HF-Protocol V1.2.1 (2017-12-18)

ULE Alliance Standard

Digital Enhanced Cordless Telecommunications (DECT);
Ultra Low Energy (ULE);

Home Area Network Functional (HAN-FUN) Protocol

Keywords

DECT, ULE, HAN, HAN-FUN

ULE ALLIANCE
Secretariat

Wabernstrasse 40
3007 Berne
Switzerland

T: +49 89 5166 2456 M: +49 160 9667 96966
Email: secretariat@ulealliance.org
http://www.ulealliance.org

Important notice

Individual copies of the present document can be downloaded from:
http://www.ulealliance.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ULE Alliance printers of the PDF version kept on a specific network drive within ULE Alliance Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status without further notice. Information on the current status of this and other ULE Alliance documents is available at
http://www.ulealliance.org

If you find errors in the present document, please send your comment to one of the following services:
secretariat@ulealliance.org

Copyright Notification / Terms and Conditions of Use *

* See the annex of the document.

Contents

2 Intellectual Property Rights

The Intellectual Property Rights (IPR) regulation is binding on all members and adopters participating in the ULE Alliance. Its purpose is to ensure the widest possible dissemination of the specifications adopted by the ULE Alliance while giving due weight and respect to the IPR of its members.

The IPR regulation can be found at http://www.ulealliance.org/downloads.aspx?c=w (Miscellaneous)

Foreword

This document has been produced by the ULE Alliance ULE WG.

The information in the present document is believed to be correct at the time of publication. However, Home Area Network Functional (HAN-FUN, or HF) may rapidly evolve, and consequently, it is possible that some of the information contained in the present document may become incomplete.

The present document is part of a multi-part deliverable covering the HF protocol as identified below:

HF-Overview [REF 1]: Overview

HF-Protocol [REF 2]: Protocol Specification

HF-Service [REF 3]: Core Services & Interfaces

HF-Interface [REF 4]: Interface Library

HF-Profile [REF 5]: Profiles

HF-ULE-Interworking [REF 6]: HF &ULE Interworking

1 Scope

The present document specifies the HAN-FUN (HF) protocol.

3 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents:

[1] ETSI EN 300 175-1: "Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 1: Overview"
http://www.etsi.org/deliver/etsi_en/300100_300199/30017501/02.02.01_60/en_30017501v020201p.pdf

4 Abbreviations

For the purposes of the present document, the following abbreviations apply:

CAT-iq	Cordless Advanced Technology - Internet and Quality
CCM	Counter with CBC-MAC mode – Mode of operation for cryptographic block ciphers with a block length of 128 bits, providing both authentication and confidentiality
DECT	Digital Enhanced Cordless Telecommunications
DFS	DECT Forum Standard
GAP	Generic Access Profile
HAN	Home Area Network
HAN-FUN (HF)	Home Area Network Functional
HF-IFL	HAN-FUN-Interface Library
HF-PRF	HAN-FUN-Profiles
ID	Identifier
LED	Light Emitting Diode
OTA	Over-the-Air
PDU	Protocol Data Unit
PP	Portable Part
SDU	Service Data Unit
TCP/IP	Transmission Control Protocol/Internet Protocol
UID	Unique Identifier
ULE	Ultra Low Energy
WG	Working Group
XML	Extensible Markup Language

5 Introduction

The HF Protocol aims to support applications from areas like home automation, security, smart energy and health within Home Area Networks (HANs).

5.1 HF-Protocol Versioning

The major number of a HF Core Release specifies a certain release of the present document. Interoperability between device profiles is ensured when based on that same release.

6 Technology

The HF Protocol can be used with different transport layers, like for example TCP/IP, it is however optimized for the DECT ULE transport layer. Such restrictions and implications of the optimization for DECT ULE are detailed on this section and affect the HF Protocol only when implemented over a DECT stack with ULE transport layer. When another transport layer, based on a protocol other than DECT is used, these restrictions may not apply and the only assumption made by the HF Protocol is the network topology and that devices are able to communicate with each other.

6.1 Network Topology

DECT is based on a star network topology, where a single Concentrator is the network's master device and supports up to thousands of devices connected to it. The same topology is assumed by the HF Protocol as shown in Figure 1.

Some DECT networks may have another type of device – repeaters – used to extend the Concentrator's range and act as a pass-through pipe for any network messages that come across it. Others employ more than one Concentrator to achieve the same range extension. Neither method is currently supported by the HF Protocol but they might be supported in some future release.

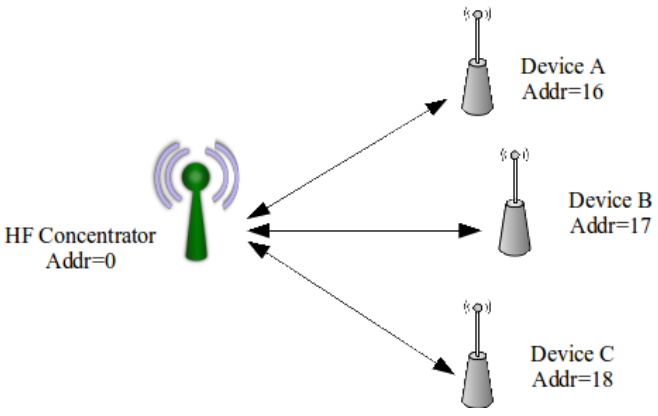

Figure 1 - HAN-FUN network topology.

6.1.1 Network Entities

6.1.1.1 HF Concentrator

HF Concentrator is the centre of a star topology HF network as depicted in Figure 1. This device may concentrate all the required processing necessary for the network to operate or delegate some or all that processing to a third party, being another device or devices, or even cloud services. Such processing involves running the HF protocol stack on top of some other protocol's transport layer and some or all HF network services described in HF-Service document [REF 3].

HF device address 0x0000 is reserved for this device and there may be only one per HF network.

6.1.1.2 HF Device

Any devices, other than the HF Concentrator that implement the HF Protocol are named HF Devices. From a DECT point of view each of these devices is a DECT Portable Part (PP).

Each HF device, after registering itself according to the normal procedures of the underlying protocol, must go through the HF specific registration procedures as well. If this latter registration is successful, the HF Concentrator will attribute a device address from the pool of 32765 addresses that range from 0x0001 to 0x7FFE. Device address 0x7FFF is a special reserved address.

6.1.1.2.1 Unit

A Unit is a conceptual entity inside a HF device that instantiates the functionality of a specific profile. Profiles, for example smoke detector, simple switch and more are described in the HF-Profile document [REF 5].

A HF device can host up to 254 units either instantiating the same profile or a variety of profiles. Each unit has a fixed pre-defined identifier (ID) that allows addressing messages to it; this is required since HF messages are always exchanged between units. Therefore, the unit ID combined with the unique HF device address produces a fully qualified HF network address, that allows units to communicate with each other independently of being in the same device or not.

Two special IDs exist, Unit with ID 0x00, on any HF device, is reserved for implementation of device management and other HF core services, as described in the HF-Service document [REF 3]; Unit ID 0xFF is a special reserved ID, required for the correct operation of some HF core services.

As an example consider Figure 2 that shows a simple HF network with the HF Concentrator having its management application on unit 0 and a single device implementing two units instantiating the *AC Outlet* profile. This device also has its management services on unit 0.

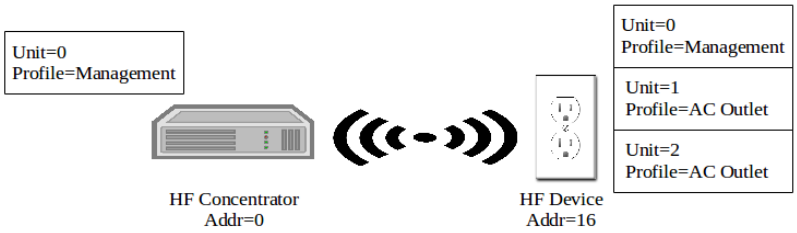

Figure 2 - Simple HF Network Emphasizing Profiles

6.1.1.2.2 Interface

An Interface is a conceptual entity inside a unit that defines a collection of commands and attributes, allowing for units to understand one another. Interfaces can be mandatory or optional to implement by a unit, and they have a role – client or server – associated with them. A single unit must implement either role, but not both. There is no limit for the total number of interfaces a unit might have, although there is a maximum of 126 optional ones.

For more details on application interfaces consult the HF-Interface document [REF 4] and the HF-Service document [REF 3] for core services interfaces.

7 HAN-FUN Message Format

This section describes the general structure of a HF message and implications of sending and receiving them. The information carried by a message is grouped per protocol layer being the HF relevant fields described in detail.

For some information regarding the general structure of a DECT ULE message, consult Annex A.

7.1 HAN-FUN message – Network Layer

Network	Transport	Application
6 bytes	2 bytes	Up to 492 bytes

Figure 3 - HF message structure: Network Layer.

The HF network layer is responsible for identifying the device or devices involved in a communication. A HF message carries the information required for it to be correctly delivered and for a receiver to know who the sender is. To store such information 6 bytes are required according to the bit distributions presented in Table 1.

Table 1 - Bit distribution of Network layer information.

Field Name	Extended Mode	SRC Address	SRC Unit ID	DST Address Type	DST Address	DST Unit ID
Size (bits)	1	15	8	1	15	8

Each of the fields in Table 1 is required and has the following meaning:

- **Extended Mode**

  Extended mode is a future feature; presently this bit should always be set to zero, indicating non-extended mode is to be used.

- **SRC Address and Unit ID**

  Together these two fields specify the full source unit address, uniquely identifying the sender of the message.

- **DST Address Type**

  This field affects the type of delivery of the message, indicating if it is meant for a single device (field set to zero) or a group of devices (field set to one). In the latter case, the Concentrator (device D'0:U'0) must support the optional group management service. Refer to the HF-Service document [REF 3] for details.

- **DST Address and Unit ID**

  Together these two fields specify the full destination unit address, uniquely identifying the receiver of the message. Both fields are required when addressing a single device (*DST Address Type* set to zero).

  When addressing a group of devices (*DST Address Type* set to one), *DST Address* field carries the group's address and *DST Unit ID* field must be set to 0xFF.

  When the full destination unit address is unknown, a special address having *DST Address* set to 0x7FFF and *DST Unit ID* to 0xFF, can be used. This special address will always be accepted by a Concentrator and forces it to check its binding tables. Binding tables shall always be supported, refer to Bind Service in HF-Service document [REF 3] for details.

These fields when in a HF OTA message are organized according to Table 2.

Table 2 - Byte ordering of HF Network Layer information in a message.

8	7	6	5	4	3	2	1	Octet
Extended Mode	SRC Address (MSB)							1
SRC Address (LSB)								2
SRC Unit ID								3
DST Address Type	DST Address (MSB)							4
DST Address (LSB)								5
DST Unit ID								6

7.2 HAN-FUN message – Transport Layer

Network	Transport	Application
6 bytes	2 bytes	Up to 492 bytes

Figure 4 - HF message structure: Transport Layer.

This field is reserved for future features and no bit distribution is fully defined yet. It should always have the value 0x0000 and can safely be ignored by applications implementing HF V1.0.0.

7.3 HAN-FUN message – Application Layer

Network	Transport	Application
6 bytes	2 bytes	Up to 492 bytes

Figure 5 - HF message structure: Application Layer.

For the correct handling and processing of application data, this layer requires a HF OTA message to carry HF specific protocol information besides the application data itself. To store such information 8 bytes are required according to the bit distributions presented in Table 3.

Table 3 - Bit distribution of Application layer information.

Field Name	App Ref	Message Type	Interface Type	Interface UID	Interface Member	Reserved	Data Length	Application Data
Size (bits)	8	8	1	15	8	7	9	0 – 485 bytes

Each of the fields in Table 3 is required and has the following meaning:

- Application Reference (App Ref)

  Consists on a value from 0x00 to 0xFF; used to match responses with requests in case several requests are issued before responses arrive.

- Message Type

  Each message is of a specific type, each type having certain implications. Check 7.3.2 for more details.

- Interface Type

  Interfaces (6.1.1.2.2) may have one of two roles, setting this field to 0 (zero) denotes a client interface while setting it to 1 (one) denotes a server interface. The interface type is of the destination (receiver) device when message type specifies a command or a request. In contrast, in response messages the interface type is of the source (sender) device.

- Interface UID

  Unique identifier of an interface (6.1.1.2.2).

- Interface Member

  Interface member specifies the particular entity from command, attribute or pack of the interface, specified by *Interface UID*, to which the message refers to. This field is intrinsically connected to the *Message Type* field as detailed in 7.3.1.

- Reserved

  Bit reserved for future features.

- Data Length

  Size of the *Application Data* (without any protocol specific information) sent in the message payload. Up to 485 bytes can be sent per OTA message. If a message has no payload, this field is set to 0 (zero).

- Application Data

  Application data should be ordered according to the Interface specification.

These fields when in a HF OTA message are organized according to Table 4.

Table 4 - Byte ordering of HF Application Layer information in a message.

8	7	6	5	4	3	2	1	Octet
Application Reference (App Ref)								1
Message Type								2
Interface Type	Interface UID (MSB)							3
Interface UID (LSB)								4
Interface Member								5
Reserved							Data Length (MSB)	6
Data Length (LSB)								7
Application Data (MSB)								8
…								⋮
Application Data (LSB)								9 + n[1]

7.3.1 Interface Member Values

An interface member value depends on the type of message being sent as shown in Table 5. HF OTA messages related to commands (message type with code between 0x01 and 0x03) will make the interface member value to be interpreted as an interface's command ID. Similarly, messages related to attributes (message type with code between 0x04 and 0x08) will make the interface member value to be interpreted as an interface's attribute ID. Messages that operate over an attribute pack (message type code with code between 0x09 and 0x0D) will make the interface member value to be interpreted as a special ID from Table 6 or from the interface itself.

Table 5 - Interface member values with their matching message type codes.

Message Type General Designator	Message Type Code	Interface Member	Description
Command	0x01	Interface Command ID	The command ID depends upon the interface to which it belongs. For core services interfaces, consult the HF-Service document [REF 3]. For functional interfaces, consult the HF-Interface document [REF 4].
	0x02		
	0x03		
Attribute	0x04	Interface Attribute ID	The attribute ID depends upon the interface to which it belongs. For core services interfaces, consult the HF-Service document [REF 3]. For functional interfaces, consult the HF-Interface document [REF 4].
	0x05		
	0x06		
	0x07		
	0x08		

[1] The value n represents an accumulation of an arbitrary number of bytes.

Message Type General Designator	Message Type Code	Interface Member	Description
Attribute Pack	0x09	Attribute Pack ID	*Attribute Pack* is an interface specific mechanism to group attributes together under an ID. This allows for an operation/action to be applied to several attributes with a single message. It can take one of the special values specified in Table 6 or any value specified by the intended interface. For core services interfaces, consult the HF-Service document [REF 3]. For functional interfaces, consult the HF-Interface document [REF 4].
	0x0A		
	0x0B		
	0x0C		
	0x0D		
	0x0E		
	0x0F		
	0x10		

Table 6 - List of Special Attribute Pack IDs and their meaning.

Attribute Pack		
ID	**Name**	**Description**
0x00	Mandatory	Request all mandatory attributes.
0xFE	All	Request all mandatory and optional attributes.
0xFF	Dynamic	Dynamically request attributes. When this attribute pack ID is used a list of attributes must be provided.

7.3.2 Message Types

The HF protocol defines different general messages (Table 7), to easily accommodate diverse behaviours that may be required by applications. These messages are categorized into Commands, Requests and Responses.

Commands are interface specific, which means, interfaces, either functional or service related, define commands and therefore payload data and structure from both the command and the optional/mandatory response are specified in their specific documents; for service related commands consult the HF-Service document [REF 3]. For functional interfaces, consult the HF-Interface document [REF 4].

Requests are general messages that usually operate over an interface's attributes (either individually or as a pack) and may or may not involve a response. The payload data and structure of requests and their optional/mandatory responses is defined in the following sections.

All protocol message types shown in Table 7 shall be supported by implementations of any interface [REF 4] and/or service [REF 3] defined in HF. The only exceptions are as follows:

- Interfaces and/or services that do not specify attributes are not required to support requests (message type codes from 0x04 to 0x10). For example the SUOTA service [REF 3].

- Interfaces and/or services that do not specify commands are not required to support message type codes from 0x01 to 0x03. For example the RSSI service [REF 3].

- Proprietary interfaces may respond with *0x03 - Fail: Not supported* to any message type[2].

Table 7 - Types of messages defined in the HF protocol.

Message Type Code	Message Type Designation	Description
0x01	Command	Triggers or accomplishes some action on the destination device. An optional response will never be sent. Example: “Turn a light on”.
0x02	Command with Response Required	Triggers or accomplishes some action on the destination device. A response will always be sent. Example: “Trigger alarm siren”.
0x03	Command Response	Response indicating the state of the command reception/processing and some payload if required. Example: “Light is on”.
0x04	Get Attribute Request	Retrieve the value(s) of some specified attribute from an interface. This command will have a response. Example: “Get number of groups”.
0x05	Response to a Get Attribute Request	Mandatory reply sent when a *Get Attribute Request* is received. Example: “There are 5 defined groups”.
0x06	Set Attribute Request	Set the value of some specified attribute on an interface. A response will never be sent. Example: “Set temperature to 23ºC”
0x07	Set Attribute Request with Response Required	Set the value of some specified attribute on an interface. A response must be sent. Example: “Set keep alive period to 500ms”
0x08	Response to a Set Attribute Request	Response indicating the state of the command reception/processing and some payload if required. Example: “Temperature was set to 23ºC”.

[2] This may be the case of proprietary interfaces that want to protect themselves. For example if such an interface receives a Get Attribute Pack Request (0x09) with pack ID 0xFE (all) it can reply with *0x03 - Fail: Not supported* instead of returning all its attributes like a standard interface does.

Message Type Code	Message Type Designation	Description
0x09	Get Attribute Pack Request	Retrieve the value(s) of more than one specified attribute from an interface. This command will have a response. Example: “Get battery voltage and type”.
0x0A	Response to a Get Attribute Pack Request	Mandatory reply sent when a *Get Attribute Pack Request* is received. Example: “Battery voltage is 1000mV and its type is AA”.
0x0B	Set Attribute Pack Request	Try to set the value of more than one specified attribute on an interface. A response will never be sent. Example: “Set Average Power Interval to 300 seconds and Report interval to 1800 seconds”.
0x0C	Set Attribute Pack Request with Response Required	Try to set the value of more than one specified attribute on an interface. A response must be sent. Example: “Set Average Power Interval to 300 seconds and Report interval to 1800 seconds”.
0x0D	Response to a Set Attribute Pack Request	Response indicating the state of the command reception/processing and some payload if required. Example: “Average Power Interval set to 300 seconds and Report interval set to 1800 seconds”.
0x0E	Atomic Set Attribute Pack Request	Atomically set the value of more than one specified attribute on an interface. A response will never be sent. Example: “Set Average Power Interval to 300 seconds and Report interval to 1800 seconds”.
0x0F	Atomic Set Attribute Pack Request with Response Required	Atomically set the value of more than one specified attribute on an interface. A response must be sent. Example: “Average Power Interval to 300 seconds and Report interval to 1800 seconds”.
0x10	Response to an Atomic Set Attribute Pack Request	Response indicating the state of the command reception/processing and some payload if required. Example: “Average Power Interval set to 300 seconds and Report interval set to 1800 seconds”.

7.3.2.1 Command

This message type is used to send an interface specific command that will not trigger the reception of an optional response. Mandatory responses will be received normally. The message's payload is specified by the interface it relates to.

7.3.2.2 Command with Response Required

This message type is equal to the *Command* (7.3.2.1) but a response (7.3.2.3) will always be sent.

7.3.2.3 Command Response

This message type is used to identify a response to a previously sent interface specific command. The response payload and format is specified by the interface it relates to, it can be either particular to that interface or the *General Response* (8).

7.3.2.4 Get Attribute Request

This message type is used to retrieve the value of an interface attribute. The attribute ID is specified in the *Interface Member* message field (see 7.3.1).

This message type has no payload and a response (7.3.2.5) is mandatory.

7.3.2.5 Response to a Get Attribute Request

This message type is used to identify the mandatory response to a *Get Attribute* Request. The message fields, *App Ref*, *Interface Type*, *Interface UID* and *Interface Member* (see Table 3) should be equal to the ones on the request (7.3.2.4). If *Response Code* field indicates a failure then *Attribute Value* field should be discarded if received. When the failure is due to a get on a non-implemented optional attribute, *Attribute Value* field may be omitted making the response only 1 byte in size.

This message type must provide the information described in Table 8, organized according to Table 9.

Table 8 - Data in the payload of a response to a *Get Attribute* Request.

Field Name	Field Description	Type	Value	M/O
Response Code	Value that indicates the state of the command reception/processing.	U8	0x00 - Ok0x03 - Fail: Not supported 0xFF - Fail: Unknown reason	M
Attribute Value	Value currently stored in the specified attribute.	-	-	M

Table 9 - Data ordering of the payload of a response to a *Get Attribute* Request.

8	7	6	5	4	3	2	1	Octet
Response Code								**1**
Attribute Value (MSB)								**2**
…								⋮
Attribute Value (LSB)								**3 + n**

7.3.2.6 Set Attribute Request

This message type is used to set the value of an interface attribute. The attribute ID is specified in the *Interface Member* message field (see 7.3.1).

This message type must provide the information described in Table 10, organized according to Table 11.

A response to this message type will never be sent. Note that this causes a set attribute on a non-implemented optional attribute or on a read only attribute to silently fail. If this behavior is not desired use the request described in 7.3.2.7 instead.

Table 10 - Data in the payload of a *Set Attribute* Request.

Field Name	Field Description	Type	Value	M/O
Attribute Value	Value to set in the specified attribute.	-	-	M

Table 11 - Data ordering of the payload of a *Set Attribute* Request.

8	7	6	5	4	3	2	1	Octet
Attribute Value (MSB)								**1**
…								⋮
Attribute Value (LSB)								**2 + n**

7.3.2.7 Set Attribute Request with Response Required

This message type is equal to the *Set Attribute* Request (7.3.2.6) but a response (7.3.2.8) is mandatory.

7.3.2.8 Response to a Set Attribute Request

This message type is used to identify the response to a *Set Attribute* Request. The message fields, *App Ref*, *Interface Type*, *Interface UID* and *Interface Member* (see Table 3) should be equal to the ones on the request (7.3.2.7).

This message type must provide a single byte with the information described in Table 12.

Table 12 - Data in the payload of a response to a *Set Attribute* Request.

Field Name	Field Description	Type	Value	M/O
Response Code	Value that indicates the state of the command reception/processing.	U8	0x00 - Ok 0x02 - Fail: Invalid Argument 0x03 - Fail: Not supported 0x04 - Fail: Read Only attribute 0xFF - Fail: Unknown reason	M

7.3.2.9 Get Attribute Pack Request

This message type is used to retrieve the value of attributes, specified by a pack ID, of an interface. The pack ID is specified in the *Interface Member* message field (see 7.3.1). It will always have a response (7.3.2.10).

This message type has no payload for any value of pack ID except when the special value *0xFF - Dynamic* is used. In this particular case, the message must provide the information described in Table 13, organized according to Table 14.

Table 13 - Data in the payload of a *Get Attribute Pack* Request, for the special *Attribute Pack ID* 0xFF - Dynamic.

Field Name	Field Description	Type	Value	M/O
Number of Attributes	Number of attributes, present in the command, for which its value is requested.	U8	0x00 - 0xFF	M
Attribute ID	Identifier of an attribute whose value is requested.	U8	0x00 - 0xFF	M

Table 14 - Data ordering of the payload of a *Get Attribute Pack* Request, for the special *Attribute Pack ID* 0xFF - Dynamic.

8	7	6	5	4	3	2	1	Octet
Number of Attributes								**1**
Attribute ID i								**2**
…								⋮
Attribute ID i+n								**3 + n**

7.3.2.10 Response to a Get Attribute Pack Request

This message type is used to identify the mandatory response to a *Get Attribute Pack* Request. The message fields, *App Ref*, *Interface Type*, *Interface UID* and *Interface Member* (see Table 3) should be equal to the ones on the request. If *Response Code* field indicates a failure then not all requested attributes exist or could be read. Attributes that fail, for example non-implemented optional attributes, will not appear in this message's payload. In the extreme case where a pack ID only references non-implemented optional attributes, the total message size is only 2 (two) bytes and *Number of Attributes* shall be set to 0x00 (zero).

This message type must provide, independently of the request's pack ID, the information described in Table 15, organized according to Table 16.

Table 15 - Data in the payload of a response to a *Get Attribute Pack* Request.

Field Name	Field Description	Type	Value	M/O
Response Code	Value that indicates the state of the command reception/processing.	U8	0x00 - Ok 0x03 - Fail: Not supported 0xFF - Fail: Unknown reason	M
Number of Attributes	Number of attributes, present in the command, for which its value was requested.	U8	0x00 - 0xFF	M
Attribute ID	Identifier of an attribute whose value was requested.	U8	0x00 - 0xFF	M
Attribute Value	Value currently stored in the specified attribute.	-	-	M

Table 16 - Data ordering of the payload of a response to a *Get Attribute Pack* Request.

8	7	6	5	4	3	2	1	Octet
Response Code								**1**
Number of Attributes								**2**
Attribute ID i								**3**
Attribute Value (MSB)								**4**
…								⋮
Attribute Value (LSB)								**5 + n**
…								⋮
Attribute ID i+n								**6 + n**
Attribute Value (MSB)								**7 + n**
…								⋮
Attribute Value (LSB)								**8 + n**

7.3.2.11 Set Attribute Pack Request

This message type is used to set the value of several attributes of an interface. The pack ID is specified in the *Interface Member* message field (see 7.3.1) and should be set to 0xFF. This request allows for some attributes to get a new value while some others may fail.

This message type must provide the information described in Table 17, organized according to Table 18.

A response to this message type will never be sent. Note that this causes a change on a non-implemented optional attribute or on a read only attribute to silently fail. If this behavior is not desired use the request described in 0 instead.

Table 17 - Data in the payload of a *Set Attribute Pack* Request.

Field Name	Field Description	Type	Value	M/O
Number of Attributes	Number of attributes, present in the command, for which its value should be set.	U8	0x00 - 0xFF	M
Attribute ID	Identifier of an attribute whose value is to be set.	U8	0x00 - 0xFF	M
Attribute Value	Value to set in the specified attribute.	-	-	M

Table 18 - Data ordering of the payload of a *Set Attribute Pack* Request.

8	7	6	5	4	3	2	1	Octet
Number of Attributes								**1**
Attribute ID i								**2**
Attribute Value (MSB)								**3**
…								⋮
Attribute Value (LSB)								**4 + n**
…								⋮
Attribute ID i+n								**5 + n**
Attribute Value (MSB)								**6 + n**
…								⋮
Attribute Value (LSB)								**7 + n**

7.3.2.12 Set Attribute Pack Request with Response Required

This message type is equal to the *Set Attribute Pack* Request (7.3.2.11) but a response (7.3.2.13) is mandatory.

7.3.2.13 Response to a Set Attribute Pack Request

This message type is used to identify the response to a *Set Attribute Pack* Request. The message fields, *App Ref*, *Interface Type*, *Interface UID* and *Interface Member* (see Table 3) should be equal to the ones on the request (0).

This message type must provide the information described in Table 19, organized according to Table 20.

The *Number of Attributes* on the response will normally be the same as on the request that originated it. The only exception is if an issue occurs when parsing the request, which can happen if the request contains an invalid or non-implemented optional *Attribute ID*. In such condition the *Attribute ID* that caused the parsing error will be the last attribute to be present on this response and its associated *Response Code* will indicate the failure reason. All attributes that were parsed before the failure will appear and can be handled normally. Any attributes that would be parsed after the failure are omitted.

Table 19 - Data in the payload of a response to a *Set Attribute Pack* Request.

Field Name	Field Description	Type	Value	M/O
Number of Attributes	Number of attributes, present in the command, for which a set value was requested.	U8	0x00 - 0xFF	M
Attribute ID	Identifier of an attribute whose value was to be set.	U8	0x00 - 0xFF	M
Response Code	Value that indicates the state of an attribute set value operation.	U8	0x00 - Ok 0x02 - Fail: Invalid Argument 0x03 - Fail: Not supported 0x04 - Fail: Read Only attribute 0xFF - Fail: Unknown reason	M

Table 20 - Data ordering of the payload of a response to a *Set Attribute Pack* Request.

8	7	6	5	4	3	2	1	Octet
Number of Attributes								**1**
Attribute ID i								**2**
Response Code								**3**
…								⋮
Attribute ID i+n								**4 + n**
Response Code								**5 + n**

7.3.2.14 Atomic Set Attribute Pack Request

This message type is used to atomically set the value of several attributes of an interface. The pack ID is specified in the *Interface Member* message field (see 7.3.1) and should be set to 0xFF. This request will only succeed if all attributes are successfully set, if at least one attribute fails, for example trying to set a non-implemented optional attribute, then the entire request fails and all specified attributes retain the value they had before this request was received.

This message type must provide the information described in Table 21, organized according to Table 22.

A response to this message type will never be sent. Note that this causes a set attribute on a non-implemented optional attribute or on a read only attribute to silently fail the entire request. If this behavior is not desired use the request described in 0 instead.

Table 21 - Data in the payload of an *Atomic Set Attribute Pack* Request.

Field Name	Field Description	Type	Value	M/O
Number of Attributes	Number of attributes, present in the command, for which its value should be set.	U8	0x00 - 0xFF	M
Attribute ID	Identifier of an attribute whose value is to be set.	U8	0x00 - 0xFF	M
Attribute Value	Value to set in the specified attribute.	-	-	M

Table 22 - Data ordering of the payload of an *Atomic Set Attribute Pack* Request.

8	7	6	5	4	3	2	1	Octet
Number of Attributes								**1**
Attribute ID i								**2**
Attribute Value (MSB)								**3**
…								⋮
Attribute Value (LSB)								**4 + n**
…								⋮
Attribute ID i+n								**5 + n**
Attribute Value (MSB)								**6 + n**
…								⋮
Attribute Value (LSB)								**7 + n**

7.3.2.15 Atomic Set Attribute Pack Request with Response Required

This message type is equal to the *Atomic Set Attribute Pack* Request (0) but a response (7.3.2.16) is mandatory.

7.3.2.16 Response to an Atomic Set Attribute Pack Request

This message type is used to identify the response to an *Atomic Set Attribute Pack* Request. The message fields, *App Ref*, *Interface Type*, *Interface UID* and *Interface Member* (see Table 3) should be equal to the ones on the request (0).

This message type must provide a single byte – Response Code – with one of the values described in Table 23.

Table 23 - Data in the payload of a response to an *Atomic Set Attribute Pack* Request.

Field Name	Field Description	Type	Value	M/O
Response Code	Value that indicates the state of the command reception/processing.	U8	0x00 - Ok 0x02 - Fail: Invalid Argument 0x03 - Fail: Not supported 0x04 - Fail: Read Only attribute 0xFF - Fail: Unknown reason	M

8 HAN-FUN General Response Format

Every HF message sent may have a response to ensure its correct reception or interpretation. Some message types (7.3.2) and some interfaces (6.1.1.2.2) define a response as mandatory, others define it as optional. In either case, when the response message format is not specified nor is a response code then the response message should have a single byte with one of the response codes described in Table 24.

Detailed information about a response code's meaning can be found in 8.1.

Table 24 - Data in the payload of a Default Response to any command or request.

Field Name	Field Description	Type	Value	M/O
Response Code	Value that indicates the state of the command reception/processing.	U8	0x00 - Ok 0x03 - Fail: Not supported 0xFF - Fail: Unknown reason	M

8.1 Global Response Codes

Some message types (7.3.2) and some interfaces (6.1.1.2.2) define their own specific response codes with the goal of giving more detailed information in case of specific errors.

Table 25 summarizes all response codes defined in the HAN-FUN protocol, including the three possible default response codes already in Table 24.

Table 25 – Summary of all response codes in HAN-FUN.

Response Code	Textual Meaning	Description
0x00	Ok	The request/command was correctly received and/or processed.
0x01	Fail: Not authorized	The requesting device needs to authenticate itself or it is simply not authorized to perform that request.
0x02	Fail: Invalid argument	One or more request/command arguments are invalid.
0x03	Fail: Not supported	Some requested feature, command or attribute is not implemented on the destination device. The operation will permanently fail.
0x04	Fail: Read Only attribute	The attribute you are trying to set is a read only attribute. The operation will permanently fail.
0x20	Fail: Read session not established	The operation requires a read session to be correctly established with the destination device.
0x21	Fail: Entries table was modified	The table, over which you were operating, changed. You should re-start the read session to avoid inconsistencies.
0xFE	Fail: Not enough resources	The available resources on the destination device are not sufficient to handle the request/command. Try again.
0xFF	Fail: Unknown reason	An unspecified error has occurred, the operation failed.

Annex A DECT ULE Messages – General Structure

Although an Over-the-Air (OTA) message can be transported by several DECT mechanisms or other transport protocols like UDP/IP or TCP/IP, it will be sent primarily using DECT ULE packet mode. The general structure of such a message, here described, justifies HF restrictions, for example in terms of application payload.

An OTA message is built from a Service Data Unit (SDU) that can contain up to 14 Protocol Data Units (PDUs). A PDU (Figure 6) has a fixed size of 38 bytes from which 2 bytes (Frame Header) are overhead due to DECT ULE protocol data using the FU10a format. Each SDU (Figure 7) employs CCM security and introduces an overhead of 4 bytes for the Message Integrity Code. This code will use 4 bytes from the last PDU and due to it, when a SDU with more than 1 PDU is sent the receiving device must receive the entire SDU before trying to decrypt it.

- "MAC size": for ULE data packets using full-slot, Ipq (protected single subfield)

PDU	
Frame Header	**Data in FU10a format**
2 bytes	36 bytes

Figure 6 - Protocol Data Unit (PDU) data diagram.

SDU							
PDU 1		**PDU 2 (Optional)**		**…**	**PDU 14 (Optional)**		
Frame Header	**Data**	**Frame Header**	**Data**	**...**	**Frame Header**	**Data**	**Message Integrity Code**
2 bytes	36 bytes	2 bytes	36 bytes	…	2 bytes	32 bytes	4 bytes

Figure 7 - Service Data Unit (SDU) data diagram.

The OTA HF message has the same structure as a SDU but the first PDU is required to have HF protocol specific information from HF Network (6 bytes), Transport (2 bytes) and Application (8 bytes) layers. The remaining bytes are for application data, except when the PDU is the only or the last in the message in which case 4 bytes are required for the Message Integrity Code.

The simplest OTA HF message is shown in Figure *8*; it contains a single PDU and uses all 24 bytes available for application data. If however the application data requires less than 24 bytes, for example only 8 bytes as depicted in Figure 9 then padding is required to fill the message to the total of 38 bytes.

Figure 10 shows an example of an OTA HF message using two PDUs while Figure 11 depicts a message with the maximum message length using all fourteen PDUs. From these two figures it is clear that each new PDU, other than the first, allows the application to send an extra 36 bytes of data. The only exception being the last PDU in the message that allows for an extra of 32 bytes.

1st PDU				
Frame Header	**Network**	**Transport**	**Application**	**Message Integrity Code**
2 bytes	6 bytes	2 bytes	24 bytes	4 bytes

Figure 8 - HF message with a single PDU using all bytes available for application data.

1st PDU					
Frame Header	**Network**	**Transport**	**Application**	**Message Integrity Code**	**Padding**
2 bytes	6 bytes	2 bytes	8 bytes	4 bytes	16 bytes

Figure 9 - HF message with a single PDU that requires padding.

1st PDU				2nd PDU		
Frame Header	**Network**	**Transport**	**Application**	**Frame Header**	**Application Data**	**Message Integrity Code**
2 bytes	6 bytes	2 bytes	28 bytes	2 bytes	32 bytes	4 bytes

Figure 10 - HF message with two PDUs.

1st PDU				2nd PDU		…	13th PDU		14th PDU		
Frame Header	**Network**	**Transport**	**App.**	**Frame Header**	**Application Data**	**…**	**Frame Header**	**Application Data**	**Frame Header**	**Application Data**	**Message Integrity Code**
2 bytes	6 bytes	2 bytes	28 bytes	2 bytes	36 bytes	…	2 bytes	36 bytes	2 bytes	32 bytes	4 bytes

Figure 11 - HF message with fourteenth PDUs, the maximum supported message length.

Annex B Terms and Conditions of Use

Version: 23 January 2014

The copyrights in this Document and the specifications contained herein are owned by the ULE Alliance and its Members (hereinafter, the 'ULE Alliance'). Use of this Document and the specifications and any related intellectual property (collectively, the "Specification"), is governed by this Notice and the IPR Regulation of the ULE Alliance, where appropriate.

Use of the Specification by anyone who is not a member of the ULE Alliance or an Adopter is prohibited. However, such parties shall be permitted to view this Document. Requests for permission to reprint this Document, in whole or in part, or requests for a license to reproduce and/or distribute this Document, in any form, must be submitted via email to secretariat@ulealliance.org or in writing to: ULE Alliance Secretariat, Wabernstr. 40, 3007 Bern, Switzerland.

Elements of this Document and Specification may be subject to third party intellectual property rights, including without limitation, patent, copyright or trademark rights (such a third party may or may not be a Member of ULE Alliance). Nothing in this Document or Specification may be construed as a license to use such intellectual property, and such license must be separately sought with the right holder. The ULE Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights

THE DOCUMENT AND SPECIFICATION ARE PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFI- CATION OR SAMPLE.

Each Member and Adopter hereby acknowledges that products equipped with the Specification (hereinafter, 'Products') may be subject to various regulatory controls under the laws and regulations of various governments worldwide. Each Member and Adopter is solely responsible for the compliance by Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their Products related to such regulations within the applicable jurisdictions. Each Member and Adopter acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses. NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS.

IN NO EVENT WILL THE ULE ALLIANCE BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OF DATA, INTERRUPTION OF BUSINESS, OR FOR ANY OTHER DIRECT, INDIRECT, SPECIAL OR EXEMPLARY, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES OF ANY KIND, IN CONTRACT OR IN TORT, IN CONNECTION WITH THIS DOCUMENT OR THE INFORMATION CONTAINED HEREIN, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The ULE Alliance reserves the right to adopt any changes or alterations to the Document or Specification as it deems necessary or appropriate without notice.