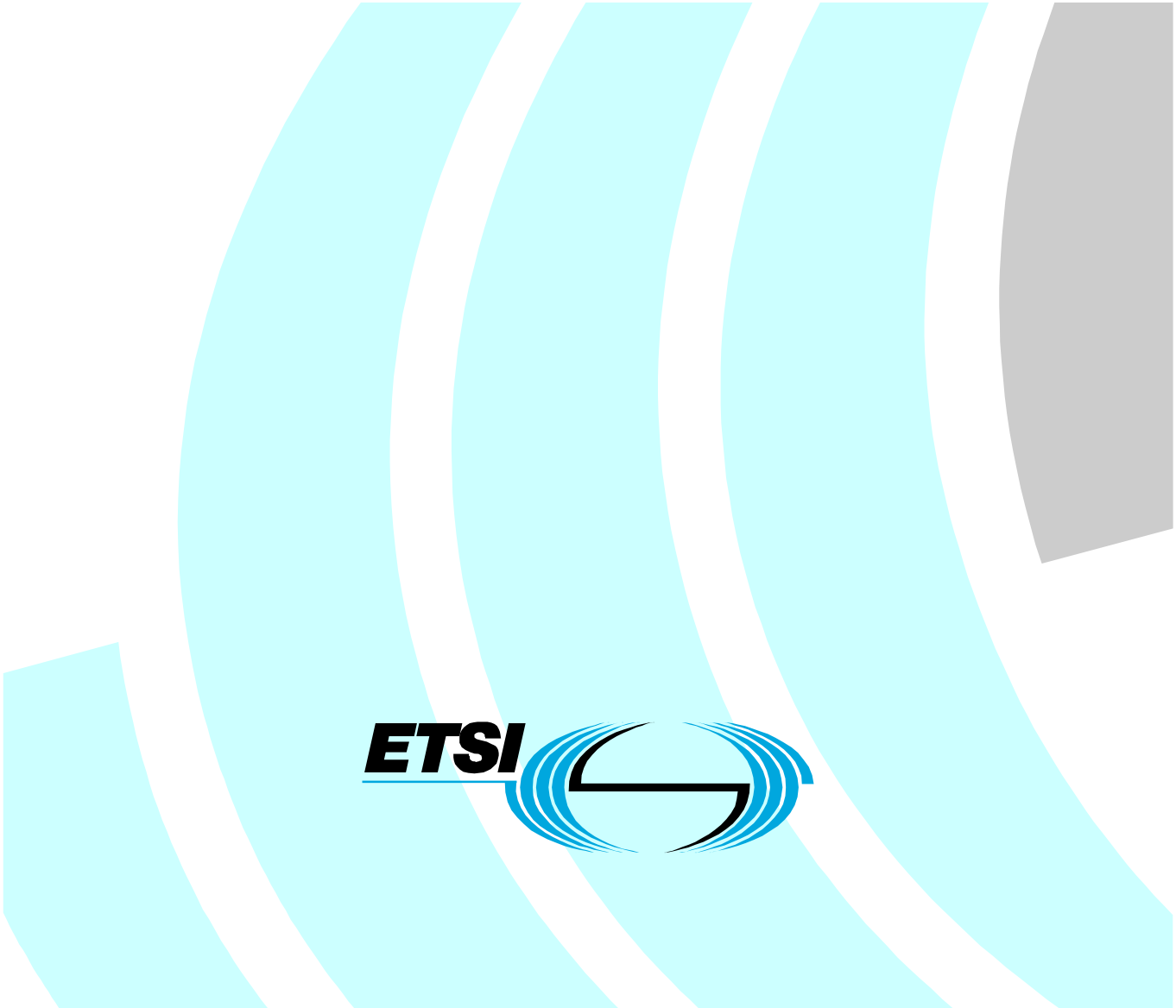
ETSI TS 102 342 V1.1.1 (2004-07)

Technical Specification

Digital Enhanced Cordless Telecommunications (DECT); Cordless multimedia communication system; Open Data Access Profile (ODAP)



Reference
DTS/DECT-000234

Keywords
data, DECT, interworking, GAP, multimedia

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.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 ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, send your comment to:
editor@etsi.org

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Project Digital Enhanced Cordless Telecommunications (DECT).

The present document is based on DECT Common Interface (CI) specification EN 300 175, parts 1 [1] to 7 [7], to enable DECT terminals to interwork in the public and private environment.

In addition, for the purpose of interoperability and wherever it is found appropriate, the present document takes into consideration the requirements of:

- the DECT Generic Access Profile (GAP), EN 300 444 [10] to enable the same DECT portable part (PT) to interwork with a DECT fixed part (FP) complying to the GAP requirements, irrespective of whether this FP provides residential, business or public access services.

General attachment requirements are based on EN 301 406 [11].

Further details on the DECT system may be found in TR 101 178 [9].

1 Scope

The present document specifies that set of technical requirements for Digital Enhanced Cordless Telecommunications (DECT) Fixed Part (FP) and DECT Portable Part (PP) necessary for the support the Open Data Access Profile (ODAP).

It defines a common architecture and protocol to support home and industrial sensors, alarms, telematics and other Machine-to-Machine (M2M) messages. The specification aims at ensuring low cost terminals and an easy to use application interface. It covers three interfaces:

- DECT air interface;
- The client to server ODAP application interface;
- Tethered PP interface to external equipment.

2 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 which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

[1] ETSI EN 300 175-1: "Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 1: Overview".

[2] ETSI EN 300 175-2: "Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 2: Physical Layer (PHL)".

[3] ETSI EN 300 175-3: "Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 3: Medium Access Control (MAC) layer".

[4] ETSI EN 300 175-4: "Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 4: Data Link Control (DLC) layer".

[5] ETSI EN 300 175-5: "Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 5: Network (NWK) layer".

[6] ETSI EN 300 175-6: "Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 6: Identities and addressing".

[7] ETSI EN 300 175-7: "Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 7: Security features".

[8] ETSI EN 300 175-8: "Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI); Part 8: Speech coding and transmission".

[9] ETSI TR 101 178: "Digital Enhanced Cordless Telecommunications (DECT); A High Level Guide to the DECT Standardization".

[10] ETSI EN 300 444: "Digital Enhanced Cordless Telecommunications (DECT); Generic Access Profile (GAP)".

[11] ETSI EN 301 406: "Digital Enhanced Cordless Telecommunications (DECT); Harmonized EN for Digital Enhanced Cordless Telecommunications (DECT) covering essential requirements under article 3.2 of the R&TTE Directive; Generic radio".

[12] ETSI EN 301 469: "Digital Enhanced Cordless Telecommunications (DECT); DECT Packet Radio Service (DPRS)".

[13] ETSI TR 102 179: "Digital Enhanced Cordless Telecommunications (DECT); AT command interface; High-level description;".

[14] IETF RFC 2616: "Hypertext Transfer Protocol - HTTP/1.1".

[15] IETF RFC 2822: "Internet Message Format".

[16] IETF RFC 2806: "URLs for Telephone Calls".

[17] ISO/IEC 646: 1991: "Information technology - ISO 7-bit coded character set for information interchange".

[18] ISO/IEC 8859-1: 1998: "Information technology - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1".

[19] IETF RFC 1123: "Requirements for Internet Hosts - Application and Support".

[20] IETF RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax".

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in EN 300 175-1 [1] and the following apply:

HTTP Client: program that establishes HTTP connections for the purpose of sending requests

HTTP Server: program that accepts HTTP connections in order to service requests by sending back responses

NOTE: Any given program may be capable of being both a client and a server; the use of these terms in the present document refers only to the role being performed by the program for a particular connection, rather than to the program's capabilities in general.

ODAP application: implementation dependant entity tailored at the provisioning of reliable communication service

NOTE: Can be used by industry specific processes for exchange of information between participants in the process and which uses an ODAP bearer for the exchange of information with another ODAP application.

ODAP bearer: protected, up to 28 Kbps, data transport service utilizing the 3,1 kHz teleservice specified in GAP, EN 300 444 [10]

ODAP client: type of ODAP application that offers direct service to an industry specific device

NOTE: e.g. alarm, sensor, controller, etc. for wireless exchange of information with an ODAP server

ODAP server: type of ODAP application that controls one or more ODAP clients

NOTE: may provide means for informing a user for the behaviour of the devices served by the ODAP clients and allow actions to be taken upon

3.2 Symbols

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

M	mandatory to support (provision mandatory, process mandatory)
O	optional to support (provision optional, process mandatory)
I	out-of-scope (provision optional, process optional) not subject for testing
C	conditional to support (process mandatory)
N/A	not applicable (in the given context the specification makes it impossible to use this capability)

Provision mandatory, process mandatory means that the indicated feature service or procedure shall be implemented as described in the present document, and may be subject to testing.

Provision optional, process mandatory means that the indicated feature, service or procedure may be implemented, and if implemented, the feature, service or procedure shall be implemented as described in the present document, and may be subject to testing.

3.3 Abbreviations

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

A/N	Acknowledged/Not acknowledged
AC	Access Code
ACK	Acknowledge
CA	Client Application
CC	Call Control
CI	Common Interface
CMC	Cordless Multimedia Communication
C-plane	Control plane
CRC	Cyclic Redundancy Check
C_S	higher layer signalling Channel (slow)
DCK	Derived Cipher Key
DECT	Digital Enhanced Cordless Telecommunications
DF	Data Framer
DLC	Data Link Control
DPRS	DECT Packet Radio Service
EN	European Norm
ES	End System
FB_N	Frame Buffer (uNprotected)
FP	Fixed Part
FR	Frame Relay
F-SMS	Fixed line Short Message Service
FT	Fixed Termination
GAP	Generic Access Profile
HTTP	Hypertext Transfer Protocol
ID	Identity
IETF	Internet Engineering Task Force
I_N	higher layer Information channel (uNprotected), (logical channels to the MAC layer)
I_P	higher layer Information channel (Protected), (logical channels to the MAC layer)
IP	Internet Protocol
IPUI	International Portable User Identity
IWU	Inter Working Unit
LAN	Local Area Network
LAPC	a DLC layer C-plane protocol entity
Lb	a DLC layer C-plane protocol entity
Lc	a DLC layer C-plane protocol entity
LI	Length Indicator
LRMS	Low Rate Messaging Service
LU1	LAP-U service 1
M2M	Machine-to-Machine

MAC	Medium Access Control
MM	Mobility Management
NWK	NetWorK
ODAP	Open Data Access Profile
PBX	Private Branch eXchange
PDU	Protocol Data Unit
PHY	PHYsical layer
PP	Portable Part
PSTN	Public Switched Telephone Network
PT	Portable Termination
RF	Radio Frequency
RFC	Request For Comment
RLL	Radio in the Local Loop
RN	Receive number
SA	Server Application
SAP	Service Access Point
SDU	Service Data Unit
SEL	SELective
SN	Send Number
SP	SPace
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TI	Transaction identifier
TRUP	TRansparent UnProtected service
U-plane	User plane
URI	Uniform Resource Indicator
URL	Uniform Resource Locator
USB	Universal Serial Buss

4 General

The present document is a part of the extensive set of DECT standards produced by EP DECT and covering various aspects and needs of communications markets and end users with services ranging from voice to data to multimedia.

The main focus of the present document is on provision of a GAP, EN 300 444 [10] based packet Cordless Multimedia Communication (CMC) End System (ES) that allows distributing the burden of the data applications and transport protocols implementation between the DECT Portable Part (PP) and the DECT Fixed Part (FP) targeting of putting the complexity into the FP and reducing the complexity, and hence reducing the cost, of the Portable Parts.

The present document provides a generic low-rate messaging encapsulation transport mechanism over the GAP, EN 300 444 [10] air interface capable of satisfying the needs of various types of devices, e.g. industrial and household sensors, alarms, machines, surveillance cameras, etc.

From implementation point of view the present document aims at economy of scale and quick product market introduction, i.e. utilization of widely available protocols, software and hardware, easy to use application interface for installers/system integrators, low power consumption.

The present document is organized in four basic groups:

- Overview to the basic ideas and system architecture (clauses 5 and 6);
- Features, services and procedures requirements lists (clause 7);
- Requirements description (clauses 8 to 12);
- Application annexes (currently only annex A).

Wherever it is possible reference to external standards is used instead of detail description.

5 Overview

DECT is a standard for short range, low power, digital cordless communications which provides access to a great variety of external (to DECT) networks. The interworking with such networks is out of the scope of the present document - depending on the implementation an appropriate DECT interworking profile standard may be used or appropriate solution provided.

The Open Data Access Profile (ODAP) specifies a reliable multimedia transport protocol. The profile provides a low rate (up to 28 Kbps) data service utilizing a single symmetric DECT channel with an option of switching the service to a high data rate.

The low-rate ODAP concept uses the requirements specified in GAP, EN 300 444 [10] to establish a reliable U-plane channel over which data packets containing various medias' information are transferred.

The reliable transmission of the data packets is achieved through the implementation of a data-link-protocol-like application called ODAP Data Framer (DF) which resides immediately above the GAP, EN 300 444 [10] DLC U-plane protocol; the ODAP Data Framer is very similar to the DLC U-plane protocol defined in DPRS, EN 301 469 [12] and in addition to framing provides CRC and retransmission mechanism to cope with errors due to RF interference. As an option, error correction in DECT MAC (Layer 2 protected mode) is provided as well.

NOTE 1: The placing of the data handling outside of the standard DECT protocol stack, i.e. out of the DLC layer, has be chosen to provide easy update to implementations that have distributed physically DECT protocols stack, such as PBX or RLL systems.

The high-rate ODAP concept provides means of switching a low-rate ODAP connection to a call that may utilize multi bearers and multi connections between a PT and a FT. After the switching the requirements specified in DPRS, EN 301 469 [12] shall basically apply.

The reliable exchange of messages over the DECT U-plane between the PT and FT is based on the utilization of a simplified version of the HTTP [14].

The HTTP [14] has been chosen due to three main reasons:

1) A reliable message exchange over the DECT U-plane requires a reliable protocol handled over the DECT U-plane. A C-plane protocol, e.g. the DECT Low Rate Message Service (LRMS), will need means for synchronization and will add unnecessary complexity.

2) To satisfy the general system architecture (see next clause for further details) where the portable device (e.g. PP) plays the role of a simple client that delivers or retrieves information when needed, and the fixed device (e.g. a FP) servers as a server that may store, request, provide and receive information and basically knows what to do with the information.

3) The provision of a standardized interface for third party application development.

NOTE 2: A "fixed device" may be a standalone FP implemented as a server or may comprises a FP connected to an external server, e.g. an IP server, via any suitable protocol/network; whereas, a "portable device" may be a PP or an end system comprising a PP connected to an external device, e.g. sensor.

For ODAP end systems comprising a PP connected to an external device, ODAP specification provides requirements for implementation of AT commands set used over a standard serial or USB physical interface between the two parts of the end system.

6 Reference configuration

6.1 General

Figure 1 provides the basic reference architecture model for ODAP. It is based on the following assumptions:

1) Use the transport pipe provided by GAP, EN 300 444 [10] (32 Kbps) for low rate Layer 2 unprotected data transmission. Allow in addition, optionally, provision of Layer 2 protected mode.

2) Allow various very simple, messaging like, media applications to use the transport.

3) ODAP applications shall run in client/server manner. The Client Application (CA) is assumed to be located into the PT or into an external device connected to the PT. The Server Application (SA) is assumed to be located into the FT or into an external device connected to the FT. A number of CAs can be handled by one and the same PT.

4) The various options in regard to number of devices involved and the exact location of the client/service application implies 6 distinctive interfaces. Only three of these should be in the scope of the present document:

   - H1 - the AT commands interface between a client and the PT;
   - H2 - the DECT air interface between a PT and a FT; and,
   - H3 - The ODAP client - server application interface. All the rest, i.e. H0, H4 and H5 are left to the designer's are dependent on the particular implementation architecture.

5) To ensure reliable message transport over the DECT U-plane a message transfer protocol based on the commonly used HTTP shall be specified. The protocol should have local implication and because it is to be used only between the FT and PT no extensive addressing is required.

6) The GAP, EN 300 444 [10] transport does not provide error free service. Consequently, in order to provide a single standard interoperable platform, the ODAP requires the specification of an error handling layer. This layer is called in the present document the ODAP Data Framer (DF).

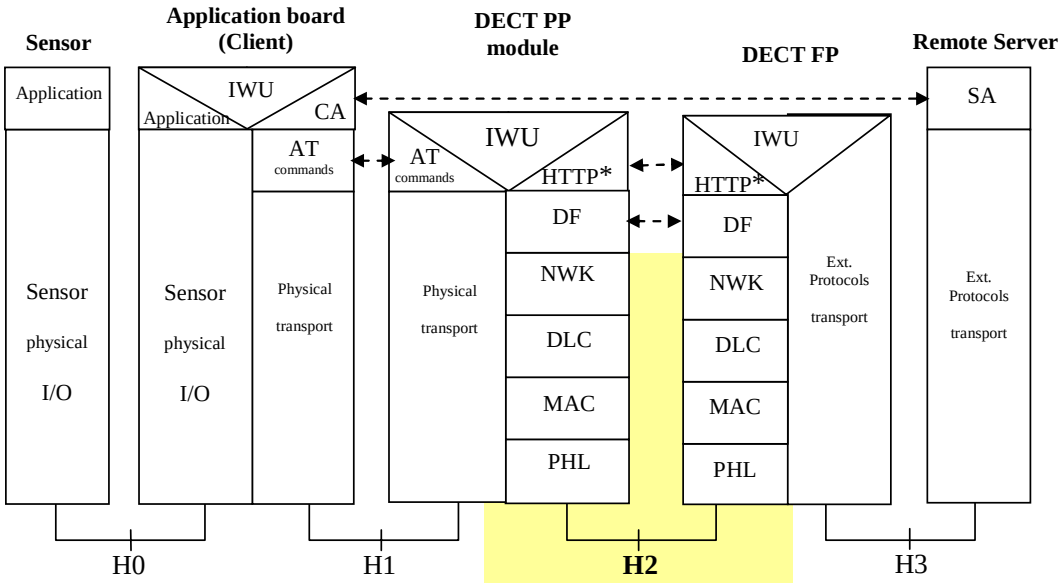
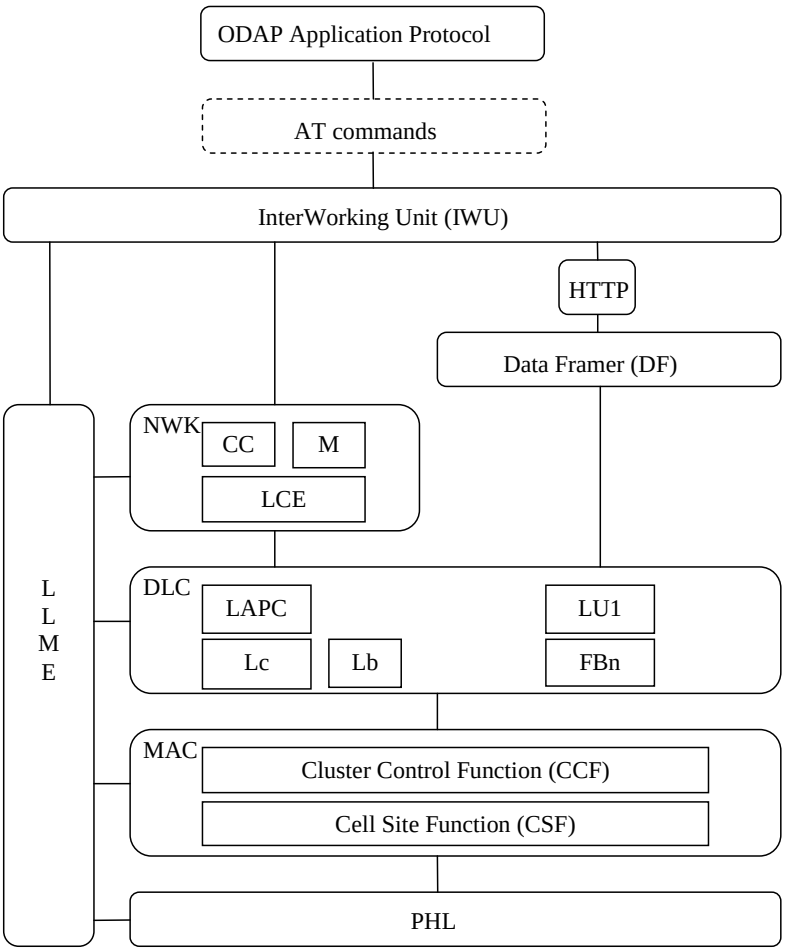


Figure 1: Basic system architecture reference configuration

The basic protocol reference model of ODAP is shown on figure 2. For simplicity multiple instances of protocol entities are not shown.



NOTE: Whether the AT commands will be used depends on the implementation.

Figure 2: Basic protocol reference configuration

6.2 Example scenarios

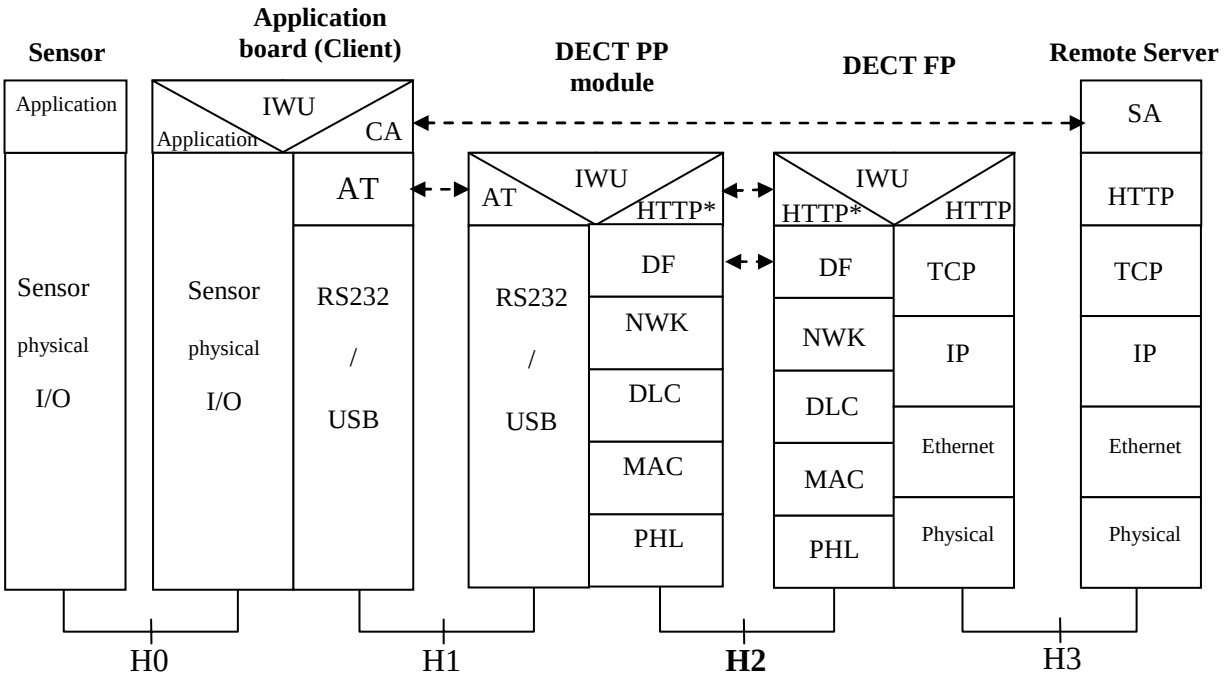
This non bounding clause provides a number of examples reflecting different applications scenarios.

6.2.1 A sensor application - remote server

A possible ODAP applications is the provision of means enabling e.g. a cordless sensor to be capable of providing indications to a server. The Machine-to-Machine (M2M) services scenario is a generalization of such an application.

Figure 3 provides a basic reference module for a M2M ODAP implementation that utilizes a remote server to deal with the information provided from the sensor(s). A number of sensors (clients) can be served by the same PT - not shown on the figure for simplicity. The server resides on a LAN and uses IP to communicate with the FT.

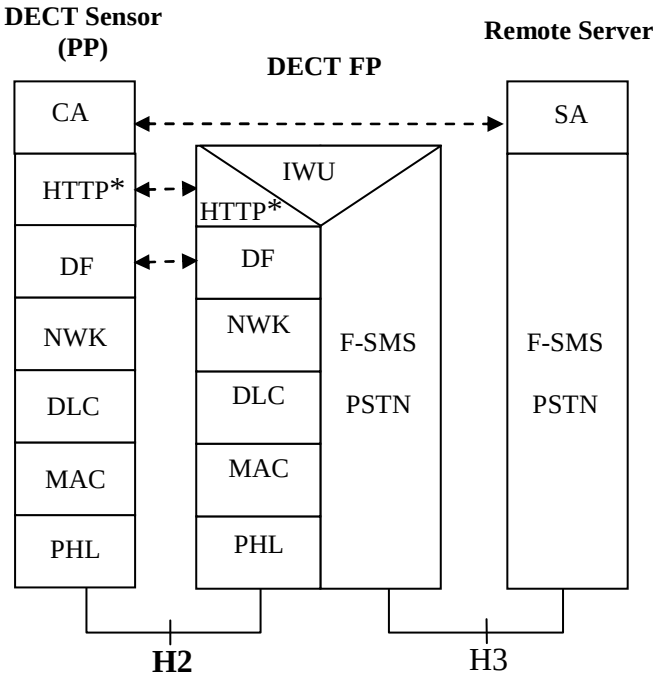
It is important to notice that the figure does not limit physically an implementation, i.e. some of the depicted entities may be implemented in one and the same physical device. Furthermore, the protocols between the DECT FP/PT and the external devices, i.e. the client and the remote server, shall be seen again only as an example.



NOTE: This scenario assumes that FP has sufficient knowledge about the addressing needed to communicate with the remote client.

Figure 3: M2M ODAP reference model

Figure 4 depicts the same application scenario however the remote server is not a LAN based server rather it utilizes the F-SMS transport protocol to communicate with the FT and the client now resides in the PT.



NOTE: This scenario assumes that FP has sufficient knowledge about the telephone number needed to communicate with the remote client via F-SMS.

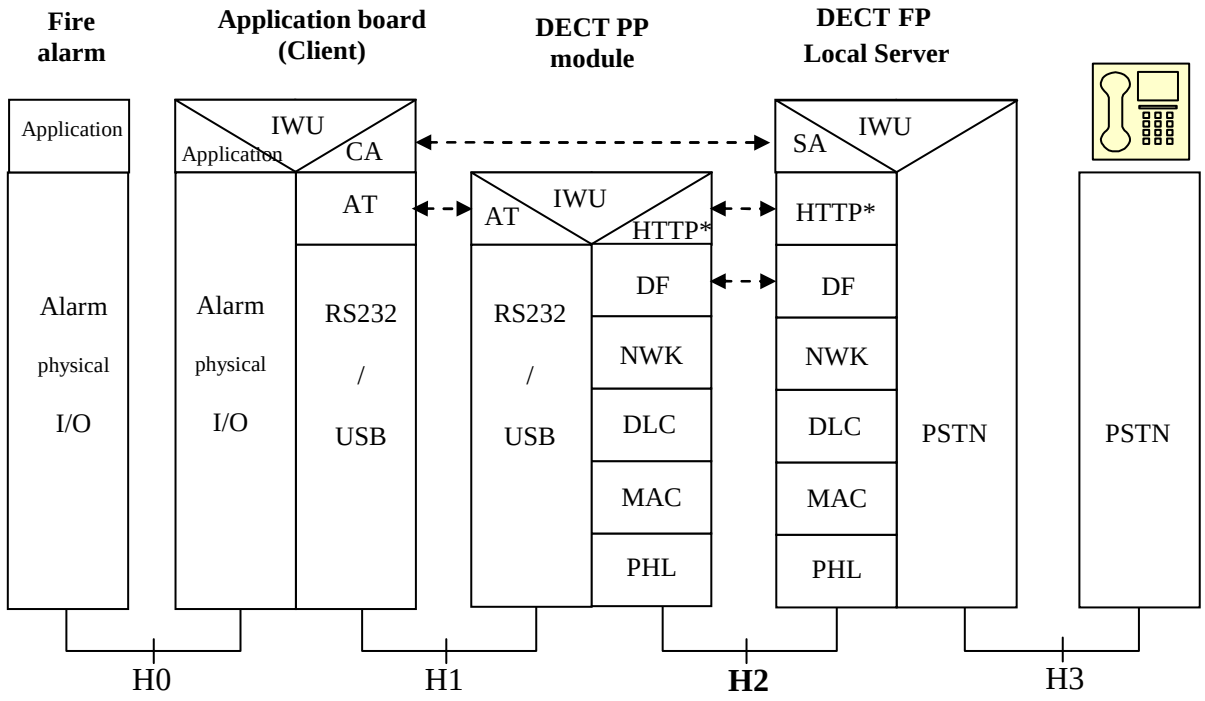
Figure 4: M2M ODAP reference model

6.2.2 A fire alarm application - local server

A possible ODAP applications is the provision of means enabling e.g. a cordless fire alarm to be capable of providing indications to a local server in the FT. What will the server do with the information could be pre-set by the user.

Figure 5 provides a basic reference module for a ODAP implementation that utilizes a DECT FT server to deal with the information provided from the alarm(s). A number of alarms (clients) can be served by the same PT - not shown on the figure for simplicity.

It is important to notice that the figure does not limit physically an implementation, i.e. some of the depicted entities may be implemented in one and the same physical device. Furthermore, the protocols between the DECT PT and the external device, i.e. the client and the PT, shall be seen again only as an example. The external to FT connection via the PSTN to a remote phone is provided as an example for an action that the local server could take, e.g. ring a remote phone.



NOTE: This scenario assumes that FP has sufficient knowledge about the telephone number needed to communicate to the remote phone.

Figure 5: M2M ODAP reference model

7 Interoperability requirements

7.1 Feature/service definitions

For the purposes of the present document the feature definitions of GAP, EN 300 444 [10] apply.

Features/services specific to the present document are defined/described in the relevant clauses.

7.2 Requirements tables

The ODAP protocol utilizes features/services/procedures specified in GAP, EN 300 444 [10]. To satisfy the specific ODAP requirements some of these features/services/procedures have been modified and some new have been added. This clause lists the features, services and procedures and their support status relevant for ODAP.

In addition ODAP equipment may implement any GAP, EN 300 444 [10] feature/service not listed in the present document - the support of them is optional. If supported, the requirements of GAP, EN 300 444 [10] shall fully apply.

The tables listed in this clause define all the protocol elements i.e. features, services, and procedures which are mandatory, optional, or conditional under the provision of another protocol element, or outside the scope of the present document, or in some context not applicable (for the interpretation of the symbols in the status column, see clause 3.2). All optional elements shall be process mandatory according to the procedures described in the present document.

Protocol elements defined as mandatory, optional or conditional in this clause are further defined in the following clauses in detail, either explicitly and/or as references to other standards.

The requirements of EN 301 406 [11] shall be met by all equipment conforming to the present document.

7.2.1 ODAP service provision specific features/procedures

Table 1 provides summary of the ODAP specific features/procedures and their required status for support.

Table 1: ODAP requirements

Feature	Procedure	Clause	PT	FT
FT ODAP features support indication		9.1, 9.1.1, 9.1.2, 9.1.2.1	M	M
	FT default ODAP support indication	9.1.2.1.1	M	M
	FT ODAP Layer 2 protected mode support indication	9.1.2.1.2	O	O
PT ODAP features support indication		9.1, 9.1.1, 9.1.2, 9.1.2.2	M	M
	PT default ODAP support indication	9.1.2.2.1	M	M
	PT ODAP Layer 2 protected mode support indication	9.1.2.2.2	O	O
ODAP call setup		9.2	M	M
	ODAP Call Setup	9.2.1	M	M
	ODAP call features/parameters negotiation	9.2.2	O	O
ODAP sleeping mode		9.3	C1	M
	ODAP PT Mode of operation	9.3	C1	M
ODAP subscription		9.4	M	M
	ODAP Subscription	9.4	M	M
ODAP layer 2 protected mode		9.5	O	O
	DECT MAC MOD2	9.5	M	M
ODAP Data Framer		10	M	M
ODAP media transfer		11, 11.1	M	M
	Hyper Media Transfer	11.2	M	M
ODAP application communication protocol		12	M	M
	Various	12	(See note 1)	(See note 1)
External PT communication		13	O	N/A
	AT commands	13.1	M	N/A
ODAP application specific services		Annex A	M	M
	Alarms and Sensors communication	Annex A	C2 (See note 2)	C2 (See note 2)
C1: IF server initiated application procedures supported THAN M, ELSE I. C2: IF application=(Alarm OR Sensor) M, ELSE I. NOTE 1: For the relevant procedures and their status see clause 12.1.1. NOTE 2: For the relevant procedures and their status see clause A.2.				

7.2.2 ODAP service provision general features/services

Tables 2, 3 and 4 provide summary of the ODAP general features and services and their required status for support. The relevant procedures are specified in GAP, EN 300 444 [10] and when necessary modified in the present document. The modifications specified in the present document have precedence.

Table 2: NWK features status

Item no.	Name of feature	Reference	Support status	
			PT	FT
ODAP-N.1	Outgoing call	[10] (See note)	M	M
ODAP-N.2	Off hook	[10] (See note)	M	M
ODAP-N.3	On hook (full release)	[10]	M	M
ODAP-N.4	Dialled digits (basic)	[10]	M	M
ODAP-N.5	Incoming call	[10]	M	M
ODAP-N.6	Authentication of PP (store DCK)	[10]	M	M
ODAP-N.7	Location registration	[10]	M	M
ODAP-N.8	On air key allocation	[10]	M	M
ODAP-N.9	Alerting	[10]	M	M
ODAP-N.10	Encryption activation FT initiated	[10]	M	M
ODAP-N.11	Subscription registration procedure on-air	[10] (See note)	M	M
ODAP-N.12	Link control	[10]	M	M
ODAP-N.13	Terminate access rights FT initiated	[10]	M	M
NOTE: The present document introduces changes to these items.				

Table 3: DLC services status

Item no.	Name of service	Reference	Support status	
			PT	FT
ODAP-D.1	LAPC class A service and Lc	[10]	M	M
ODAP-D.2	C_S channel fragmentation and recombination	[10]	M	M
ODAP-D.3	Broadcast Lb service	[10]	M	M
ODAP-D.4	Encryption activation	[10]	M	M
ODAP-D.5	LU1 TRUP Class 0/min_delay	[10]	M	M
ODAP-D.6	FU1	[10]	M	M

Table 4: MAC services status

Item no.	Name of service	Reference	Support status	
			PT	FT
ODAP-M.1	General	[10]	M	M
ODAP-M.2	Continuous broadcast	[10]	M	M
ODAP-M.3	Paging broadcast	[10]	M	M
ODAP-M.4	Basic connections	[10]	M	M
ODAP-M.5	C_S higher layer signalling	[10]	M	M
ODAP-M.6	Quality control	[10]	M	M
ODAP-M.7	Encryption activation	[10]	M	M
ODAP-M.8	Extended frequency allocation	[10]	M	M
ODAP-M.9	Bearer Handover, intra-cell	[10]	M	M

8 Requirements description rules

The following clauses define the process mandatory requirements which are in the scope of the present document.

All protocol elements listed in the following clauses are process mandatory, i.e. the FT and PT depending on their role in the procedure shall send or shall receive and process the relevant protocol elements as listed in the respective tables if not explicitly stated as being optional.

9 DECT protocol elements of procedures

This clause specifies the DECT protocol procedures, messages and information elements required in the present document.

This profile does not prevent any PT or FT from implementing requirements not specified in the profile. A PT or FT receiving an unsupported message or information element, which it does not recognize, shall ignore it, as specified in EN 300 175 parts 3 [3] to 5 [5].

9.1 ODAP features/parameters support indication/negotiation

9.1.1 General

Indication of "ODAP support" implies provision at least of the features, procedures, parameters and values which have been set as default (mandatory) for support for all ODAP compliant equipment; some terminals may provide additional (set to be optional for ODAP) features/parameters; terminals may also provide support for features/parameters that are out of the scope of the present document.

Terminals that claim support of the ODAP mandatory and optional features, procedures, parameters and values are required to comply with the requirements specified in the present document.

The ODAP features/parameters and their range of values set to be mandatory are listed in the table 5:

Table 5: ODAP default (mandatory) features/parameters/values

Parameter/Feature		Support status/Value	
		FT	PT
3	ODAP Layer 2 protection	No	No
4	Maximum SDU size	128 Bytes	128 Bytes
5	SDU Life time	Unlimited	Unlimited
6	Window size	4	4
7	Modulo	8	8

9.1.2 ODAP features support indication

Terminals that support ODAP shall announce "ODAP support":

- FPs by broadcasting "ODAP support" in the Qt message Extended Fixed Part capabilities.
- PPs shall indicate "ODAP support" during subscription and location registration in the <<Terminal Capability>> information element.

Terminals that support additional optional ODAP features shall indicate this explicitly. Terminals that support additional DECT profiles shall indicate this as specified in the relevant profile specification, e.g. terminals that support GAP voice as well shall provide the relevant indication as specified in GAP, EN 300 444 [10].

The ODAP optional features that, if supported, need to be indicated are listed in the table 6:

Table 6: ODAP optional features/parameters/values

Parameter/Feature		Support status/Value	
		FT	PT
1	ODAP normal Layer 2 protection	Yes	Yes

Some ODAP optional parameters need to be set up per call (see clause 9.2.2).

If a particular indication has been said to be mandatory for transmission it shall also be understood to be mandatory for reception and interpretation.

9.1.2.1 FT ODAP features support indication

An ODAP compliant FT shall support the broadcasting of MAC layer {Q3 - Standard FP capabilities} and {Q4 - Extended FP info} messages as specified in GAP, EN 300 444 [10] and the in EN 300 175-3 [3] with the modifications specified in this clause.

9.1.2.1.1 FT default (mandatory) ODAP support indication

Every FT that claims compliance to ODAP shall support the broadcast of the following information:

Table 7: Values used in standard {FP capabilities} (QH=3)

Bit Nr.	Attribute	Value	Note
a12	Extended FP info	1	
a28	I_N_normal_delay	1	
a36	Standard authentication required	1	
a37	Standard ciphering supported	1	
a38	Location registration supported	1	
a44	Access rights requests supported	0/1	The FP can toggle this bit to enable or disable on air subscription.

Table 8: Values used in Extended FP capabilities (QH=4)

Bit Nr.	Attribute	Value	Note
a27	Basic ODAP supported	1	Mandatory - implies all ODAP default (mandatory) requirements are supported (see table 5)

9.1.2.1.2 FT ODAP Layer 2 protected mode support indication

Every FT that claims the support of the optional ODAP feature "Layer 2 protected mode" shall support the broadcast of the following information in addition to the default ODAP features support indication (see clause 9.1.2.1.1).

Table 9: Values used in standard {FP capabilities} (QH=3)

Bit Nr.	Attribute	Value	Note
a30	Ip_error_correct	1	Optional - ODAP layer 2 normal protected mode

9.1.2.2 PT ODAP features support indication

To indicate its support for the ODAP mandatory features the PT shall be able to send the <<Terminal capability>> and for the support of the ODAP optional features PT should be able to send the <<Setup Capability>> information elements as specified in GAP, EN 300 444 [10] and in EN 300 175-5 [5] and the FP shall be able to receive them at least in the {ACCESS-RIGHTS-REQUEST} and the {LOCATE-REQUEST} messages.

The following text together with the associated clauses define the mandatory requirements with regard to the present document.

9.1.2.2.1 PT default (mandatory) ODAP support indication

Every PT that claims compliance to ODAP shall support the indication of the following information:

Table 10: Values used within the <<TERMINAL CAPABILITY>> information element

Information element	Field within the information element	Standard values within the field/information element	Normative action/comment
<<Terminal capability>>	<ext 3>	1	No other octets from Octet group 3 included (see note 1)
	<Tone capability>	001	No tone capability
	<Display capability>	0001	No display
	<ext 4>	0	
	<Profile indicator_1>	0000000	(See note 2)
	<ext 4a>	0	
	<Profile indicator_2>	0000000	(See note 2)
	<ext 4b>	0	
	<Profile indicator_3>	0000000	(See note 2)
	<ext 4c>	1	No other octets from Octet group 3 included (see note 3)
	<Profile indicator_4>	1000000	Basic ODAP supported (See note 4)
	<ext 5>	1	
	<Control codes>	000	Not specified
	<ext 6>	0/1	
	<Blind Slot Indication>	All	As appropriate
	<SPx>	All	As appropriate
	<ext 6a>	1	Present if <ext 6> = 0
	<SPx>	All	As appropriate
NOTE 1: Implies that <Slot type capability> = Full slot. NOTE 2: An ODAP compliant terminal is not required to support any other DECT profiles. NOTE 3: Implies that 2-level modulation scheme is used for A + B + Z. NOTE 4: Implies that default (Mandatory) ODAP requirements are supported (see table 5).			

9.1.2.2.2 PT ODAP Layer 2 protected mode support indication

Every PT that claims the support of the optional ODAP feature "Layer 2 protected mode" shall support the provision of the following information in addition to the default ODAP features support indication (see clause 9.1.2.2.1).

Table 11: Values used within the <<SETUP CAPABILITY>> information element

Information element	Field within the information element	Standard values within the field/information element	Normative action/comment
<<Setup capability>>			
	<ext 3>	1	No other octets from Octet group 3 included
	<Protocol Discriminator>	010	ODAP
	<Setup>	01	Normal
	<Page>	01	Normal
	<ext 4>	1	No other octets from Octet group 4 included
	<Service_settings_1>		
		xxxxxx1	Ip_error_correct supported (ODAP Layer 2 normal protected mode)

The PT shall not send this information, even if the PT supports ODAP Layer 2 protected mode, if the FT does not broadcast support for ODAP Layer 2 protected mode.

If both, FT and PT, indicate support of Layer 2 protected mode, then all communication during the existence of the subscription shall be performed using the Layer 2 protected mode. If one of the terminal does not indicate support of Layer 2 protected mode, all communication during the existence of the subscription shall be performed without using the Layer 2 protected mode.

9.2 ODAP call setup procedure

9.2.1 ODAP default call setup

In the case of outgoing call the procedure shall be performed as defined in GAP, EN 300 444 [10], clauses 8.2, 8.6 and 8.10; the provision of the requirements specified in clauses 8.3, 8.4 and 8.5 is optional for both PT and FT. Provision of called party number or addressing information is not required - it is assumed that FT shall have information for the location of the server and the relevant addressing information needed to route the call to it.

In the case of an incoming call the procedure shall be performed as defined in GAP, EN 300 444 [10], clauses 8.11, 8.12, 8.13 and 8.10.

The following requirements additionally apply in regard to the present document (i.e. if not explicitly indicated GAP, EN 300 444 [10] requirements apply).

Figure 6 (outgoing) and figure 7 (incoming) provide examples for basic call setup messages flows. The exact place where some MM related procedures take place is informative.

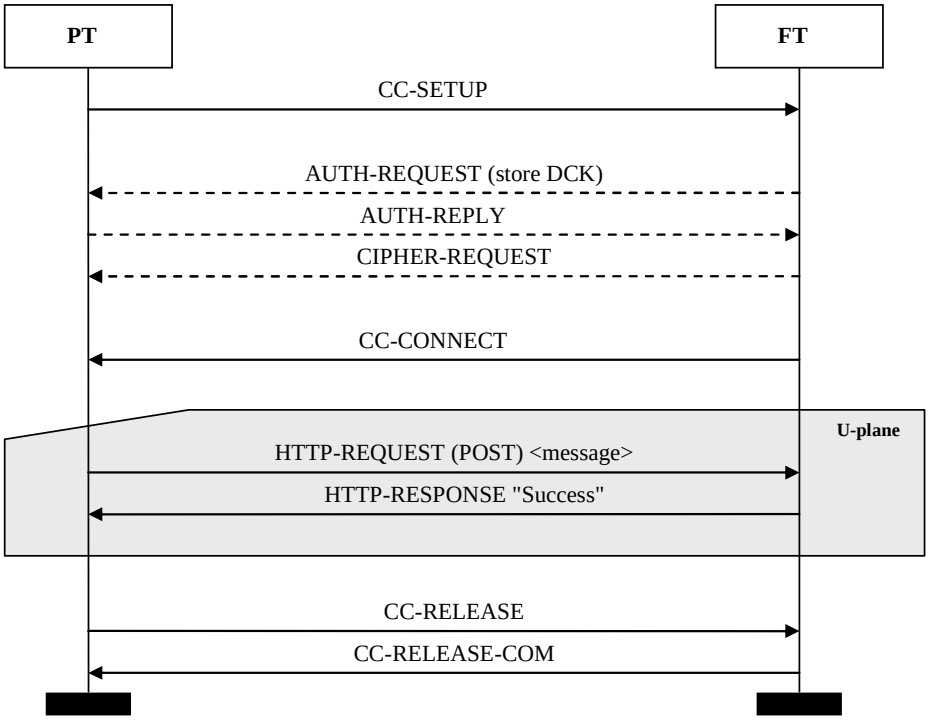


Figure 6: Outgoing call message flow example

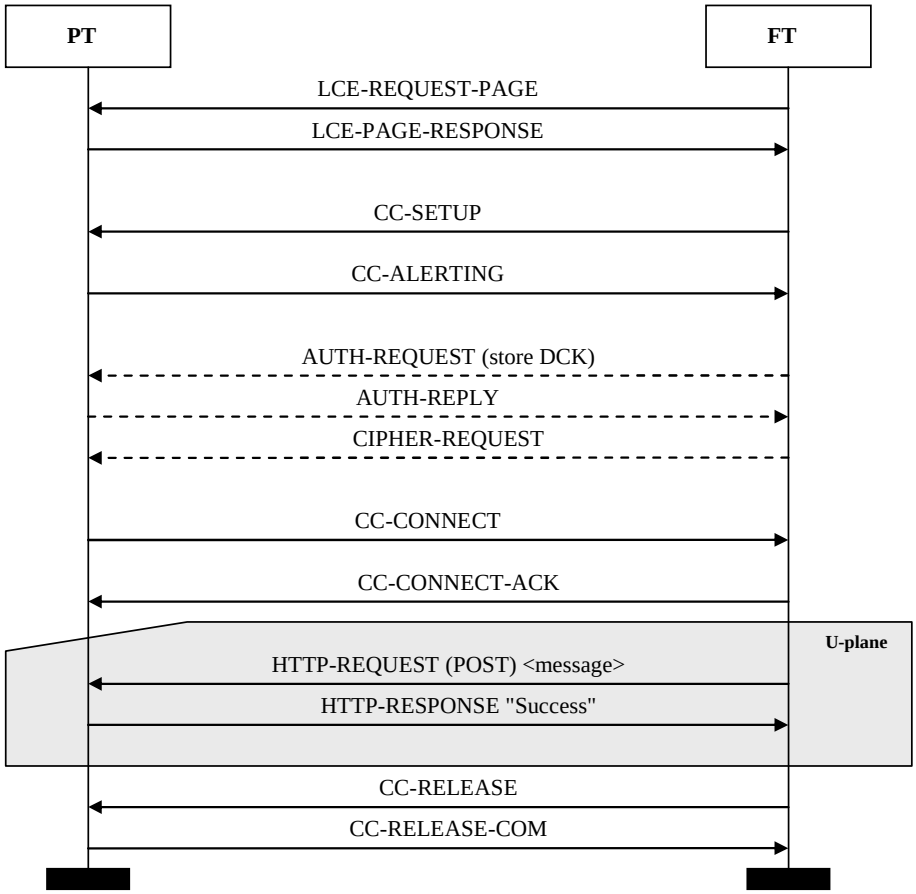


Figure 7: Incoming call message flow example

The {CC-SETUP} message shall always indicate the type of the call, i.e. "ODAP" call.

Table 12: Values used within the {CC-SETUP} message ODAP default

Information element	Field within the information element	Standard values within the field/information element	Normative action/comment
<<Basic service>>			
	<Basic service>	1111	Other
<<Call Attributes>>			
	<Coding standard>	00	DECT standard
	<Network layer attributes>	00101	ODAP Basic data
	<C-plane class>	010	Class A; shared
	<C-plane routing>	0000	CS only
	ext5	1	No octet 5a
	<U-plane symmetry>	00	Symmetric
	<LU identification>	00001	LU1
	ext6	1	No octet 6a
	<U-plane class>	001	Class 0 normal_delay
	<U-plane frame type>	0001	FU1

9.2.2 ODAP optional call features/parameters negotiation procedure

An ODAP terminal may support additional (optional) features/parameter settings. Before proceeding with a call both sides shall agree on the features and/or parameters that shall apply to the call. Otherwise the call shall be released.

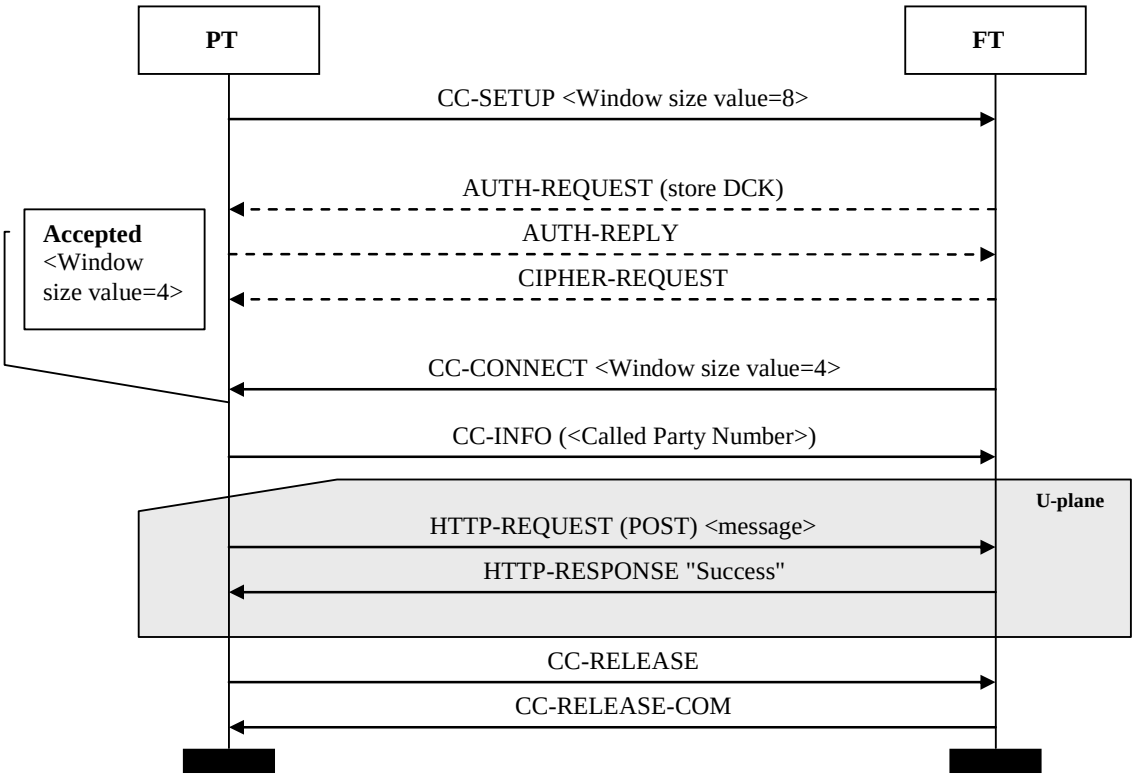


Figure 8: Outgoing call optional parameters negotiation message flow example

The features/parameters indication/negotiation procedure is based on submission of set of values by the call originator followed by a respond from the call destination with the same or different values. Return of the same parameters, or not providing indication of certain parameters, shall be understood as acceptance of the call originator suggestion. Return of "lower" values is only allowed and those should be accepted by the call originator; alternatively, the call may be released and new call initiated with a different proposition.

In case the both sides are unable to negotiate acceptable values they both shall resolve to the default values indicated in table 5.

The procedures shall be perform as specified in EN 301 469 DPRS, clause 12.5, with the following modifications/clarifications.

To indicate/negotiate the exact parameters of the requested service the initiating side shall include into the {CC-SETUP} message:

- a << IWU ATTRIBUTES >> information element used to indicate the type/characteristics of the service requested;
- a << CALL ATTRIBUTES >> information element;
- a << CONNECTION ATTRIBUTES >> information element;
- a << TRANSIT DELAY >> information element;
- a << WINDOW SIZE >> information element.

If the inclusion of these information elements will lead to message with length longer than 63 octets then successive {CC-INFO} messages shall be used to carry them.

If the parameters indicated are not acceptable and support of negotiation is indicated in the << IWU-ATTRIBUTES >> the receiving side may attempt negotiation if different services are possible, otherwise the call shall be rejected using the abnormal call release procedure as defined in GAP, EN 300 444 [10], clause 8.7.

For negotiation of << IWU-ATTRIBUTES >> and << CONNECTION ATTRIBUTES>> the peer attribute negotiation procedure as defined in EN 300 175-5 [5] , clause 15.2.5, shall be used. For negotiation of the << TRANSIT DELAY >> and the << WINDOW SIZE >> the operating parameter negotiation procedure as defined in EN 300 175-5 [5] , clause 15.2.4 shall be used. The following text together with the associated clauses defines the mandatory requirements with regard to the present document.

If the receiving side decides to attempt negotiation it shall continue the call set-up procedure by including one alternative service description returning the appropriate <<CONNECTION ATTRIBUTES >> and/or <<CALL ATTRIBUTES >> and/or <<IWU-ATTRIBUTES >> and/or <<WINDOW SIZE>> and/or <<TRANSIT DELAY>> elements in the first response message (i.e. {CC-CONNECT} for FT, {CC-ALERTING} for PT).

If one or more of the values are acceptable the receiving side shall return unmodified parameters as formal acceptance of these unmodified values.

The values that can be used during negotiation are indicated into the table 13. The negotiable values are indicated with shaded boxes.

Table 13: Values used within the relevant message

Information element	Field within the information element	Standard values within the field/information element	Normative action/comment
<< IWU attributes >>			
	< Coding standard >	01	Profile defined coding.
	< Profile >	00101	ODAP basic data.
	< Negotiation Indicator >	010	Peer attribute negotiation.
	< Profile Subtype >	1000	ODAP messaging.
	<Max SDU Size - MS 7 bits>	0000000	Mandatory minimum (see table 5).
		All	Optional
	<Max SDU Size - LS 7 bits>	11111111	128 - Mandatory minimum (see table 5).
		All	Negotiable
<< Call attributes >>			Mandatory (see table 12).
<< Connection attributes >>			E.g. to indicate Layer 2 protected mode.
	< Symmetry >	001	Symmetric.
	< Connection identity >	0000	Not yet numbered.
	<ext4>	1	
	< Target bearers (P = > F direction) >	1	
	< ext5 >	1	
	< MAC slot size >	100	Full slot.
	< MAC service P = > F >	0001	I_N; normal delay. See note.
		0011	I_P; Mod-2 correct. See note.
	< Ext6 >	1	
	< C_F channel attributes P = > F >	000	C_F never (C_S only).
	< MAC packet life time P = > F >	0 to 7	Values > 0 only for I_P; Mod-2 correct or ODAP layer 2 protected (Repeater).
	< Ext7 >	1	See note 5.
	< A-attributes >	000	2-level modulation scheme.
	< B-attributes >	000	2-level modulation scheme. See note.
		001	4-level modulation scheme (Optional) See note.
		010	8-level modulation scheme (Optional) (See note).
<< Transit delay >>			The SDU life time.
	< Forward Delay >	0	Infinite (Default - M).
		>0	Optional.

Information element	Field within the information element	Standard values within the field/information element	Normative action/comment
	< Backward Delay >	0	Infinite (Default - M).
		All	It is not required to support different values in Backwards direction.
<< Window size >>			The modulo value used for numbering the PDUs in the DF shall be 2 times higher than the window size value negotiated.
	<ext3>	0	
	< Window size value (forward) >	0000000	The default value if IE not included.
		000000x	Negotiable value.
	<ext3a>	1	
	< Window size value (forward) continue >	0000100	4 - the default value if IE not included.
		xxx1000	Negotiable values.
	<ext4>	0	
	< Window size value (backward) >	0000000	The default value if IE not included.
		000000x	Negotiable value.
	<ext4a>	1	
	< Window size value (backward) continue >	0000100	4 - the default value if IE not included.
		xxx1000	Negotiable values.
NOTE: Negotiation of these values is not allowed. The receiving side shall accept a particular value if supported or release the call otherwise.			

9.3 ODAP PT mode of operation

In order to provide means for handling various possible applications power consumption requirements ODAP defines a special ODAP PT mode of operation.

The ODAP PT mode of operation has impact only on the PT's capabilities of receiving incoming calls and does not impose any restrictions on the PT in regard to making outgoing calls.

The ODAP PT mode of operation defines a PT "sleeping" time and a PT "awake" time.

During the "sleeping" time the PT does not listen for paging from the FT, needs not be locked to the FT and, depending on implementation, may be switched-off. During sleeping time the PT is not capable of receiving incoming calls but could still make outgoing calls if requested by the client.

During the "awake" time the PT shall be at least in idle locked state and may be in active locked state (i.e. engaged in communication with the FT). During idle locked state the normal GAP, EN 300 444 [10] requirements for duty cycle mode shall apply. During the time the PT is awake both incoming and outgoing calls are possible.

Two parameters are defined that determine the ODAP PT mode of operation. The T_sleeping timer defines the time the PT can remain "sleeping"; the T_awake timer defines the time the PT can remain "awake". How long a PT could remain "sleeping" and how long PT could remain "awake" is application dependent. Clients which operation does not require support of incoming calls may continuously remain in sleeping mode. For application clients requirements see annex A.

The values for the T_sleeping and T_awake are assigned by the server during the client registration or client/server configuration as specified in clause 12. It is the client's and the server's responsibility for handling the timers and for instructing the PT/FT to act correspondingly. The T_sleeping and T_awake shall be aligned with the FT's multi-frame counter.

9.4 ODAP subscription

The procedure shall be performed as defined in GAP, EN 300 444 [10], clauses 8.30 and 14. The following text together with the associated clauses define the mandatory requirements with regard to the present document.

The PT is not required to be capable of storing more than one subscription, i.e. more than one pair of IPUI and PARK and associated subscription data.

If a PT handles more than one physical application, e.g. two or more sensors, a separate subscription per application shall be performed; the support for more than one application is optional for the PT. PTs that support more than one application shall be capable of storing more than one subscription, i.e. more than one pair of IPUI and PARK and associated subscription data.

New subscriptions shall be started always with the default PT IPUI. FTs shall be capable of accepting multiple subscriptions from one and the same PT identified by its default IPUI and shall assign different assigned IPUI for each subscription. This IPUI may be used by the FT to differentiate the application that is engaged in a call. Subscription requested with an assigned IPUI shall be treated as a request for overriding of the relevant subscription record.

During subscription the AC used shall be the last 32 bits of the PP's identity provided into the {ACCESS-RIGHTS-REQUEST} message. The provision of manual AC entry means is optional for both PT and FT. If provision of manual AC entry is supported it should be properly communicated to the user, e.g. written in the user manual or other terminal's documentation; care shall be taken not to introduce interoperability problems when communicating with a terminal that supports only the mandatory AC based on the PP identity.

The relation between a PP identity and the relevant AC shall be as follows:

Table 14: IPUT to AC mapping

	MSB	**LSB**
IPUI (Or default IPUI = IPEI)	0000 1001	0001 0011 1111 0000 0000 0000 0000 0001
AC	-	0001 0011 1111 0000 0000 0000 0000 0001
		AC[31] AC[0]

9.5 ODAP Layer 2 protected mode

In addition to the error protection provided by the DF, ODAP equipment may optionally provide error protection in DECT MAC layer. For the ODAP Layer 2 protected mode "normal" the requirements specified in EN 300 175-3 [3], clause 10.8.2 shall apply.

The ODAP Layer 2 protected mode imposes limitation on the length of the DF PDUs. The length of a PDU when ODAP Layer 2 protected mode is used is 32 octets, see ODAP DF (clause 10).

10 ODAP Data Framer (DF)

This clause specifies the requirements in regard to the ODAP Data Framer.

10.1 ODAP Vs. GAP U-plane requirements

10.1.1 Overview in the GAP U-plane requirements

The GAP, EN 300 444 [10] U-plane service does not guarantee data integrity. No error protection is applied, and octets may be lost, erroneous or duplicated. The (lower) frame buffering functions are provided by FB_N: this provides synchronized fragmentation of the continuous higher layer data into I_N logical channel fragments for delivery to the I_N channel in the transmit direction, and recombination of the I_N fragments into a continuous higher layer data flow in the receive direction.

The FU1 frame is designed to meet the special requirements of speech, in particular the capability for introducing minimum delay between the higher layers and the MAC layer. FU1 can also be used for more general data use. In all cases, the actual delay is defined by the MAC layer.

The FU1 frame is a fixed length fragmentation which in the case of GAP, EN 300 444 [10] has 40 octets length:

Bit:	8	7	6	5	4	3	2	1	Octet:
	Higher layer info								1
	Higher layer info								2
	Higher layer info								3
	....								....
	Higher layer info								39
	Higher layer info								40

Figure 9: Frame format type FU1 full slot 2 level modulation

The FB_N entity provides a data buffering function between the service user and the MAC layer. It supplies data to the MAC layer (at the transmit side) or accepts data from the MAC layer (at the receive side) on demand and with minimum delay. The following inter layer (DLC-MAC) operation shall be provided:

Transmit side: on receipt of a MAC_CO_DTR-ind primitive, one complete frame of data shall be submitted to the MAC layer in a MAC_CO_DATA-req primitive.

Receive side: each MAC_CO_DATA-ind primitive shall contain one complete frame of data from the MAC layer.

Data overflow: in the event of data overflow, octets should be deleted, starting with the oldest non-transmitted octet.

Data underflow: in the event of data underflow, selected octets should be repeated in a manner that preserves the data order.

A GAP, EN 300 444 [10] FU1 (class 0/min_delay) frames should deliver the most recent data to the MAC layer. The transmitted contents of each frame are therefore dependant on the exact time that the data is needed by the MAC layer (i.e. on the slot position in use by the MAC layer) AND the time the higher layers data has been submitted to the FB_N entity.

Class 0 provides a transparent interface to the MAC layer. The sending entity submits frames to the MAC layer in the order of arrival. No sequence numbers are added, and no retransmission is used. The MAC layer service should normally provide sequencing (by invoking a constant single bearer full slot connection). At the receiving side, the receiving entity shall immediately deliver frames in the order they are received from the MAC layer. All packets are assumed to contain valid frames, and shall be processed in their order of arrival. Frames indicated as type "unknown" or containing errors (i.e. errors indicated by the MAC layer) are nonetheless delivered to the higher entity, together with the error indication.

10.1.2 Requirements of the utilization of the GAP U-plane for the purposes of the ODAP Data Framer

The service shall be used as it is with the following modifications:

- The DF shall dispatch the user data through the LU1 SAP to FB_N entity data buffering function on regular intervals of 10 ms (one DECT frame)
- The time of submitting should be synchronous to the beginning of a DECT frame.

NOTE: These requirements aim at overcoming possible duplication or lost of data during connection handover (for further information on this issue see annex A). In any case, if arose, any such problem will be handled by the DF retransmission function.

10.2 ODAP Data Framer implementation requirements

10.2.1 General

The service provided by the Data Framer is called DFU10 (the name is chosen to maintain similarity with the DECT LU10 service). The lower function however, is the one used by DLC LU1. The DFU10 is a Frame Relay service that is accessed from the higher layers (e.g. HTTP, IP) through a single DFU10 SAP.

The reference model of the DF is depicted on the figure 10:

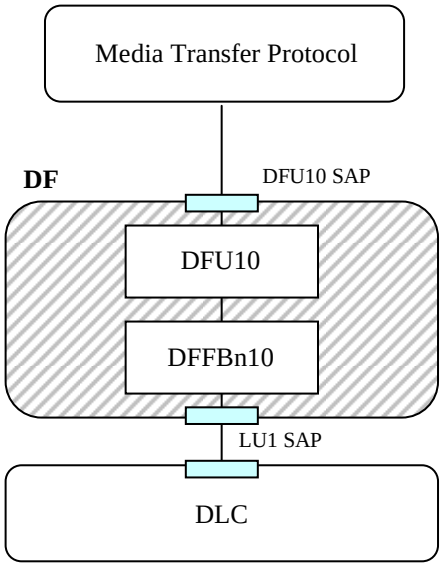


Figure 10: DF protocol architecture

The DFU10 shall operate on a generic field of user data that shall be transferred into and out of the DF as a single SDU. This SDU is assumed to contain one external frame (e.g. one HTTP protocol datagram), but the operation of DFU10 shall be independent of the actual contents of the SDU.

The DFU10 shall provide the following functions:

- peer-to-peer transmission of user data (SDUs) (through DFU10SAP);
- segmentation of SDUs into DF PDUs;
- re-assembly of DF PDUs into user data SDUs;
- management of the SN and RN numbers;
- provision of CRC check sum;
- error correction based on ARQ-type Selective retransmission protocol (SEL).

The DFFB10 is the lower function of the DF. It is based on the FBn functionality of GAP, EN 300 444 [10] and is responsible for:

- transmission/reception of DF PDU to/from DLC LU1 (through LU1 SAP).

10.2.2 DFU10 operation requirements

The SDU size supported shall depend on the application. Simple applications like sensors for example may produce small SDUs. The DFU10 shall support SDU frames of at least 128 and optionally frames of up to 1 528 octets if the application supports IP or F-MMS for example.

A SDU can be originated by different higher data protocol entities.

The SDU shall be segmented, if necessary, into fixed length segments. The segmentation shall start from the least significant byte and this shall be understood as the first segment. The maximum length of each segment depends on the Layer 2 protected mode being supported or not. If Layer 2 protected mode is not provided (the default mandatory ODAP requirement), each segment shall be of length maximum 35 octets (i.e. the last segment may be shorter); if Layer 2 protected mode is provided each segment shall be of length maximum 28 octets. A SDU need not be segmented if its length is smaller than 35 (28 respectively) octets.

Using the SDU segments a DF PDU shall be assembled starting with the first segment. A PDU, as shown on the figure below, comprises a header, a payload part (a SDU segment), a 16-bits CRC and possibly a number of don't care octets resulting in a total of 40 octets (32 octets in the case of the Layer 2 protected mode supported). The CRC shall be placed immediately after the SDU segment.

The structure of a DF PDU is depicted on the figures 11 and 12 and the settings for the fields are provided afterwards:

Bit:	8	7	6	5	4	3	2	1	Octet
	M	Send Sequence nr. (SN)							1
	A/N	Receive Sequence nr. (RN)							2
	Length indicator (LI) field								3
	SDU part								4
	…								
	SDU part								LI + 2
	CRC								LI + 3
	CRC								LI + 4
	don't care bits								LI + 5
	...								
	don't care bits								40

Figure 11: Frame format type DFFU10 for layer 2 non-protected

Bit:	8	7	6	5	4	3	2	1	Octet
	M	Send Sequence nr. (SN)							1
	A/N	Receive Sequence nr. (RN)							2
	Length indicator (LI) field								3
	SDU part								4
	…								
	SDU part								LI + 2
	CRC								LI + 3
	CRC								LI + 4
	don't care bits								LI + 5
	...								
	don't care bits								32

Figure 12: Frame format type DFFU10 for layer 2 protected

Each PDU shall be assigned a SN number. The SN shall be set to 0 on the establishment of the connection and sequencing shall be maintained throughout the existence of the call. The RNs provide both window control to avoid possible sequencing errors, and also invoke automatic PDU retransmission. For all operations Modulo 128 shall be used, i.e. SN = [0 ... 127], RN = [0 ... 127].

The PDUs shall be submitted to the DLC layer in the order of ascending SN, taking into account the modulus operation of the sequence numbering.

The sending entity shall maintain a maximum window size between the SN and the last received RN. The default value of this window size shall be 4.

Whenever the window size limit is reached no new PDUs will be inserted in the window and the sending side shall commence retransmission of all outstanding PDUs not already expired, starting from the oldest unacknowledged PDU. This automatic retransmission shall be stopped whenever a usable RN is received (i.e. a RN that acknowledges one or more outstanding PDUs), and normal transmission or retransmission procedures will be resumed.

Received RN with the A/N bit set to "1" shall be treated as a positive acknowledgement for all PDUs up to and including the PDU number RN. This positive acknowledgement shall cause an immediate stop to any redundant (unnecessary) retransmissions that may have been scheduled as a result of previously received negative acknowledgements.

Received RNs with the A/N bit set to "0" shall be treated as a negative acknowledgement for the single PDU number RN. Receipt of a NACK shall cause a retransmission of the indicated PDU(s).

The LI (0<LI) means that the length indicator byte is followed by a segment of payload of the length indicated by the length indicator.

The role of the "M" bit is to indicate if a PDU carries part from a SDU or a complete SDU. The following rules for setting/understanding the setting of the M bit shall apply:

M = "1" and a value of the length of Information field equal to the remaining number of octets till the end of the PDU indicates that the information field only contains part of a SDU. There is more to follow in the next PDU.

M = "0" and a value of the length of Information field different from zero, indicates one of two things:

1. that the information field contains a complete SDU, provided that the M bit of the previous length indicator was also set to "0";

2. that the information field contains the last segment of a SDU, provided that the M bit of the previous length indicator was set to "1".

For the CRC, the ARQ checksum defined in clause 11.9.4.2.3 of EN 300 175-4 [4] shall be used with the difference that the constant "k" is a variable and depends on the length of the SDU segment included into the PDU. The maximum value k = (LI - 2) will be used if the CRC occupies the last two octets of the PDU.

NOTE 1: This CRC is the only U-plane error protection that is provided by the ODAP protocol when no Layer 2 protected mode is being supported. Implementations may decide to add additional protection in the higher layers.

The PDUs shall be submitted to the DLC layer on regular intervals of 10 ms (one DECT frame) one at a time. If there is no valid data for transmission the DF shall provide a void PDU (an "empty" frame) with LI = 0.

The LI = 0 indicates that this is an "empty" PDU frame. Empty frames shall be sent only when:

1) There are no pending PDUs for transmission or re-transmission.

2) In the case if window synchronization is required due to the expiry of the life time of a not-successfully-sent SDU.

NOTE 2: The purpose of the synchronization message is to allow shift in the transmission window if a SDU life time expires. The life time of a SDU depends on the application. Some applications may disallow SDU expiration, whereas others may set a suitable time. SDU life time is be set by NWK layer exchange messages using the <<transit delay>> information element, the default value is "infinite TDMA frames".

In case (1) the Empty frame shall be treated as a normal PDU with the exception that it shall have LI = 0 and M = 0; that is, the SN, RN and A/N shall be set according to the rules that apply for not empty frames.

In case (2) the Synchronization Empty frame shall have its SN set to the SN of the PDU with highest SN (modulo rules apply) that shall be discarded. At the receiving side the frame shall be recognized to be a synchronization, and not a "normal" empty frame based on the verification if there are stored PDUs that cannot be re-assembled; that is, the transmitter can send a "normal" empty frame only when all transmitted PDUs from a SDU have been successfully ACKnowledged.

As soon as a SDU is delivered from the upper layer to the DLC, a timer equal to the agreed SDU life time shall be associated with it. Whenever a timer reaches its limit, the respective SDU shall be considered expired and not (re)transmitted anymore.

The transmitting window shall not shift due to the expiry of those PDUs belonging to the expired SDU. When one or more PDUs expire, then the last expired PDU shall be replaced by the synchronization message. The synchronization message shall contain the sequence number of this PDU. When the transmitter receives the acknowledgement of the successful reception of the synchronization message, it knows the receiver has re-synchronized its window and the transmitter is also allowed to move the transmit window forward, accepting new PDUs for transmission.

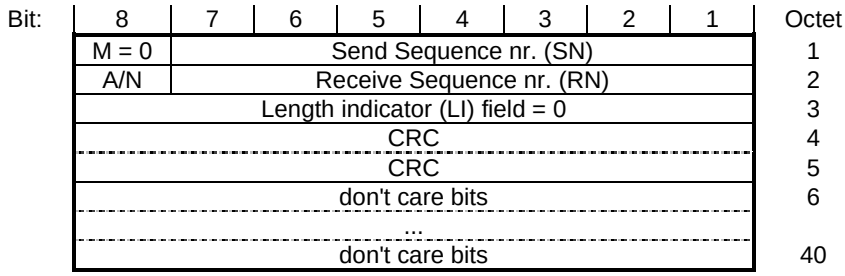


Figure 13: Synchronization frame format type DFFU10

For data flow control between the DF and the DLC a procedure based on the procedure defined in EN 300 175 (parts 3 and 4) between the MAC and the DLC layers for FBn function (as in GAP, EN 300 444 [10]) may be implemented when DF will submit data on request from the DLC, this is however internal implementation dependant issue.

NOTE 3: The usage of primitives in DECT is not mandated and they are specified for descriptive purposes.

Transmit side: on receipt of a DF_CO_DTR-ind primitive, one complete frame of data is submitted to the DLC layer in a DF_CO_DATA-req primitive.

Receive side: each DF_CO_DATA-ind primitive contains one complete frame of data from the DLC layer.

NOTE 4: The primitives above should be similar as the primitives in the EN 300 175-3 [3] (only the names are different).

At the receiving entity data packets shall be accepted from the DLC in the order of their delivery. The packets shall be assumed to contain valid PDUs, and shall be processed in their order of arrival. The DLC shall submit the packets to the DF which shall check the CRC. If it proves to be correct the PDU sequence number shall be examined. PDUs with sequence number outside the receive window shall be discarded. Only a synchronization message shall be accepted always, even if the sequence number is outside the receive window. Duplicate PDUs shall also be discarded.

In-sequence PDUs are defined as a series of one or more PDUs that contain no errors and that contain SN(s) that together form a continuous series of SNs when considered together with other received but undelivered PDUs. All correct PDUs shall be acknowledged by including an appropriate RN in a PDU sent back to the transmitter. If there is no pending PDU to be sent to the transmitter one shall be constructed with LI = 0. It is not required (but recommended) to send ACK immediately after a PDU is received.

Synchronization messages shall always be acknowledged as soon as possible. For the acknowledgment the same format as the synchronization message shall be used (see earlier) with ACK bit set accordingly.

When a SDU has been determined to be completed the DF shall submit it to the higher layer.

11 ODAP media transfer

This clause specifies the protocol requirements in regard to exchange of ODAP media information.

11.1 General

The present document defines the following transfer services:

- Hypermedia Transfer – The hypermedia transfer services provides for the transfer of self-describing hypermedia resources (including the transfer of asynchronous multimedia messages such as email or instant messages). It is based on the HTTP (Hypertext Transfer Protocol) [14].

- Streaming – The streaming services that provide a means for transferring isochronous data such as audio and video are for further studies.

The ODAP transfer services specify only the transfer between a DECT PT and a DECT FT. That is, the present document limits the application of the ODAP - HTTP to communication between two DECT terminals over the DECT air interface and does not cover communication outside of this pair, i.e. if for example the FT is to communicate to a remote server which requires HTTP communication, the ODAP Hypermedia Transfer Service does not guarantee the provision of a transparent for the FT HTTP communication between the PT and the remote server. For such a communication it is the responsibility of the FT to establish a separate HTTP communication to the remote server; a fully compliant with RFC 2616 [14] protocol needs to be implemented in the FT for such an application.

11.2 Hyper media transfer basic requirements

HTTP is specified in RFC 2616 [14]. In the following, elements, described in RFC 2616 [14], have been taken and declared as mandatory or optional for ODAP and wherever necessary clarifications, modifications or new requirements are provided.

The ODAP HTTP implementation provides both the "pull" and "push" data transfer models. Pull is achieved by a terminal (PT or FT) when acting as a client; Push is achieved by switching the role of the same terminal to a server (both ways using the request/response mechanism from HTTP/1.1).

The basic reference model relevant for the present document is depicted figure 14:

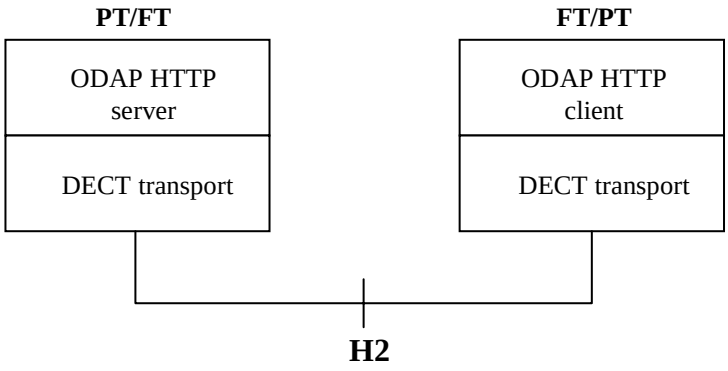
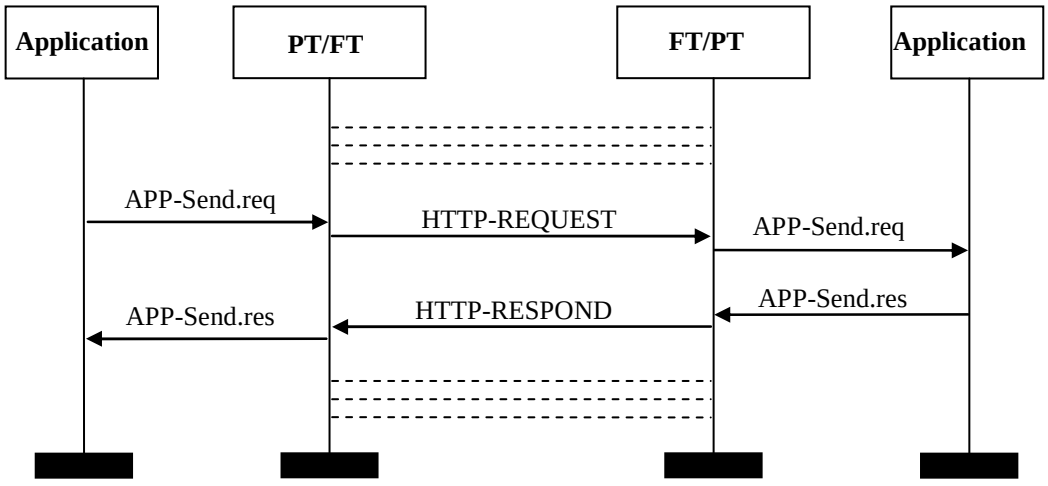


Figure 14: ODAP HTTP protocol reference model

ODAP HTTP shall support persistent connections and pipelining as specified in RFC 2616 [14] with the modification that the term "TCP connection" shall be replaced by a DECT connection.

The basic model of interaction between the PT and FT is a HTTP request/response. Requests and responses are triggered by the Application and in the normal case will carry Application PDUs. An example is provided on the figure 15:



NOTE: APP-Send.req and APP-Send.res are defined in clause ODAP application communication protocol in the present document.

Figure 15: Media transfer example

The HTTP request/response messages as defined in RFC 2616 [14] consist of a start-line, zero or more header fields, an empty line indicating the end of the header fields, and possibly a message-body. The specific requirements in regard to the ODAP media transfer utilization of HTTP are described in the following clauses.

11.2.1 Messages specification

11.2.1.1 Message start-line requirements

11.2.1.1.1 Request start-line

The first line in a request is called a Request-Line: The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF.

Mandatory for support methods: GET and POST.

The HTTP version shall be "HTTP/1.1". As ODAP HTTP usage is ODAP specific and is not to be used outside of the communication between 2 DECT terminals over the DECT air interface change in the HTTP version is not envisaged.

The HTTP Request-URI is a Uniform Resource Identifier and identifies the resource upon which to apply the request. ODAP terminals shall set this field to "*", which shall be understood as that the request/response applies to the terminal itself (i.e. FT or PT that receives the message).

NOTE: ODAP addressing is implemented by DECT C-plane means.

An example of HTTP Request start line is:

GET * HTTP/1.1

11.2.1.1.2 Response start-line

The first line in a Response is called a Status-Line: The Status-Line consists of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by SP characters.

The HTTP version shall be "HTTP/1.1".

The status codes mandatory for support: "200" - OK, "400" - Bad Request, "405" Method Not Allowed, "415" - Unsupported Media Type, "500" - Internal Server Error, "501" - Not Implemented. The support of other codes is optional.

An example of HTTP Request start line is:

HTTP/1.1 200 OK

11.2.1.2 Message header requirements

RFC 2616 [14]defines the HTTP message headers as: HTTP header fields, which include general-header, request-header, response-header, and entity-header fields. The support of the HTPP message headers is optional with the exceptions/clarification defined in this clause.

Equipment claiming compliance to the present document shall support the following headers as specified in RFC 2616 [14]:

- Content-Length (signalling the presence of a message-body);
- Content-MD5 (providing an end-to-end message integrity check of the entity-body for the purpose of detecting accidental modification of the entity-body in transit).

NOTE: The usage of the Content-MD5 provides a second level of "error protection" above the ODAP DF CRC.

ODAP equipment is not required to be able to apply encoding to the HTTP message or the message body.

ODAP equipment is not required to support any other media types or character sets than those specified in the clause application communication protocol in the present document; the default media type is "text/xml".

The RFC 2616 [14] mandates that if accept header is not included all media types are acceptable in the response. ODAP equipment is not required to send nor to understand the accept header; OADP equipment is not allowed to send other media types than "text/xml" as defined in the clause application communication protocol in the present document.

ODAP equipment is not required to send nor to understand the Accept-Character header. The character set to be used by ODAP equipment is defined in the clause application communication protocol in the present document.

ODAP equipment is not required to send nor to understand the Accept-Encoding header. ODAP equipment is not required to be able to apply encoding to the HTTP message.

ODAP equipment is not required to send nor to understand the Allow entity-header field. The omission of this header shall be understood as indication that only GET and POST methods are supported.

11.2.1.3 Message-body requirements

RFC 2616 [14] defines the message-body (if any) of an HTTP message as being used to carry the entity-body associated with the request or response. For the purpose of the present document, the message body, if included, shall contain one of the ODAP application PDUs as defined in the present document.

11.2.2 Procedures specification

The ODAP Hyper media transfer service shall be used as a transport mechanism for the exchange of the ODAP application elements of procedures. The usage of the GET and POST methods by the application procedures is described in this clause; the application procedures are described in clause 12.

The ODAP Client Application that wishes to send a message shall invoke an ODAP/HTTP POST operation with the relevant Application PDU message embedded as the content body. POST shall be used for all application procedures with the exception of the fetch message procedure (see clause 12). Upon receipt of the application message the server (FT or PT) shall submit it to the destination application. Depending on the implementation the serving terminal may:

a) Immediately, without awaiting respond from the application, send a response to the POST which shall not include entity body (a consecutive response from the application shall be treated as a new message to be sent).

b) Await the respond from the application. Upon the reception of the message the serving entity shall respond to the POST operation sending a HTTP response with the application message embedded as the content body.

NOTE 1: Case a) option aims at allowing terminals to release a call and go in lower duty cycle mode to preserve battery life when awaiting response from a remote application. Terminals should avoid entering in a long sleep mode when waiting response from the application.

NOTE 2: The relation between applications requests and responds as part of one and the same application procedure or relations between procedures, e.g. a delivery report relation to a send message, are handled by the application message-ID as specified in clause 12.

The ODAP Client Application that wishes to retrieve a message shall invoke an ODAP/HTTP GET operation with the Application PDU message embedded as the content body and indicating the message-ID for the message to be retrieved. The server (FT or PT) shall deliver the relevant message in the ODAP/HTTP response to the GET.

11.3 ODAP Streaming media protocol basic requirements

FFS.

12 ODAP application communication protocol

This clause describes the procedures and the format of the PDUs exchanged to ensure interoperability of various ODAP applications. Further requirements are provided in annex A.

This version of ODAP assumes a client/server model in which:

- The client resides or is connected to the PT using AT commands for its communication with the PT. It is possible that a number of Client Applications are connected to a single PT.
- The server resides or is connected to the FT (a remote server) using means and non DECT protocols which are out of the scope of the present document.

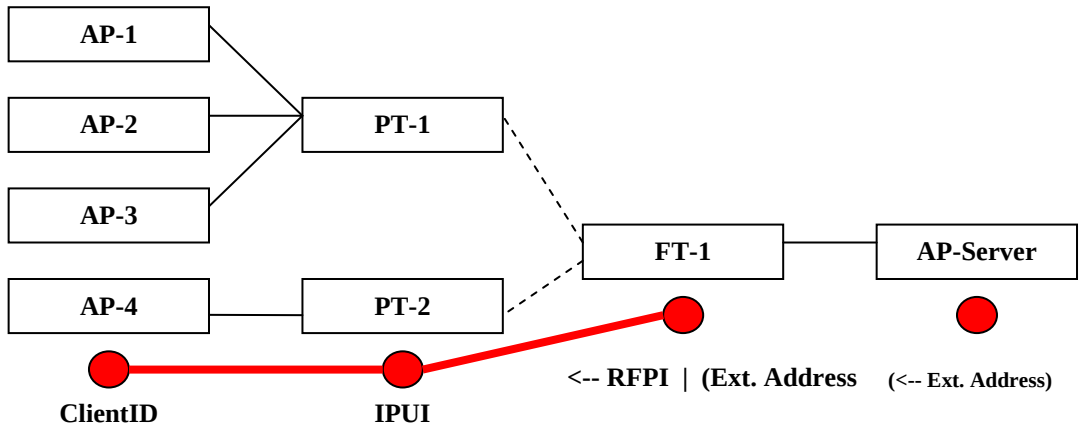
In the case of a remote server, FT shall play a role similar to a proxy server. This makes 3 basic ODAP FP product categories: ODAP FP server inside (See clause 6.2.2 for an example), ODAP FP remote server LAN (FT has all relevant protocols to be able to exchange messages with the server, i.e. ODAP PDUs on a LAN, e.g. all IP transport relevant protocols - see clause 6.2.1 for an example), ODAP FP remote server dial-up (FT has all relevant protocols to be able to exchange messages with the server, i.e. ODAP PDUs, on a dial-up connection - see clause 6.2.1 for an example).

ODAP targets simple PT/Client Application implementations. Knowledge at the PT side as what for a FT category the PT will be connected to will complicate the situation and may lead to interoperability problems. Consequently all ODAP PTs shall assume that the FT category is "ODAP FP server inside". It is the FT responsibility, if FT falls in another category, to properly handle the ODAP PDUs.

A direct consequence of this approach is the fact that ODAP PT may not provide server's address and consequently, if the FP is not of category "ODAP FP server inside", the FT needs to know the server address itself.

In any case, the handling of the application PDUs and the possible actions that may result from the service data they carry, is the responsibility of the server.

The reference model of such a configuration is depicted on the figure below. In colour, in the lower part of the figure, are indicated the nodes and paths in the scope of the present document as well as the addressing information needed for the identification of the nodes and the paths.



NOTE: Other configurations, e.g. similar to ad hoc networking may be defined in other versions of the present document or elsewhere.

Figure 16: Client/Server reference configuration

The application procedures and their support status are described in clause 12.1.1. The messages used by the procedures and their content are described in clause 12.1.2. Some application specific procedures and messages are defined in annex A as well.

Which of these procedures are used by an implementation depends on the application needs - specific application requirements are provided in clause 12.2. Further editions of the present document may cover new application and provide additional requirements.

12.1 Application elements of procedures

12.1.1 Application procedures

The table 15 indicates the required support status for the procedures currently defined:

Table 15: Application procedures support status

Procedure	Status	
	Client	Server
Client initiated register	M	M
Client initiated de-register	M	M
Server initiated de-register	O	O
Client initiated submit	M	M
Server Initiated submit	O	M
Client configuration	O	M
Client alive indication	M	M
Client/server sleeping mode management	C1	M
C1: IF (Server initiated de-register = M) OR (Server initiated submit = M) OR (Client configuration = M) THEN M, ELSE I.		

12.1.1.1 Register

The register procedure provides means that a Client Application can register with the server. Multiple clients can register with the same server. Furthermore the register procedure provides means for client operation configuration

It is assumed that the PT that handles the client has DECT registered to the FT using the normal DECT subscription procedure. PTs that handle multiple Client Applications can register only one application at a time and shall not start a new registration procedure before the completion of a registration procedure in progress.

To register to the server a client shall send an {APP-REGISTER-REQ} PDU. The {APP-REGISTER-REQ} shall always include a <TI> to identify the transaction and allow the request message to be linked with the response message. Optionally a <Date> header may be included indicating the time and date the PDU was sent.

The PDU may include a <To> header field indicating the address of the registrar (the server). The present document does not require the clients to know anything about addressing.

The PDU shall include a message body with the service data identifying the type of application, e.g. a temperature sensor, a fire alarm, a door bell, etc., and <ApplicationCapability> field listing the application procedures supported by the client (application procedures are defined in annex A in the present document). Additionally, if a client has parameters that need configuration, the client shall list all such parameters and their initial values; if a client can report on a number of non-configurable parameters/events, the client shall list all such parameters/events; if the client can report on one parameter/event only, the client may indicate the parameter/event.

If the client supports the ODAP sleeping mode of operation and/or the alive indication it shall include empty <TSleep>, <TAwake> and/or <TAlive> fields.

The PDU shall be delivered to the Hyper media transfer function in the PT which is responsible for the transfer of the PDU to the FT. The PT shall start T_app_reg.

Upon receiving the PDU the FT shall deliver it to the server. The means for transporting the PDU to the server and any possibly related addressing issues are out of the scope of the present document.

The server shall respond with an {APP-REGISTER-RES} PDU indicating the acceptance or the rejection of the request in the <Status> header. Depending on the content of the service data provided by the client, before sending the message the server may engage the user in client operation configuration or may need to retrieve configuration information from a storage. Configuration data if provided shall be included in the message body as service data.

If the Status="rej" a Reject Reason Code shall be provided in the <RCode> header and optionally a Reject Reason Text in a <RText> header.

If the server accepts the registration, i.e. Status="ack", it shall allocate an <ClientID> to the client. The {APP-REGISTER-RES} PDU, and any other PDU exchanged between the server and the client from this point on, shall include the allocated client ID to identify communication between the server and this particular client. When a PDU is received by the server, the <ClientID> may be used by the server to forward all messages coming from that particular application to a particular destination.

NOTE 1: Servers should avoid re-using client ID that has been freed by a de-registered client. The client ID field specification provides a great variety of formats and a large number of possible client ID values, see clause 12.1.2.5.1.

The {APP-REGISTER-RES} PDU may optionally include a <Date> header indicating the time and date the PDU was sent.

The {APP-REGISTER-RES} PDU may include a message body with service data configuring the client operation in the following cases:

1) If the client has provided Parameters with parameters' values, i.e. configurable parameters, the respond my provide parameters' values and Update rate (i.e. how often report should be provided) if relevant. The server may chose to postpone the configuration for a later stage in which case a combination of GetParameters, SetParameters and TraceStart application procedures as defined in annex A shall be performed. In this case the client shall not start operation before the configuration is completed and the server has explicitly requested the TraceStart.

2) If the client has provided parameters without parameters' values, i.e. non-configurable parameters, and, if the client is expected to report on a regular basis, the respond shall provide Update Rate (i.e. how often report should be provided) per parameter or a single Update Rate valid for all parameters. Change of the rate may later be accomplished through invocation of a TraceStart Application procedure as defined in the relevant Application. If TraceStart is not supported the operation can only be achieved through a new registration procedure. In this case the client can start monitoring (trace) immediately after the successful completion of the client registration procedure.

3) If the client is not expected to report on a regular basis, e.g. a fire alarm that reports only if there is a fire, the respond should not provide Update Rate but still may if the user wishes so. If Update Rate is provided the client shall obey the request. In this case the client can start monitoring (trace) immediately after the successful completion of the client registration procedure.

4) If the client has indicated support by including the <TAlive> field in the request message and if the Server/User would like to receive on a regular bases information from the client if the client is still operational, a <TAlive> field shall be included specifying the intervals at which Client alive indication procedure shall be performed (for further details see clause 12.1.1.5).

5) If the client has indicated support of server initiated procedures and the server, depending on the application requirements, will be using sever initiated procedures with the particular client, i.e. would require the PT handling the client to listen for incoming calls, the server shall configure the client/server sleeping mode by including a <TSleep> and a <TAwake> fields providing values for the relevant to the mode timers (for further details see clause 12.1.1.6)

If, by any reasons, the server is unable to complete the configuration parameters setting within the time of the T_app_reg, e.g. user is too slow in providing them, the server should REJ the registration with <Reject Reason Code> = "320".

NOTE 2: It is assumed that if configuration would be required relevant information is provided/chosen by the user to the server during the registration or is set in advance. The means for provision of such information is implementation dependent and left to the designer.

The PDU shall be delivered to the Hyper Media Transfer function in the FT which is responsible for the transfer of the PDU to the PT. Upon receipt of the {APP-REGISTER-RES} the FT shall take notice of the <ClientID> and shall use it for further PDU routing, e.g. in case of incoming calls being supported.

Upon receipt of the {APP-REGISTER-RES} the PT shall take notice of the <ClientID> and shall use it for further PDU routing, e.g if a PT supports multiple clients per call when a PDU is received by the PT from a server, the <ClientID> shall be used to distinguish the client addressed.

Upon receipt of the {APP-REGISTER-RES} the Client Application shall take notice of the <ClientID> and shall use it for every future communication until the registration is cancelled.

At the procedure initiating side, if timer T_app_reg expires prior to the reception of an {APP-REGISTER-RES}, the procedure may be repeated. If a REJ was received with <Reject Reason Code>=320 the procedure shall be repeated.

12.1.1.2 Submit

12.1.1.2.1 Client initiated submit (PT to FT)

If a Client Application residing in, or connected to, a PT has information to be sent to the server it shall initiate a Client initiated application submit procedure. The Client initiated application submit procedure is a procedure based on a handshake between the sending and the receiving applications.

When information is to be sent to the peer application, the sending application shall construct an {APP-SUBMIT-REQ} PDU. The {APP-SUBMIT-REQ} shall always contain a Transaction Identifier (TI) to identify the transaction and allow multiple simultaneous transactions to take place, and an <ClientID> header to identify the client. The information (service data) submitted shall be included into the PDU's message body.

The PDU may include a <To> header field indicating the address of the registrar (the server). The present document does not require the clients to know anything about addressing.

The PDU shall be delivered to the Hyper Media Transfer function in the PT which is responsible for the transfer of the PDU to the FT. The PT shall start T_app_submit.

Upon receiving the PDU the FT shall deliver it to the server. The means for transporting the PDU to the server and any possibly related addressing issues are out of the scope of the present document.

The server shall respond with an {APP-SUBMIT-RES} PDU indicating the acceptance or the rejection of the request in the <Status> header. The PDU shall include the <ClientID> for routing purposes. If a message body is to be included depends on the application and the service data send in the {APP-SUBMIT-REQ} (see annex A for further details on specific application procedures that may require message body to be included).

If the Status="rej" a Reject Reason Code shall be provided in the <RCode> header and optionally a Reject Reason Text in a <RText> header.

The PDU shall be delivered to the Hyper Media Transfer function in the FT which is responsible for the transfer of the PDU to the PT.

At the procedure initiating side, if timer T_app_submit expires prior to the reception of an {APP- SUBMIT-RES} message, the procedure may be repeated. Alternatively, if the application supports incoming calls, it may decide to release the call and await the response in idle lock state. In this case the responding side, as soon as the response is ready for transmission, shall initiate a new DECT call and transmit the response.

If an {APP-SUBMIT-REQ} is re-sent before the reception of the {APP-SUBMIT-RES} the <TI> of the original request shall be used - this allows for example delayed responses to the original request to be handled.

If a "REJ" message is received, the initiator may try to resubmit the initial {APP-SUBMIT-REQ} or construct a modified one and initiate the procedure again.

NOTE: It is possible in rare occasions that REJ is done due to error introduced in the REQ message during the transmission and which was no caught by the DF error correction scheme.

12.1.1.2.2 Server initiated submit (FT to PT)

The Server initiated submit procedure is similar to the client initiated procedure and shall be implemented as defined in clause 12.1.1.2.1 with the following modifications:

- The roles of the client and server shall be exchanged.
- FT shall base its decision about which PT shall be paged on the <ClientID> carried by the {APP-REGISTER-REQ} and locally stored information that links the <ClientID> to the relevant IPUI (and TPUI). Such mapping table should be set up during the client registration phase.

12.1.1.3 De-register

12.1.1.3.1 Client initiated de-register (PT to FT)

The de-register procedure provides means that a Client Application can de-register itself from the server.

To de-register from the server a client shall send an {APP-DEREGISTER-REQ} PDU. The {APP-DEREGISTER-REQ} shall always include a <TI> to identify the transaction and allow the request message to be linked with the response message, and an <ClientID> header to identify the client. Optionally a <Date> header may be included indicating the time and date the PDU was sent.

The PDU may include a <To> header field indicating the address of the registrar (the server). The present document does not require the clients to know anything about addressing.

The PDU shall be delivered to the Hyper media transfer function in the PT which is responsible for the transfer of the PDU to the FT. The PT shall start T_app_dereg.

Upon receiving the PDU the FT shall deliver it to the server. The means for transporting the PDU to the server and any possibly related addressing issues are out of the scope of the present document.

The server shall respond with an {APP-DEREGISTER-RES} PDU indicating the acceptance or the rejection of the request in the <Status> header. The PDU shall include the <ClientID> for routing purposes.

If the Status="rej" a Reject reason code shall be provided in the <RCode> header and optionally a Reject reason text in a <RText> header.

If the server accepts the registration, i.e. Status="ack", it shall remove any association made to that particular Client Application and free the <ClientID> allocated to the client for use with other clients.

The {APP-DEREGISTER-RES} PDU may optionally include a <Date> header indicating the time and date the PDU was sent.

The PDU shall be delivered to the Hyper Media Transfer function in the FT which is responsible for the transfer of the PDU to the PT.

At the procedure initiating side, if timer T_app_reg expires prior to the reception of an {APP-DEREGISTER-RES}, the procedure may be repeated. Clients shall not de-register themselves without instruction from the server.

12.1.1.3.2 Server initiated de-register (FT to PT)

The server initiated de-register procedure is similar to the client initiated procedure and shall be implemented as defined in clause 12.1.1.3.1 with the following modifications:

- The roles of the client and server shall be exchanged.
- FT shall base its decision about which PT shall be paged on the <ClientID> carried by the {APP-DEREGISTER-REQ} and locally stored information that links the <ClientID> to the relevant IPUI (and TPUI). Such mapping table should be set up during the client registration phase.
- Server may de-register a client locally if the server initiated de-register procedure fails. If subsequently the server receives information from a de-registered client it shall reject the transaction with Reject Reason set to 404 - Client-ID not known. To avoid being contacted by a de-registered client which has the same client ID as another client, registered after the de-registration, the server should not re-use de-registered client IDs.

NOTE: The client ID field specification provides a great variety of formats and a large number of possible client ID values, see clause 12.1.2.5.1.

12.1.1.4 Client configuration

Client configuration shall be performed during registration phase. Some implementations may provide client configuration after the registration has taken place. Configuration after registration phase may use the submit procedure exchanging specific Client. Application specific procedures for such purposes are defined in the annex A.

12.1.1.5 Client alive indication

A server/user may request the client automatically to indicate if it is still in operation.

NOTE: Alternatively a server/client may decide to do this manually, which is out of the scope of the present document.

If a server/user wishes the client automatically to indicate if it is still in operation, it shall assign to the client a T_alive value. The value shall be assigned during client registration procedure and may be inquired and modified after the registration procedure depending on the provision and support of relevant specific application procedures (see annex A). The value shall be a decimal number representing a number of units based on multiframes as specified in table 16.

Table 16: T_Alive timer value

Timer	Value	Comment
T_alive	A Decimal number of units	1 unit = 2^8 multiframes (~41 sec)

The client shall start the timer immediately after it receives the assigned value. It is recommended that the start is aligned with the multiframe number that is transmitted by the FT as specified in the EN 300 175-3 [3]. The server should base its expectations about the reception of the alive indication on some time tolerance due to possible differences between the time the server assigns the value, the client receives the value and starts the timer.

After the T-alive expires the client shall initiate a client initiated submit procedure as specified in clause 12.1.1.2.1. The Client shall include an <Action> field. Optionally the client may include some status information.

Table 17: Alive indication PDU format

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> AliveIndication </Action>	M
		<Status> OK|NotOK </Status>	O
		<Battery> > OK|NotOK </Status>	O
		<StatusString> Human readable text </StatusString>	O
	</Body >		M
</AppSubReq >			M

Before re-starting the T_alive the client shall re-synchronize it with the multiframe number if necessary.

12.1.1.6 Client/server sleeping mode management

If a client supports incoming calls, e.g. it supports server initiated procedures, it shall implement the requirements specified in this clause. All servers shall support client/server sleeping mode management and shall enforce it if a client supports it too. Clause 9.3 specifies the PT/FT behaviour in regard to the sleeping mode of operation. This clause defines the client/server behaviour.

Clients shall indicate if they support the procedure and servers, if the client supports the procedure, shall assign values to T_sleep and T_awake timers during client registration procedure (see clause 12.1.1.1). Servers may inquire and modify the assigned values after the registration procedure depending on the provision and support of relevant specific application procedures (see annex A). The value shall be a decimal number representing a number of units based on multiframes as specified in the table 18.

Table 18: Sleeping mode timers values

Timer	Value	Comment
T_sleep	A Decimal number of units	1 unit = 2^8 multiframes (~41 sec)
T_awake	A Decimal number of units	1 unit = 2^8 multiframes (~41 sec)

The client shall start timer T_sleep immediately after it receives the assigned value. It is recommended that the start is aligned with the multiframe number that is transmitted by the FT as specified in the EN 300 175-3 [3]. The server should base its expectations about the time the client would be accessible for incoming calls on some time tolerance due to possible differences between the time the server assigns the value, the client receives the value and starts the T_sleep; furthermore, the server shall envisage some time needed by the PT to find the FT and log on. The client may, depending on the implementation, instruct the PT to power down or enter an implementation defined low power consumption mode.

After the T_sleep expires the client shall instruct the PT to go to Idle Locked state which may include power-up if the PT was powered down. The client shall start T_awake.

After the T_awake expires the client may, depending on the implementation, instruct the PT to power down or enter an implementation defined low power consumption mode. The client shall start T_sleep.

When PT is in listening (awake) mode the client shall re-synchronize its timers with the multiframe number if necessary.

12.1.1.7 Timers

The following requirements are mandatory for the support of the timers defined in this clause.

<T_app_reg> Registration timer.

Server value: None.

Client value: 120 seconds.

Start: An {APP-REGISTER-REQUEST} message is sent.

Stop: An {APP-REGISTER-RESPOND} message is received.

<T_app_dereg> De-Registration timer.

Server value: 30 seconds.

Client value: 60 seconds.

Start: An {APP-DEREGISTER-REQUEST} message is sent.

Stop: An {APP-DEREGISTER-RESPOND} message is received.

<T_app_submit> Message submit timer.

Server value: 30 seconds.

Client value: 30 seconds.

Start: An {APP-SUBMIT-REQUEST} message is sent.

Stop: An {APP-SUBMIT-RESPOND} message is received.

12.1.2 Application PDUs

The ODAP application PDUs are XML documents.

The element names used within the PDU are written using only the characters 0x20 - 0x7F in US-ASCII (ISO 646[17]). Maximum length of an element name is 25 characters. Numerical digit is not used as first character (reserved for future usage).

The data content between a start and an end tag must be escaped according to XML standard.

EXAMPLE 1: The "&" character equals to "&".

All characters within a message block must use ISO 8859-1 encoding [18]. This is not compatible with all known languages as UTF-8, but much easier to represent. The encoding must be specified on the first line of the XML-block in the following way:

<?xml version='1.0' encoding='ISO-8859-1'?>

It is up to the receiving application to decide if the received and decoded characters are supported and what to do with unknown characters.

The XML element consists of a start tag and end tag. All tag names start with an uppercase letter. If an element contains no data the start and end tag may be within a single tag.

EXAMPLE 2: "<Test></Test>" is equivalent to "<Test/>".

A PDU block is an XML document consisting of two parts, header and body. The header contains address and other message related information and the body contains application specific service data. Whether service data is included into the body depends on the PDU type and application requirements.

Service data encryption is not used in this version of the standard. This does not preclude implementations from one and the same manufacturer to use encryption that the application peers understand.

The general format for the application PDU is provided in table 19 and each element is described in the following clauses and annexA.

Table 19: General ODAP PDU format

PDU content	Status
<?xml version="1.0" encoding="ISO-8859-1"?>	M
<ODAP-PDU-type version="X.Y" TI="xxxxx">	M
<Header>	M
INFORMATION	M
</Header>	M
<Body >	M
SERVICE DATA	C1
</Body>	M
</ODAP-PDU-type>	M
C1: IF Procedure requires it THEN M, OTHERWISE I	

12.1.2.1 ODAP PDU types

Table 20: ODAP PDU types

Message type name	Message type coding	Status
APP-REGISTER-REQUEST	AppRegReq	Mandatory
APP-REGISTER-RESPONSE	AppRegRes	Mandatory
APP-SUBMIT-REQUEST	AppSubReq	Mandatory
APP-SUBMIT-RESPONSE	AppSubRes	Mandatory
APP-DEREGISTER-REQUEST	AppDeregReq	Mandatory
APP-DEREGISTER-RESPONSE	AppDeregRes	Mandatory

12.1.2.2 ODAP version identifier

The published version of the present document, without the least significant digit, as indicated on the cover page, specifies the ODAP protocol version. For example the current version of the present document specifies ODAP version 1.1 (least significant digit "1" is used for working drafts and does not represent interoperability requirements). The major number of the received message version is a compatibility number that must be equal to the client supported ODAP version. E.g., a received message with version="1.2" is handled if the server supports version "1.x".

Table 21: ODAP version identifier

ODAP Version Identifier	
Occurrence	Mandatory
Content	version="X.Y"
Type	ISO-8859-1 characters

12.1.2.3 Transaction Identifier (TI)

This identification, TI, holds a hexadecimal number 1-0xFFFFFFFF. The number is used as reference when sending a response to a received service (message request) and allows multiple transactions to take place simultaneously or in sequence during a call. The value shall be initiated to 0 at the beginning of each call and increased by one with each new transaction.

An implementation should ensure that the possibility of repetition of one and the same number within short period of time is minimized.

Table 22: Transaction Identifier (TI)

Transaction Identifier (TI)	
Occurrence	Mandatory
Content	1-8 characters. Ascii representation of the hexadecimal value of the id.
Type	ISO-8859-1 characters

12.1.2.4 PDUs content description

The specified in this clause PDU content does not prevent applications of using additional fields. For example an application may add other header fields to those specified here. If new fields are added the rules defined in clause 12.1.2 shall be obeyed. An application that does not support a field shall ignore it.

If the PDU and its elements status indicated in this clause differ from the status indicated into the procedures using them the status indicated into the procedures description (clause 12.1.1) has precedence.

12.1.2.4.1 APP-REGISTER-REQUEST

The {APP-REGISTER-REQUEST} PDU shall be coded as described in table 23. The coding of each element is described later on.

Table 23: {APP-REGISTER-REQUEST}

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppRegReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<Date> Sending Date and Time</Date>	O
		<To> Address of the called party </To>	O
	</Header>		M
	<Body>		M
		<ClientType> Client Type : Client Description <ClientType>	M
		<TAlive></TAlive>	C3 (Note 2)
		<TSleep></TSleep>	C3 (Note 2)
		<TAwake></TAwake>	C3 (Note 2)
		<PV> Parameter Name 1 = Parameter Value1 </PV>	C1 (Note 1)
		<PV> Parameter Name 1</PV>	C2 (Note 1)
		...	
		...	
		<PV> Parameter Name N = Parameter ValueN </PV>	C1 (Note 1)
		<PV> Parameter Name N </PV>	C2 (Note 1)
	</Body >		M
</AppRegReq >			M

C1: IF Parameter configurable THEN M, ELSE X.
C2: IF Parameter NOT configurable THEN M, ELSE X.
C3: IF relevant procedures supported THEN M, ELSE X (Note2).
NOTE 1: Clients that need not to report on regular bases, e.g. an alarm that will report only if certain conditions are met, need not including these fields.
NOTE 2: Fields if included shall be empty, i.e. no values shall be included.

12.1.2.4.2 APP-REGISTER-RESPONSE

The {APP-REGISTER-RESPONSE} PDU shall be coded as described in table 24. The coding of each element is described later on.

Table 24: {APP-REGISTER-RESPONSE}

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppRegRes version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
		<Status> Status </Status>	M
		<RCode> Reject Reason Code </RCode>	M
		<RText> Reject Reason Text </RText>	O
		<Date> Sending Date and Time</Date>	O
	</Header>		M
	<Body>		M
		<TAlive>T_alive value</TAlive>	O
		<TSleep>T_sleep value</TSleep>	C5
		<TAwake>T_awake value</TAwake>	C5
		<PV> Parameter Name 1 = Parameter Value1 </PV>	C1
		<PV> Parameter Name 1</PV>	C2
		<Rate> Parameter Name 1 Rate>	C3 (Note 1)
		...	
		...	
		<PV> Parameter Name N = Parameter ValueN </PV>	C1
		<PV> Parameter Name N </PV>	C2
		<Rate> Parameter Name N Rate>	C4
	</Body >		M
</AppRegRes>			M

C1: IF Parameter configurable THEN M, ELSE X.
C2: IF Parameter NOT configurable THEN M, ELSE X.
C3: IF Client needs to report on regular bases THEN M, else O.
C4: IF Client needs to report the value of the "Parameter N" on regular bases THEN M, else O.
C5: IF Client initiated procedure SUPPORTED THEN M, ELSE X.
NOTE: If parameter name was provided before the <Rate> field the Rate value, if provided, is relevant for the parameter only. If no <P> was provided the value is relevant for all parameters handled by this client - in this case no parameter or other rates are allowed.

12.1.2.4.3 APP-SUBMIT-REQUEST

The {APP-SUBMIT-REQUEST} PDU shall be coded as described in table 25. The coding of each element is described later on.

Table 25: {APP-SUBMIT-REQUEST}

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
		<Date> Sending Date and Time</Date>	O
		<To> Address of the called party </To>	O
		<ContentType>Type of the message included</ContentType>	O
	</Header>		M
	<Body>		M
		<SERVICE DATA (Message) > (See note)	M
	</Body >		M
</AppSubReq >			M
C1: IF (Server-to-Client) AND (follow-up actions supported) THEN M, ELSE IF Client-to-Server THEN X, ELSE I.			
NOTE: If Service data shall be included and its content are application specific issues and are described elsewhere in the present document.			

12.1.2.4.4 APP-SUBMIT-RESPONSE

The {APP-SUBMIT-RESPONSE} PDU shall be coded as described in table 26. The coding of each element is described later on.

Table 26: {APP-SUBMIT-RESPONSE}

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubRes version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
		<Status> Status </Status>	M
		<RCode> Reject Reason Code </RCode>	M
		<RText> Reject Reason Text </RText>	O
		<Date> Sending Date and Time</Date>	O
	</Header>		M
	<Body>		M
		<SERVICE DATA (Message) > (See note)	O
	</Body >		M
</AppSubRes >			M
NOTE: If Service data shall be included and its content are application specific issues and are described elsewhere in the present document.			

12.1.2.4.5 APP-DEREGISTER-REQUEST

The {APP-DEREGISTER-REQUEST} PDU shall be coded as described in table 27. The coding of each element is described later on.

Table 27: {APP-DEREGISTER-REQUEST}

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppDeregReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
		<Date> Sending Date and Time</Date>	O
		<To> Address of the called party </To>	O
	</Header>		M
	<Body/> (See note)		M
</AppDeregReq >			M
NOTE: An application is allowed to use the body for proprietary purposes in PDUs that does not assume message body being present. Receiving applications that do not support this feature shall ignore the whole body all together and shall not reject PDUs that contain such a body.			

12.1.2.4.6 APP-DEREGISTER-RESPONSE

The {APP-DEREGISTER-RESPONSE} PDU shall be coded as described in table 28. The coding of each element is described later on.

Table 28: {APP-DEREGISTER-RESPONSE}

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppDeregRes version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
		<Status> Status </Status>	M
		<RCode> Reject Reason Code </RCode>	M
		<RText> Reject Reason Text </RText>	O
		<Date> Sending Date and Time</Date>	O
	</Header>		M
	<Body/> (See note)		M
</AppDeregRes>			M
NOTE: An application is allowed to use the body for proprietary purposes in PDUs that does not assume message body being present. Receiving applications that do not support this feature shall ignore the whole body all together and shall not reject PDUs that contain such a body.			

12.1.2.5 PDU headers elements

The elements included into the PDUs' headers field are described in this clause.

12.1.2.5.1 Client identifier

Table 29: Client identifier

Client identifier	
Identifier	<ClientID>
Content	Format according to - A decimal digit. Ascii representation. - RFC 2822 [15]: "Internet Message Format". - RFC 2396 [20]: "Uniform Resource Identifiers (URI): Generic Syntax". - RFC 2806 [16]: "URLs for Telephone Calls."
Type	ISO-8859-1 characters.

12.1.2.5.2 Client type

Table 30: Client type

Client type	
Identifier	<ClientType>
Content	Client Type: "Alarm" or "Sensor". Client Description: User Readable text describing the client, e.g. "Fire Alarm", "Temperature Sensor".
Type	ISO-8859-1 characters.
NOTE: Future versions of the present document or profiles that are using it may define new Client types.	

12.1.2.5.3 Date

Table 31: Date

Date	
Identifier	<Date>
Content	As specified in RFC 1123 [19] and required for support by RFC 2616 [14] - HTTP (e.g. Sun, 06 Nov 1994 08:49:37 GMT).
Type	ISO-8859-1 characters

12.1.2.5.4 Reject reason code

Table 32: Reject reason code

Reject Reason Code	
Identifier	<RCode>
Content	3 decimal code corresponding to the reason for rejection (see reasons below) . Ascii representation.
Type	ISO-8859-1 characters.

12.1.2.5.5 Reject reason text

Table 33: Reject reason text

Reject Reason Text	
Identifier	<RText>
Content	1-160 characters of text (see reasons in table 24).
Type	ISO-8859-1 characters.

Table 34: Reject reason code to text mapping

Reject Reason Code	Reject Reason Text
300	Unknown error, try again
320	Congestion , try again
400	Unknown error
401	ODAP version mismatch
402	Unsupported PDU type
403	TI mismatch
404	Client-ID not known
405	PDU format corrupt
406	Service denied
407	Application not supported
408	Content not accepted
409	Address Unresolved
410	Message not found

12.1.2.5.6 Status

Table 35: Response status

Response Status	
Identifier	<Status>
Content	"ack" or "rej"
Type	ISO-8859-1 characters.

12.1.2.5.7 To

Table 36: To

To	
Identifier	<To>
Content	Characters. Format according to: RFC 2822 [15]: "Internet Message Format". RFC 2396 [20]: "Uniform Resource Identifiers (URI): Generic Syntax". RFC 2806 [16]: "URLs for Telephone Calls."
Type	ISO-8859-1 characters.

12.1.2.5.8 T_alive

Table 37: T_alive

T_alive	
Identifier	<TAlive>
Content	Decimal value as specified in clause 12.1.1.5. Ascii representation.
Type	ISO-8859-1 characters.

12.1.2.5.9 T_sleep

Table 38: T_sleep

T_sleep	
Identifier	<TSleep>
Content	Decimal value as specified in clause 12.1.1.6. Ascii representation.
Type	ISO-8859-1 characters.

12.1.2.5.10 T_awake

Table 39: T_awake

T_awake	
Identifier	<TAwake>
Content	Decimal value as specified in 12.1.1.6. Ascii representation.
Type	ISO-8859-1 characters.

12.1.2.6 PDU message body elements

This clause describes the message body field in general terms only. The client (message) to be included is specified for each application separately in a annex A or elsewhere in the present document when appropriate.

Table 40: Service data body

Service Data Body (Message)	
Identifier	<Body>
Content	Depends on application/procedure.
Type	ISO-8859-1 characters.

13 External PT communication

13.1 AT commands

PTs that provide interfaces to external devices, see interface H1 on figure 1, shall support the provision of relevant AT commands on that interface as specified in TR 102 179 [13].

Annex A (normative): Alarms and sensors applications

A.1 Principles and assumptions

This annex describes procedures and service data and its format transmitted during the procedures. All alarms and sensors implementations that claim support of ODAP shall comply with the requirements specified in this annex in addition to the requirements specified in the main part of the present document.

Application clients cannot assume any knowledge of the functionality of the whole system. Therefore they should report their status but should not suggest any corrective action. It is up to the particular server to decide on any action in respond to the status indicated by the client.

NOTE: This allows product differentiation on the FP/system side.

In many cases such an implementation may not require server initiated actions, i.e. clients would not need to handle incoming calls; in some cases however, e.g. if a sensor requires remote configuration, server initiated actions, and hence ODAP incoming calls, may be needed.

A.2 Procedures support requirements

The requirements for support of the procedures and the relevant service data specified in this annex are related to the role of the entity, i.e. a client or a server and are indicated in the table 41:

Table 41: Alarms/Sensors Procedures support Requirements

Entity	Client Initiated Procedures Support	Server Initiated Procedures Support	Incoming ODAP Calls Support
Client	M	O	C1
Server	M	M	M
C1: IF (Server Initiated Procedure Support = M) THEN M, ELSE I.			
NOTE: These requirements imply that a server shall support all procedures specified in this annex.			

A.3 Procedures specification

A.3.1 Client initiated procedures

A.3.1.1 Alarm raise

The client sends "AlarmRaise" when the sensor indicates an alarm condition.

Table 42: AlarmRaise

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> AlarmRaise </Action>	M
	</Body >		M
</AppSubReq >			M

A.3.1.2 Alarm clear

The client sends "AlarmClear" when the sensor detects that an alarm condition has cleared.

Table 43: AlarmClear

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> AlarmClear </Action>	M
	</Body >		M
</AppSubReq >			M

A.3.1.3 Battery low indication

The client sends "BatteryLow" when the battery voltage dips below a level such that a replacement is imminently required.

Table 44: BatteryLow

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> BatteryLow </Action>	M
	</Body >		M
</AppSubReq >			M

A.3.1.4 Battery OK indication

The client sends "BatteryOK" when the battery voltage returns above a safe level following a battery low condition.

Table 45: BatteryOk

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> BatteryOK </Action>	M
	</Body >		M
</AppSubReq >			M

A.3.1.5 Sensor data send

If the client supports the receipt of "TraceStart" then the client sends "SensorData" only when "TraceStart" is received and at the rate governed by "Rate" in "TraceStart". If the client does not support "TraceStart" then it may begin sending "SensorData" periodically as appropriate.

Table 46: SensorData

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> SensorData </Action>	M
		<Data> Parameter1=value1, Parameter2=value2, ..., ParameterN=valueN </Data>	M (See note)
	</Body >		M
</AppSubReq >			M
NOTE: If there is no data to send then the field is present but empty			

A.3.2 Server initiated procedures

A.3.2.1 Reset procedure

The server sends "Reset" when it wishes the client to perform a soft reset. This might be necessary for maintenance purposes for example. Upon receipt, the client shall clear all alarms that may be raised and to stop sending any sensor data.

Table 47: Reset

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> Reset </Action>	M
	</Body >		M
</AppSubReq >			M

A.3.2.2 Status request

The server sends "StatusReq" when it wishes the client to give an update of its status. Upon receipt, the client shall indicate its status by sending "StatusReqAck OK|Not OK BatteryOK|BatteryNotOK Status=STRING" where STRING is a human readable status or problem indication and | indicates the OR operator.

Table 48: StatusReq

PDU content	Status
<?xml version="1.0" encoding="ISO-8859-1"?>	M
<AppSubReq version="1.0" TI="xxxxx">	M
<Header>	M
<ClientID> Client ID </ClientID>	M
</Header>	M
<Body>	M
<Action> StatusReq </Action>	M
</Body >	M
</AppSubReq >	M

Table 49: StatusReqAck

PDU content	Status
<?xml version="1.0" encoding="ISO-8859-1"?>	M
<AppSubReq version="1.0" TI="xxxxx">	M
<Header>	M
<ClientID> Client ID </ClientID>	M
</Header>	M
<Body>	M
<Action> StatusReqAck </Action>	M
<Status> OK|NotOK </Status>	M
<Battery> > OK|NotOK </Status>	O
<StatusString> Human readable text </StatusString>	O
</Body >	M
</AppSubReq >	M

A.3.2.3 Get parameter

The server sends "GetParameters" when it wishes the client to report on the settings of the parameters that can be modified by the server or to request the status of a specific parameter. This procedure can also be used by the server to find out which parameters are controllable in the client. Upon receipt, the client shall indicate the parameter and the parameter value by sending "GetParametersAck" service data and including a list of parameters and their current values in the form "parameterName1=parameterValue1", "parameterName2=parameterValue2", ..., "parameterNameN=parameterValueN" where N indicates the last parameter. The parameters names and values should be human readable so that they can be displayed by the server if necessary. If the client does not have configurable parameters it shall include only the parameter's name providing for means, e.g. reporting rate to be set by parameter. If no parameter is listed in the "GetParameters" then all parameters shall be returned by the client in the "GetParametersAck". The server may also ask for the current values of the timers related to the ODAP PT sleeping mode and Alive Indication procedures defined in clause 12.

Table 50: GetParameters

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> GetParameters </Action>	M
		<P> Parameter Name 1 </P>	O
		...	
		<P> Parameter Name N </P>	O
		<TAlive></TAlive>	O (Note 1)
		<TSleep></TSleep>	O (Note 1)
		<TAwake></TAwake>	O (Note 1)
	</Body >		M
</AppSubReq >			M
NOTE 1: Fields if included shall be empty, i.e. no values shall be included. NOTE 2: <P> stands for <Parameter>. Due to the fact that multiple parameters can be included this short tag has been chosen to reduce the length of the PDU.			

Table 51: GetParametersAck

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> GetParametersAck </Action>	M
		<PV> Parameter Name 1 = Parameter Value1 </PV>	C1
		<PV> Parameter Name 1</PV>	C2
		...	
		...	
		<PV> Parameter Name N = Parameter ValueN </PV>	C1
		<PV> Parameter Name N </PV>	C2
		<TAlive>T_alive value</TAlive>	O
		<TSleep>T_sleep value</TSleep>	O
		<TAwake>T_awake value</TAwake>	O
	</Body >		M
</AppSubReq >			M
C1: IF Parameter configurable THEN M, ELSE X. C2: IF Parameter NOT configurable THEN M, ELSE X. NOTE: <PV> stands for <ParameterValue>. Due to the fact that multiple parameter values can be included this short tag has been chosen to reduce the length of the PDU.			

A.3.2.4 Set parameter

The server sends "SetParameters parameterName1=parameterValue1 parameterName2=parameterValue2 parameterNameN=parameterValueN" when it wishes the client to modify one or more of its parameters. Upon receipt, the client shall indicate the new parameter values by sending "SetParametersAck parameterName1=parameterValue1 parameterName2=parameterValue2 parameterNameN=parameterValueN". If the client is unable to change any value then the current maintained value shall be indicated in the SetParametersAck message. This procedure may be used, in addition to the registration procedure, for setting up the values of the timers related to the ODAP PT sleeping mode and Alive Indication procedures defined in clause 12.

Table 52: SetParameters

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> SetParameters </Action>	M
		<PV> Parameter Name 1 = Parameter Value1 </PV>	M
		...	
		<PV> Parameter Name N = Parameter ValueN </PV>	M
		<TAlive>T_alive value</TAlive>	O
		<TSleep>T_sleep value</TSleep>	O
		<TAwake>T_awake value</TAwake>	O
	</Body >		M
</AppSubReq >			M
NOTE 1: Parameter names can be determined using the GetParameters procedure. NOTE 2: <PV> stands for <ParameterValue>. Due to the fact that multiple parameter values can be included this short tag has been chosen to reduce the length of the PDU.			

Table 53: SetParametersAck

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> SetParametersAck </Action>	M
		<PV> Parameter Name 1 = Parameter Value1 </PV>	M
		...	
		<PV> Parameter Name N = Parameter ValueN </PV>	M
		<TAlive></TAlive>	O
		<TSleep></TSleep>	O
		<TAwake></TAwake>	O
	</Body >		M
</AppSubReq >			M
NOTE: <PV> stands for <ParameterValue>. Due to the fact that multiple parameter values can be included this short tag has been chosen to reduce the length of the PDU.			

A.3.2.5 Trace start

The server sends "TraceStart" when it wishes the client to start regular reporting of its measurements. The parameterNames are optional and may be omitted. If so, then the Default Update Rate value applies to all the parameters. Upon receipt, the client begins issuing "SensorData" messages according to the rate specified by R. R indicates the integer number of minutes between updates. If R = 0 then the client issues an updates as soon as new data is available. It is the responsibility of the client in the case that R = 0 to ensure that the data rate is limited to the capabilities of the ODAP bearer. Trace start can also be used to modify the rate of an existing trace which is already reporting "SensorData" messages.

Table 54: TraceStart

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> TraceStart </Action>	M
		<P> Parameter Name 1 </P>	O
		<Rate> Parameter Name 1 Rate </Rate>	M (See note 1)
		...	
		<P> Parameter Name N </P>	O (See note 2)
		<Rate> Parameter Name N Rate </Rate>	O (See note 2)
	</Body >		M
</AppSubReq >			M
NOTE 1: If parameter name was provided before the <Rate> field the Rate value provided is relevant for the parameter only. If no <P> was provided the value is relevant for all parameters handled by this client - in this case no parameter or other rates are allowed. NOTE 2: The fields are to be seen as a filed pair, i.e. either both or non of the fields shall be included. NOTE 3: <P> stands for <Parameter>. Due to the fact that multiple parameters can be included this short tag has been chosen to reduce the length of the PDU.			

A.3.2.6 Trace stop

The server sends "TraceStop" and 0, 1 or more "parameterName N" when it wishes the client to stop regular reporting of either all or part of its measurements as indicated by the list of parameterNames. The parameterNames are optional and may be omitted. If so, then the TraceStop command applies to all the parameters. Upon receipt, the client shall cease issuing "SensorData" messages for the corresponding parameters. If no parameters are listed in the "TraceStop" then tracing of all parameters shall be stopped by the client.

Table 55: TraceStop

PDU content			Status
<?xml version="1.0" encoding="ISO-8859-1"?>			M
<AppSubReq version="1.0" TI="xxxxx">			M
	<Header>		M
		<ClientID> Client ID </ClientID>	M
	</Header>		M
	<Body>		M
		<Action> TraceStop </Action>	M
		<P> Parameter Name 1 </P>	O
		...	
		<P> Parameter Name N </P>	O
	</Body >		M
</AppSubReq >			M
NOTE: <P> stands for <Parameter>. Due to the fact that multiple parameters can be included this short tag has been chosen to reduce the length of the PDU.			

History

Document history		
V1.1.1	July 2004	Publication