 10014 > < I 1 < U 1132 > > < I PROTOCOLDESCRIPTORLIST
< < U L2CAP > < U RFCOMM I 0d > < U OBEX > > > < I 9 < < U 1134 I 100 > > >
< I SERVICENAME S " OBEX Message Access Server " > < I 315 I 00 > < I 316 I 02 >

SDP
```

OBEX Message Access Server	= Service name
0d	= RFCOMM channel for MAP server

4.3 Connection establishment

The MAP client (iWRAP) will always initiate the *Bluetooth* connection.

The connection is established with the following command:

"**CALL {*bd_addr*} 1132 MAP**"

bd_addr	Bluetooth device address of the remote device.

Below is an example how to set up a MAP from iWRAP to a MAP server device.

```
CALL 00:07:80:aa:bb:cc 1132 MAP
CALL 0
CONNECT 0 MAP 6
OBEX 0 READY
```

NOTE: On Apple iOS devices, the first MAP call will fail with an "**OBEX {link_id} FORBIDDEN**" error. This is because iOS requires that notifications to be specifically enabled for each paired *Bluetooth* device. In order to do this, **after calling the first time and receiving the FORBIDDEN error message**, go to the Bluetooth Settings app and tap on the small "Information" icon on the right side of the paired device entry in the list. Then, enable the "Show Notifications" slider. **MAP and MNS connections will not be possible until you do this.**

On Android devices when MAP (or PBAP) connection is established the Android phone will pop up dialog asking if the *Bluetooth* accessory is allowed to access the messages and/or phone book.

4.4 Navigating the folder structure

The MAP specification defines a folder structure every MAP server must present to the client. The MAP connection starts in the root folder, which always contains a telecom folder, which in turn always contains a msg folder. Inside the msg folder are inbox, sent, and deleted folders; and if the server supports sending messages to external networks (GSM, email etc.) via MAP, an outbox folder will be there as well.

The inbox contains at the very least all the incoming messages that have arrived during the MAP connection. In addition, it may contain other incoming messages either in the inbox folder, or messages sorted under different subfolders.

The same applies for the sent and deleted folders; they will contain at minimum the messages sent and deleted during the MAP connection.

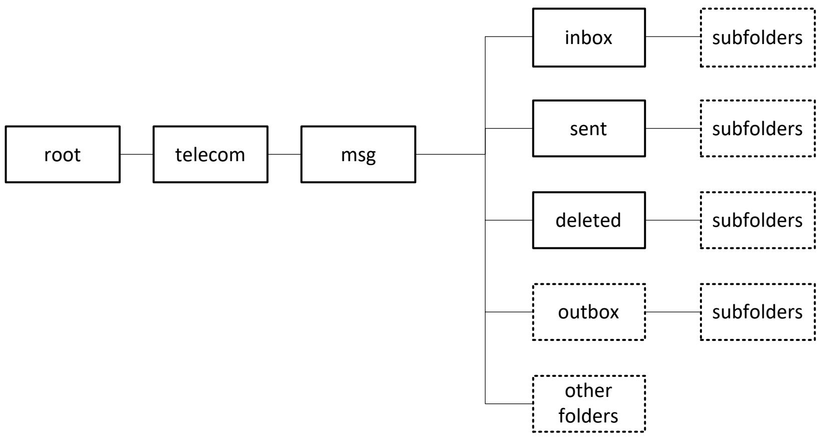


Figure 6: MAP Server Folder Structure

The commands **CD {folder}**, **CDUP**, and **DIR / LS** can be used to navigate the folder structure. Please see the iWRAP OBEX application note for detailed information about them. It should also be noted that in MAP exchanges, the directory listing will only contain folders, not files or messages.

NOTE: Folder browsing feature implementation may vary between devices, there are devices that do not support folder browsing at all, in that case it is necessary to issue **MAP telecom/msg/{folder}** command directly. In case of other devices, it may be necessary to navigate to the appropriate folder before sending **MAP** command:

```
CD telecom/msg/{folder}
MAP
```

Additionally there are devices that support folder browsing but **CD telecom/msg/{folder}** command may not be recognized properly, set of commands need to be used instead:

```
CD telecom
CD msg
CD {folder}
```

4.5 MAP command

MAP command is used to retrieve message listings from a MAP server.

4.5.1 Syntax

Synopsis
MAP {count} {status} [folder]

Description	
count	Number of entries to be retrieved. If empty, retrieves all messages.
status	Status messages to be retrieved. **0** No filtered messages **1** Unread messages **2** Read messages Optional parameter but it has to be provided with ***count***.
folder	The subfolder from which to retrieve the message listing. If empty, retrieve from the current folder.

Response	
{OBEX header} {MAP Messages-Listing}	
OBEX header	OBEX header. See the header description in Table 1.
MAP M-L	Message list data. Please see the appendix for details.

4.5.2 Response

The response is an OBEX-wrapped MAP Messages-Listing object. Please see the appendix for detailed format information.

4.5.3 Examples

MAP telecom/msg/inbox

<MAP-msg-listing version "1.0">

< msg handle = "20000100001" subject = "Hello" datetime = "20071213T130510Z" sender_name = "Jamie" sender_addressing = "+1-987-6543210" recipient_addressing = "+1-0123-456789" type = "SMS_GSM" size = "256" attachment_size = "0" priority = "no" read = "yes" sent = "yes" protected = "no"/>

< msg handle = "20000100002" subject= "Guten Tag" datetime = "20071214T092200Z" sender_name = "Dmitri" sender_addressing = "8765432109" recipient_addressing = "+49-9012-345678"type = "SMS_GSM" size = "512" attachment_size = "3000" priority "no" read "no" sent "yes" protected "no"/>

</MAP-msg-listing>

The listing is sorted according to date in descending order; that is newest message first.

Note:

There is no support in iOS 7.1 or older devices to read messages with the MAP command.

4.6 MSG command

MSG command is used to retrieve a message and possibly its attachment.

4.6.1 Syntax

Synopsis
MSG {get-attachment} {charset} {handle}

Description	
get-attachment	A flag telling if attachments should be retrieved or not. **0** Ignore attachments **1** Retreive attachments
charset	A setting telling which character set is used in the message. **0** Use remote devices native character set. **1** Transcode messages to UTF-8 If the message does not contain text data, the server will reject the UTF-8 transcode request.
handle	Message handle shown in the Messages-Listing object.

Response	
{OBEX header} {bMessage}	
OBEX header	OBEX header. See the header description in Table 1.
bMessage	bMessage data. See the appendix for details.

4.6.2 Response

The response is an OBEX-wrapped bMessage object. Please see the appendix for detailed format information.

4.6.3 Examples

```
MSG 0 0 20000100001
BEGIN:BMSG
VERSION:1.0
STATUS:UNREAD
TYPE:SMS_GSM
FOLDER:INBOX
BEGIN:VCARD
VERSION:2.1
N:Jamie
TEL:+1-987-6543210
END:VCARD
BEGIN:BENV
BEGIN:BBODY
ENCODING:G-7BIT
LENGTH:98
BEGIN:MSG
… the SMS PDU …
END:MSG
END:BBODY
END:BENV
END:BMSG
```

4.6.4 Connection termination

The MAP connection should be terminated on iWRAP using the "**DISCONNECT**" or "**CLOSE {link_id}**" command.

MAP connection termination:

```
CLOSE 0
NO CARRIER 0 ERROR 0
```

4.6.5 Example connection diagram

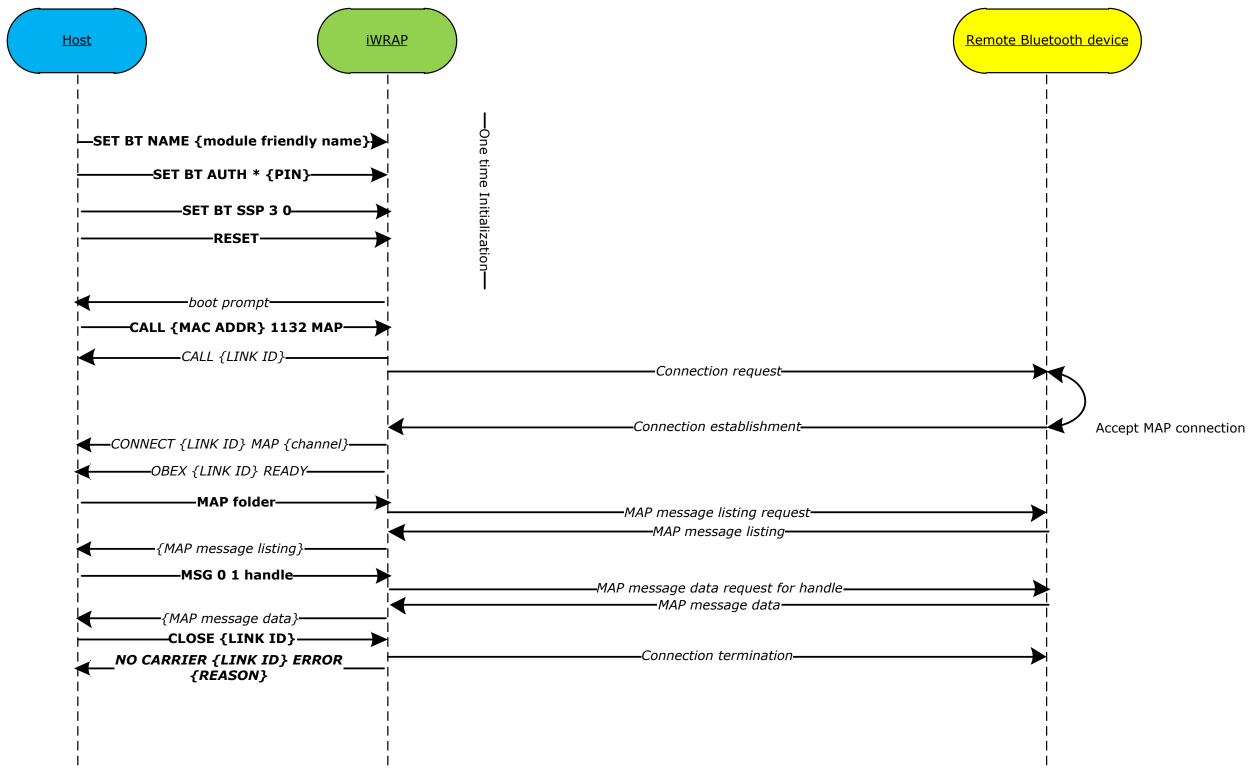


Figure 7: MAP Connection Example

In the above exchange, the Host uses the MAP command to list all messages and their handles in a folder. After receiving the handle of a message of interest, it uses the MSG command to retrieve the message, ignoring any attachments, transcoded into UTF-8.

5 Using Message Notification Service with iWRAP

Message notification service (MNS) is one of the Message Access Profile composing sub-features which allows to send notification messages from server (Message Server Equipment or MSE) to client (Message Client Equipment or MCE). For more information about Message Notification Service capabilities refer to [3].

5.1 Profile configuration

In order to be able to establish MNS connection MCE should present Message Notification Service record. Therefore MNS profile initialization is required on the iWRAP side.

```
SET PROFILE MNS ON
RESET
```

5.2 MAP connection establishment

As Message Notification Service is a MAP feature, it requires ongoing Message Access Service connection explained in the section 4.3.

5.3 Notification registration

After setting up MAP connection between the iWRAP and a MSE, the iWRAP needs to register notification function on the MSE (remote device).

```
NOTIFY ON
OBEX 0 OK
RING 1 d8:9e:3f:bb:07:65 5 MNS
OBEX 1 READY
```

After remote device receives the notification registration request, it will indicate separate MNS connection to the iWRAP.

Note: From this moment MNS connection is on the top of the running connections, therefore performing MAP commands is possible with use of @ parameter (see section 5.6).

5.4 Send Event Function messages example

After MCE (iWRAP) successfully registers itself at the MSE (remote device) for notification, the MSE shall notify the MCE about any change of a Messages-Listing of any folder on the MSE side if it not has been initiated by the MCE itself. Below there are two notification messages examples presented.

```
[HH][HH] … [HH][HH]<MAP-event-report version="1.0">

<event type="NewMessage" handle="0000000000006e50" folder="TELECOM/MSG/INBOX"
msg_type="SMS_GSM" /></MAP-event-report>
```

Message used by MSE to notify about the new SMS being received.

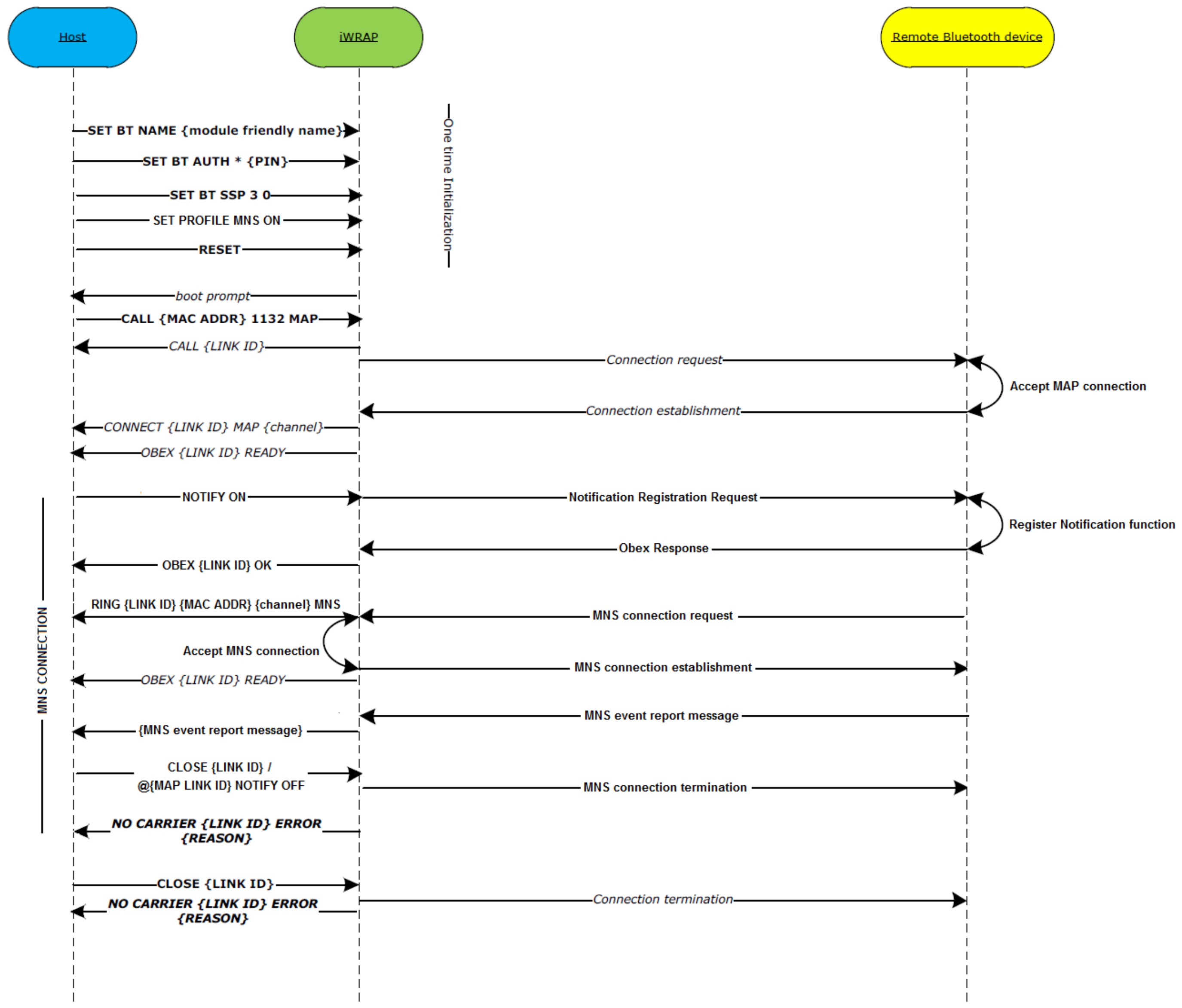
```
[HH][HH] … [HH][HH]<MAP-event-report version="1.0">

<event type="MessageDeleted" handle="0000000000006e20" folder="TELECOM/MSG/INBOX"
msg_type=" SMS_GSM" /></MAP-event-report>
```

Message used by MSE to notify about the SMS being deleted.

The [HH][HH] … [HH][HH] strings in the above examples corresponds to the OBEX headers received with every OBEX type message.

5.5 MNS connection example



5.6 Connection termination

The MNS connection should be terminated on iWRAP using either "**CLOSE {link_id}**" or **@{MAP link_id} NOTIFY OFF**"

MAP connection termination:

```
LIST
LIST 3
LIST 0 CONNECTED HFP 667 0 0 295 8d 8d d8:9e:3f:bb:07:65 2 INCOMING SNIFF SLAVE
ENCRYPTED 0
LIST 1 CONNECTED MAP 666 0 0 252 8d 1 d8:9e:3f:bb:07:65 2 OUTGOING SNIFF SLAVE ENCRYPTED
0
LIST 2 CONNECTED MNS 667 0 0 240 8d 8d d8:9e:3f:bb:07:65 5 INCOMING SNIFF SLAVE
ENCRYPTED 0
CLOSE 1 / @1 NOTIFY OFF
OBEX 0 OK
NO CARRIER 2 ERROR 0
```

6 References

[1] The Bluetooth SIG, Phone Book Access Profile overview, URL:
https://www.bluetooth.com/specifications/profiles-overview

[2] VCARD File Format, URL: http://en.wikipedia.org/wiki/VCard

[3] The Bluetooth SIG, Message Access Profile specification, URL:
https://www.bluetooth.com/specifications/profiles-overview

7 Appendix

7.1 MAP Messages-Listing object description

The Document Type Definition for the message listing is as follows.

Each attribute is declared in triplets consisting of attribute name, attribute data type, and default value. An attribute's default value #REQUIRED means that the attribute must always be set. #IMPLIED means that it is optional to include. #FIXED means that it must be set to the specified value. Any specified default value (for example "no") means that if the attribute is not included in the listing, the parser should treat it as if it had received the attribute with the default value.

```
<!DTD for the MAP Messages-Listing Object-->

<!DOCTYPE MAP-msg-listing [
<!ELEMENT MAP-msg-listing ( msg )* >
<!ATTLIST MAP-msg-listing
version CDATA #FIXED "1.0"
>
<!ELEMENT msg EMPTY>
<!ATTLIST msg

handle CDATA #REQUIRED

subject CDATA #REQUIRED

datetime CDATA #REQUIRED

sender_name CDATA #IMPLIED

sender_addressing CDATA #IMPLIED

replyto_addressing CDATA #IMPLIED

recipient_name CDATA #IMPLIED

recipient_addressing CDATA #REQUIRED

type CDATA #REQUIRED

size CDATA #REQUIRED

text (yes|no) "no"

reception_status CDATA #REQUIRED

attachment_size CDATA #REQUIRED

priority (yes|no) "no"

read (yes|no) "no"

sent (yes|no) "no"

protected (yes|no) "no"

>

]>
```

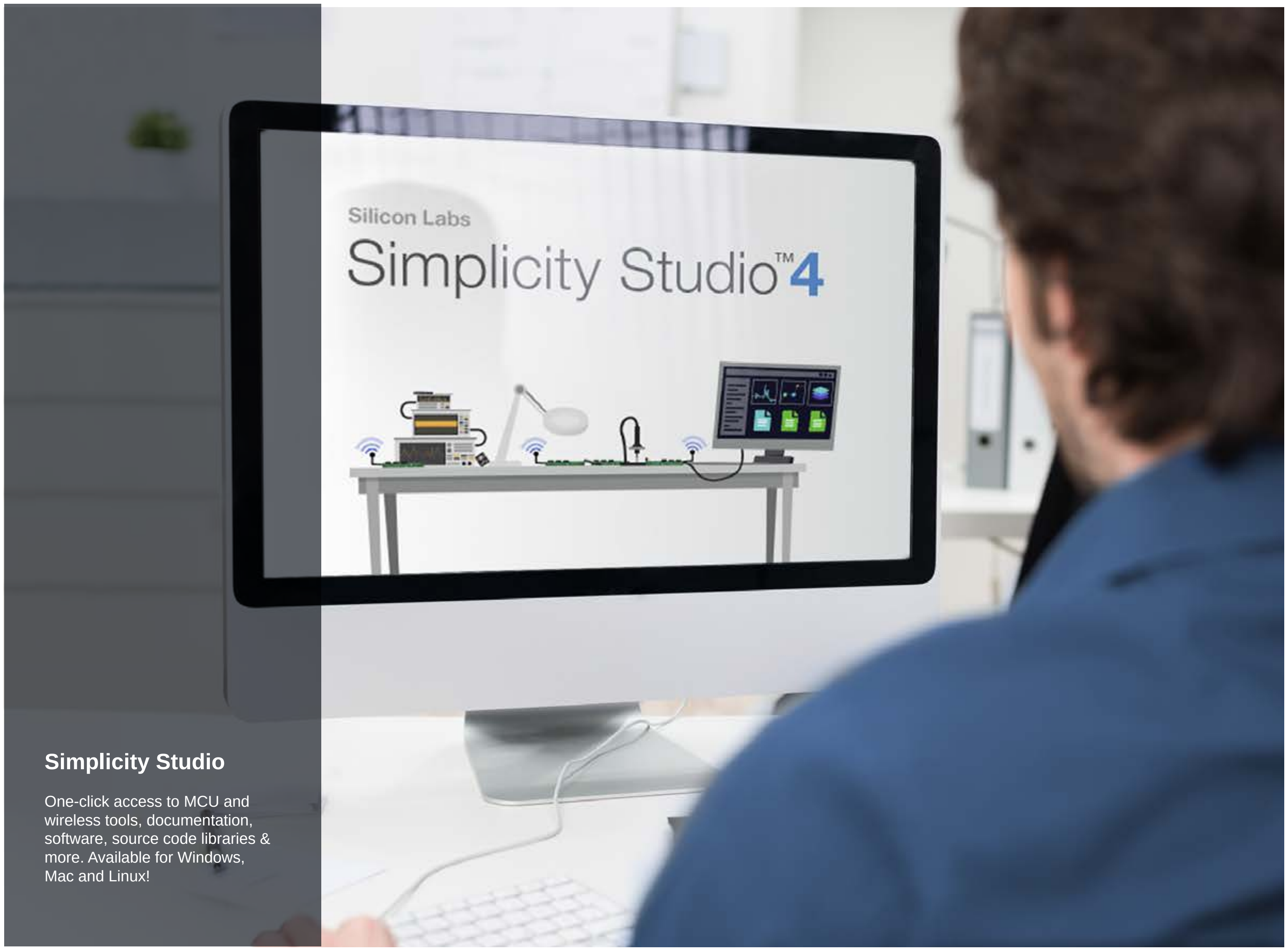
7.2 bMessage object description

The Backus-Naur Form of the bMessage is as follows.

It should be noted that the maximum depth of bEnvelope encapsulation in the recipients list is 3; if the original message has more than 3 levels, only the 3 most recent ones shall be delivered. Supported values for the type property are “EMAIL”, “SMS_GSM”, “SMS_CDMA” and “MMS”. In the vCards contained in the bMessage, only the properties VERSION, N, TEL, EMAIL can be used for vCard v2.1; VERSION, N, FN, TEL, EMAIL can be used for vCard v3.0. No other properties shall be used.

```
<bmessage-object>::= {
"BEGIN:BMSG" <CRLF>
<bmessage-property>
[<bmessage-originator>]*
<bmessage-envelope>
"END:BMSG" <CRLF>
}
<bmessage-property>::=<bmessage-version-property>
<bmessage-readstatus-property> <bmessage-type-property>
<bmessage-folder-property>
<bmessage-version-property>::="VERSION:"
<common-digit>*"."<common-digit>* <CRLF>
<bmessage-readstatus-property>::="STATUS:" 'readstatus' <CRLF>
<bmessage-type-property>::="TYPE:" 'type' <CRLF>
<bmessage-folder-property>::="FOLDER:" 'foldername' <CRLF>
<bmessage-originator>::= <vcard> <CRLF>
<bmessage-envelope> ::= {
"BEGIN:BENV" <CRLF>
[<bmessage-recipient>]*
<bmessage-envelope> | <bmessage-content>
"END:BENV" <CRLF>
}
<bmessage-recipient> ::= <vcard> <CRLF>
<bmessage-content>::= {
"BEGIN:BBODY"<CRLF>
[<bmessage-body-part-ID> <CRLF>]
<bmessage-body-property>
<bmessage-body-content>* <CRLF>
"END:BBODY"<CRLF>
}
<bmessage-body-part-ID>::="PARTID:" 'Part-ID'
<bmessage-body-property>::=<bmessage-body-encoding-property>
[<bmessage-body-charset-property> <CRLF>]
[<bmessage-body-language-property> <CRLF>]
<bmessage-body-content-length-property> <CRLF>
<bmessage-body-encoding-property>::="ENCODING:"'encoding' <CRLF>
<bmessage-body-charset-property>::="CHARSET:"'charset' <CRLF>
<bmessage-body-language-property>::="LANGUAGE:"'language' <CRLF>
<bmessage-body-content-length-property>::=
"LENGTH:" <common-digit>* <CRLF>
```

```
<bmessage-body-content>::={
"BEGIN:MSG"<CRLF>
'message'<CRLF>
"END:MSG"<CRLF>
}
```



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio *www.silabs.com/IoT*	**SW/HW** *www.silabs.com/simplicity*	**Quality** *www.silabs.com/quality*	**Support and Community** *community.silabs.com*

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com