BS EN 62516-3:2013

BSI Standards Publication

Terrestrial digital multimedia broadcasting (T-DMB) receivers

Part 3: Common API

raising standards worldwide™

National foreword

This British Standard is the UK implementation of EN 62516-3:2013.

The UK participation in its preparation was entrusted to Technical Committee EPL/100, Audio, video and multimedia systems and equipment.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Published by BSI Standards Limited 2013

ISBN 978 0 580 79793 4

ICS 33.160.25; 33.170

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 April 2013.

Amendments issued since publication

Date	Text affected

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 62516-3

April 2013

ICS 33.160.25; 33.170

English version

Terrestrial digital multimedia broadcasting (T-DMB) receivers - Part 3: Common API

(IEC 62516-3:2013)

Récepteurs pour diffusion multimédia numérique terrestre (T-DMB) - Partie 3: API commune (CEI 62516-3:2013)

Empfänger für terrestrischen Multimediadigitalrundfunk (T-DMB) - Teil 3: Allgemeine API (IEC 62516-3:2013)

This European Standard was approved by CENELEC on 2013-04-15. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Management Centre: Avenue Marnix 17, B - 1000 Brussels

Ref. No. EN 62516-3:2013 E

Foreword

The text of document 100/2020/CDV, future edition 1 of IEC 62516-3, prepared by Technical Area 1 “Terminals for audio, video and data services and contents” of IEC/TC 100 “Audio, video and multimedia systems and equipment” was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 62516-3:2013.

The following dates are fixed:

• latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement	(dop)	2014-01-15
• latest date by which the national standards conflicting with the document have to be withdrawn	(dow)	2016-04-15

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

Endorsement notice

The text of the International Standard IEC 62516-3:2013 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following note has to be added for the standard indicated :

IEC 62104:2003	NOTE	Harmonised as EN 62104:2007 (not modified).

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE When an international publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

Publication	Year	Title	EN/HD	Year
IEC 62516-1	2009	Terrestrial digital multimedia broadcasting (T-DMB) receivers - Part 1: Basic requirement	EN 62516-1	2009
IEC 62516-2	2011	Terrestrial digital multimedia broadcasting (T-DMB) receivers - Part 2: Interactive data services using BIFS	EN 62516-2	2011
ETSI EN 300 401 V1.3.3	-	Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers	-	-

CONTENTS

TERRESTRIAL DIGITAL MULTIMEDIA BROADCASTING (T-DMB) RECEIVERS –

Part 3: Common API

1 Scope

This part of IEC 62516 describes the T-DMB common application program interface (API). It provides a software platform that, when combined with the T-DMB O/S, forms a universal interface for application programs. This interface allows application programs to be written in such a way that they run on any T-DMB receiver unit, as described in IEC 62516-1:2009 and IEC 62516-2:2011 regardless of its manufacturer.

This part of IEC 62516 also defines a software environment that allows multiple application programs to be interoperable on a single receiver unit by sharing the fixed resources of the receiver, and it provides a set of interfaces that the T-DMB middleware and the ASIC specific software use.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 62516-1:2009, *Terrestrial digital multimedia broadcasting (T-DMB) receivers – Part 1: Basic requirements*

IEC 62516-2:2011, *Terrestrial digital multimedia broadcasting (T-DMB) receivers – Part 2: Interactive data services using BIFS*

ETSI EN 300 401 v1.3.3, *Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers*

3 Abbreviations

ADC	Analog to Digital Converter
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
FIC	Fast Information Channel
HAL	Hardware Abstraction Layer
ISR	Interrupt Service Routine
MAC	Media Access Control
PAD	Program Associated Data
RF	Radio Frequency
R-S	Reed Solomon
SDIO	Secure Digital Input/Output
SI	Service Identifier
T-DMB	Terrestrial-Digital Multimedia Broadcasting

O/S Operating System

4 T-DMB common API overview

4.1 T-DMB receiver overview

A T-DMB receiver provides the device functionality specified in the T-DMB receivers (see IEC 62516-1:2009 and IEC 62516-2:2011). Figure 1 shows the block diagram of a typical T-DMB receiver. For the T-DMB receiver depicted in Figure 1, only those blocks that conform to the scope of this standard are shown.

Figure 1 also shows the T-DMB common API with respect to the T-DMB receiver block diagram.

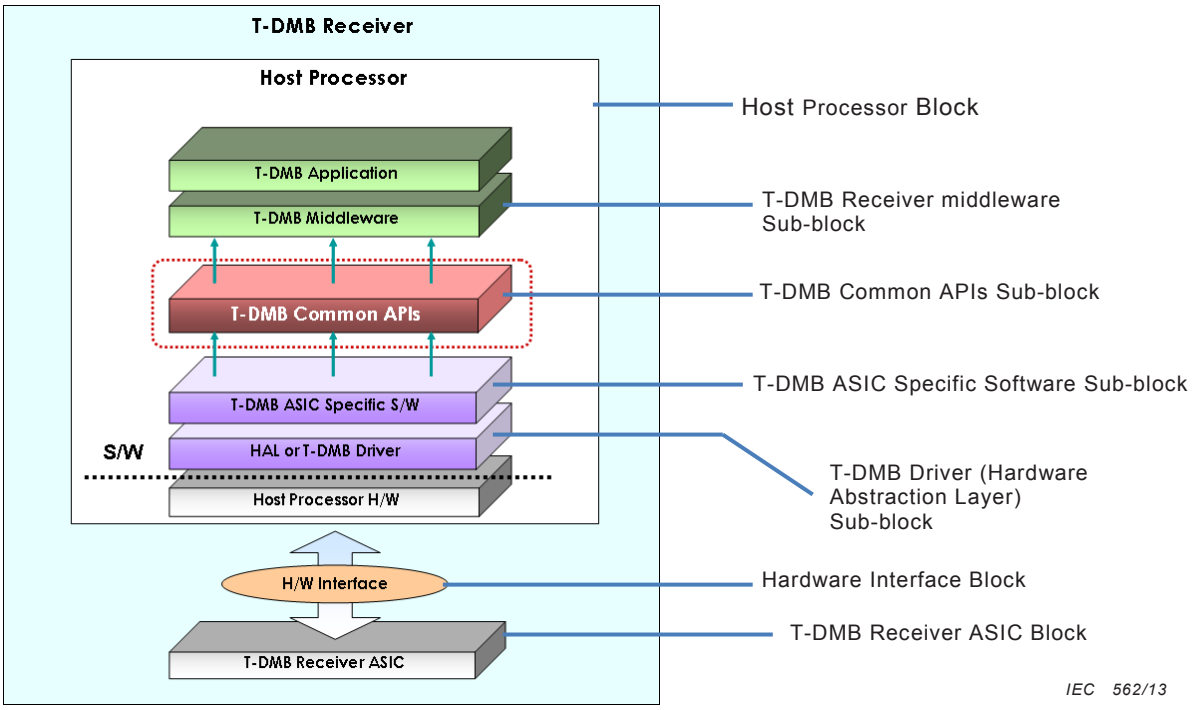


Figure 1 – Block diagram of a typical T-DMB receiver

4.2 T-DMB receiver ASIC block

The T-DMB receiver ASIC block represents the semiconductor hardware that provides the functionality of demodulating a T-DMB signal and retrieving data carried by the T-DMB physical layer. This block provides functionality like RF front-end, ADC, timing and frequency estimation, channel estimation, viterbi decoding, etc. In short this block provides the complete physical layer implementation of ETSI EN 300 401 v1.3.3. Depending upon the implementation, this block can also provide outer decoding functionality (e.g. R-S decoding and/or convolutional de-interleaving).

4.3 Host processor block

4.3.1 General

The host processor block represents the T-DMB functionality provided by the host processor in a T-DMB based device. In other words, this represents the host processor hardware and the software implementation residing in the host processor. The host processor block retrieves and processes the T-DMB information obtained from the T-DMB receiver ASIC block. The T-DMB information retrieved consists of multiplex configuration information received on fast

information channel (FIC), content received on the main service channel. This block communicates with the T-DMB receiver ASIC block to retrieve the information received from the T-DMB signal. The host processor block consists of the following functional sub-blocks.

4.3.2 T-DMB driver (hardware abstraction layer) sub-block

The T-DMB Driver or hardware abstraction layer (HAL) Block represents the driver level software in the main processor that directly interfaces with the T-DMB receiver ASIC block. The T-DMB driver sub-block provides controller functions (e.g. turning on or turning off the T-DMB receiver ASIC block) and data exchange functions (e.g. retrieving the data from the T-DMB receiver ASIC block or conveying the characteristics of a sub-channel to be received) for a given T-DMB receiver ASIC hardware. The T-DMB Driver software is specific to the type of hardware interface mechanism that exists between the Host Processor Block and the T-DMB Receiver ASIC block.

For example, the T-DMB driver software will be different depending upon whether the hardware interface between the main processor and the T-DMB receiver ASIC block is interrupt driven, implemented with memory mapped address/registers or packet based transaction interface like SDIO. Some examples of tasks performed by T-DMB driver sub-block are:

- hardware interactions such as initialization, sleep or wakeup triggers;
- data exchange with hardware such as emptying hardware buffers into main memory or providing ISR implementation.

The T-DMB driver software functions are tightly coupled with the T-DMB receiver ASIC hardware and are considered time sensitive in nature. Therefore the T-DMB driver software is typically given a higher priority with respect to other sub-blocks shown. For example, the T-DMB driver performs the tasks of retrieving the data received by the T-DMB receiver ASIC block or instructing the T-DMB receiver ASIC block to tune to a frequency as requested by the application layer.

4.3.3 T-DMB ASIC specific software sub-block

T-DMB ASIC specific software sub-block provides the MAC layer functionality not covered by the T-DMB driver sub-block. Depending upon the division of MAC layer functionality across different sub-blocks, it may provide complete or partial MAC layer functionality. At the least, the T-DMB ASIC specific software sub-block is expected to provide high level MAC layer functionality that is not practical to be delegated to T-DMB driver sub-block. It interfaces with the T-DMB receiver middleware sub-block using the T-DMB common APIs.

4.3.4 T-DMB common APIs sub-block

The T-DMB common APIs sub-block defines the interfaces that allow the T-DMB ASIC specific software sub-block to communicate with the T-DMB receiver middleware. Any T-DMB receiver middleware that adheres to the interfaces defined by the T-DMB common APIs will work with any T-DMB ASIC specific software sub-block that adheres to these interfaces as well. More details on this interface are provided in the rest of this standard.

4.3.5 T-DMB receiver middleware sub-block

The T-DMB receiver middleware sub-block communicates with the T-DMB ASIC specific software sub-block using the T-DMB common APIs. The T-DMB receiver middleware implements the control and stream layer and provides the interface with application layer protocols. It triggers the T-DMB ASIC specific software to receive the specified contents as requested by the application layer. It acts on the notifications or content provided by the T-DMB ASIC specific software. It delivers any content received from the T-DMB ASIC specific software to the application layer protocols.

4.4 Hardware interface block

The hardware interface block represents the hardware interface mechanism that exists between the host processor block and the T-DMB receiver ASIC block. This interface provides the communication and data exchange functionality. The T-DMB driver sub-block uses this block to exchange commands and data with the T-DMB receiver ASIC block. The hardware interface block can be any desired interface, such as proprietary bus interface or a standard based interface (e.g. SDIO).

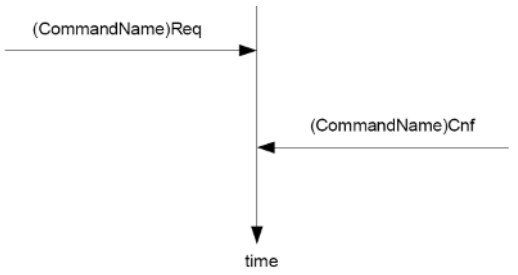
5 API description

5.1 T-DMB common APIs

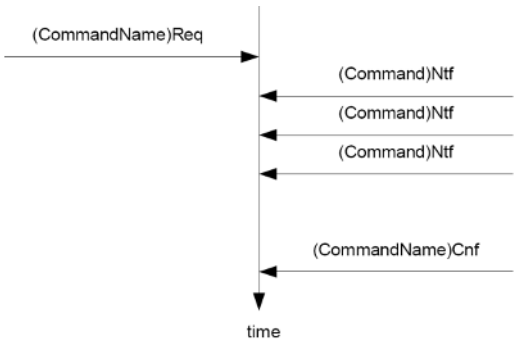
This clause provides a detailed description of each T-DMB common API. The API function prototype details are provided along with defined types needed by the T-DMB common APIs.

Commands are executed by sending requests, confirmations and notifications. Figure 2 shows three different command patterns. These are used in the T-DMB common APIs.

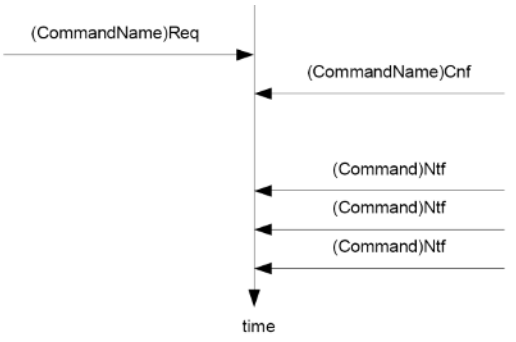
If commands are interleaved which means two commands running at the same time, an arbitrary sequence of message types is possible.



a) Req - Cnf Command Pattern



b) Req - Ntf - Cnf Command Pattern



c) Req - Cnf - Ntf Command Pattern

IEC 563/13

Figure 2 – Three different command patterns

5.2 Command types

5.2.1 General

The commands supported by the T-DMB common APIs can be categorized as follows.

- API-inquiry functions:
  - GetAPIVersion: Returns the API version.
  - Get T-DMBCapability: Returns API's T-DMB receiver capabilities and properties.
- Selecting an ensemble:
  - Tune: Tunes directly to a specified frequency.
  - Search: Searches for an ensemble.
- Accessing service directory:
  - SelectSI: Subscribes to service directory information.
  - GetEnsembleInfo: Gets information about a specified ensemble.
  - GetServiceInfo: Gets information about a specified service.
  - GetComponentInfo: Gets information about a specified component.
- Monitoring reception conditions:
  - SelectReceptionInfo: Subscribes to reception condition information.
- Selecting services:
  - SelectComponent: Starts or stops a service. In case of an audio/video service decoding of audio/video samples is started automatically. In case of a data service, the service can be accessed with the SelectObject command.
  - SelectApplication: Launches a T-DMB application.
  - SelectComponentStream: Gets access to the packet stream of the component.
- Selecting objects:
  - SelectObject: Requests data objects for delivery with or without automatic updating.
- Scanning for T-DMB services:
  - Scan: Scans a specified frequency range for T-DMB ensembles and updates the service directory.
- Miscellaneous:
  - OperationControl: accesses and modifies parameters of the receiver.
  - GetLocationInfo: retrieves location information from the receiver.

5.2.2 Get receiver capability

Figure 3 shows the get receiver capability. The T-DMB common APIs asks the T-DMB receiver for its capabilities.

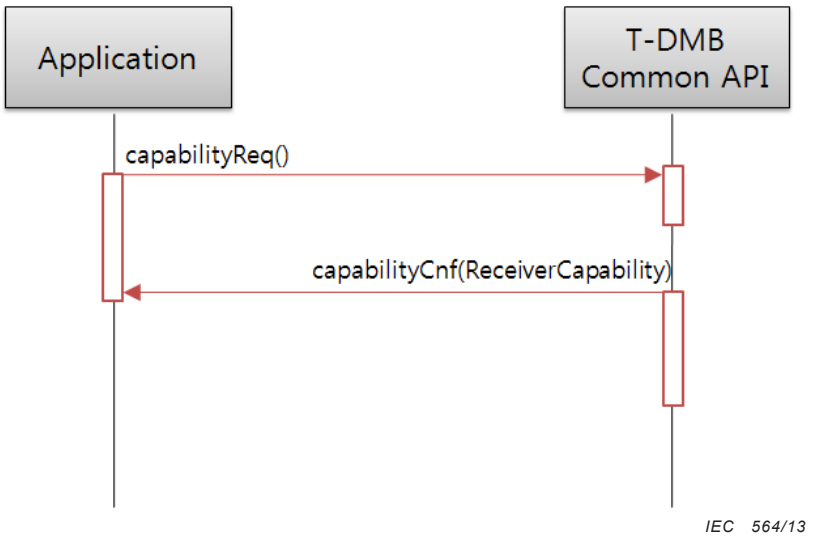


Figure 3 – Get receiver capability

void capabilityReq ()

The T-DMB Common APIs asks the T-DMB receiver for its capabilities.

Parameters

None

void capabilityCnf(ReceiverCapability)

T-DMB receiver provides its capabilities to the API.

5.2.3 Tuning

Figure 4 shows the tuning. The T-DMB receiver is tuned by calling tuneReq. The receiver tunes to the requested frequency and responds afterwards with tuneCnf confirmation. The confirmation contains information about the reception quality.

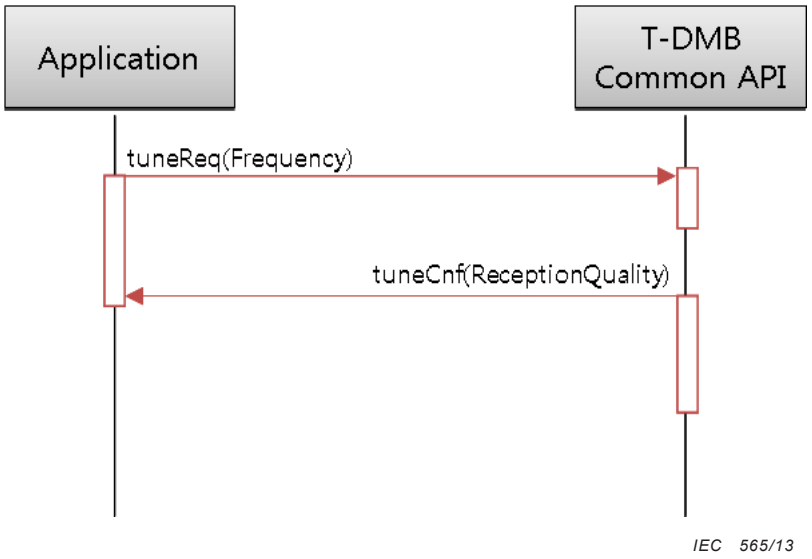


Figure 4 – Tuning

void tuneReq(int tuneFrequency,int transmissionMode)

The tuneReq request initiates the Tune command. The Tune command sets directly a specified T-DMB frequency. A T-DMB receiver shall be tuned to a T-DMB frequency and synchronized in order to get access to T-DMB services. A tuned T-DMB receiver tries automatically to synchronize on a T-DMB ensemble. The Tune command is used to select a

specified T-DMB frequency. Depending on the specification for the transmissionMode it is tested if a T-DMB ensemble can be detected. If the connected T-DMB receiver supports automatic detection the default setting for transmissionMode can be used. Otherwise it has to be specified which transmission modes should be tested. The result of the command is delivered by the tuneCnf confirmation. All currently existing selections of audio and data services or selections of data objects are automatically stopped before tuning is performed by the T-DMB receiver.

Parameters

- tuneFrequency – This parameter specifies the frequency the T-DMB receiver will be tuned to in hertz.
- transmissionModes –
  This parameter specifies the transmission modes a T-DMB receiver tests for T-DMB ensembles. The default value is T-DMBConstants.transmissionModeAutomatic which means that the receiver is automatically detecting the transmission mode. The parameter is a flag field supporting the following flags which can be specified together:
  - T-DMBConstants.transmissionModeAutomatic: The transmission mode is automatically detected. All other flags are ignored in this case.
  - T-DMBConstants.transmissionMode1: At the specified frequency it is tested if a T-DMB ensemble is sent in transmission mode 1.
  - T-DMBConstants.transmissionMode2: At the specified frequency it is tested if a T-DMB ensemble is sent in transmission mode 2.
  - T-DMBConstants.transmissionMode3: At the specified frequency it is tested if a T-DMB ensemble is sent in transmission mode 3.
  - T-DMBConstants.transmissionMode4: At the specified frequency it is tested if a T-DMB ensemble is sent in transmission mode 4.

void tuneCnf(TuneCnfEvent e)

The TuneCnf method finalizes a Tune command and is sent as a response to a TuneReq message. It provides information about the currently selected T-DMBfrequency and reception conditions. The Tune command is used to select a specified T-DMBfrequency. The tuneReq request initiates the Tune command. The tuneCnf finalizes the Tune command and provides information about the reception state. This includes the selected frequency, the detected transmission mode and the synchronization state of the receiver.

5.2.4 Searching

Figure 5 shows the searching. To search for some ensemble, the application calls searchReq. The T-DMB common APIs respond with a notification that the search has started. Other notifications are sent in between depending on the search method (e.g., a 16 kHz step was made). The transaction ends with a searchCnf confirmation containing the resulting state of the search process.

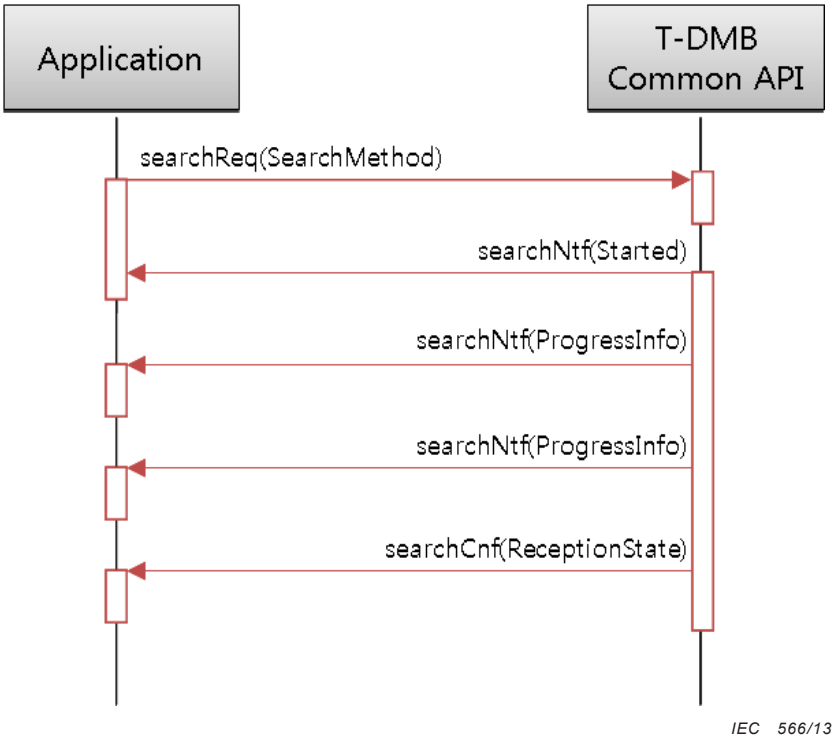


Figure 5 – Searching

void searchReq(
int searchMode,
int tables,
int startFrequency,
int stopFrequency,
int transmissionModes,
int notifications)

The searchReq request initiates a Search command. The Search command searches for a T-DMB ensemble according to a specified search mode. After a successful execution of the Search command a T-DMB ensemble has been found, the state Tuned is entered and the T-DMB receiver tries to synchronize automatically to the found T-DMB ensemble. The Search command is used to search for a T-DMB ensemble. The searchReq request initiates the search and specifies the frequencies and transmission modes to test. Additionally, the notifications that the T-DMB client gets can be specified while the command is executed. Searching for an ensemble may require a substantial amount of time from only a second up to several minutes. This depends also on the search mode specified. If the reception conditions are bad it is possible that no T-DMB ensemble at all is detected. In order to stop searching for a T-DMB ensemble the Tune command can be used which tunes the T-DMB receiver to a certain frequency independent from the reception conditions. The start of searching is indicated by a SearchNtf event with a status code 'Started'. In this case the state machine of the Tune state enters the searching state (see Figure 5). If the previous state has been tuned all currently existing selections of services or objects are stopped automatically. While searching is performed, several notifications delivering information about the current status are sent to the client. The command ends with a SearchCnf event.

Parameters

- **searchMode** – This parameter specifies the way the T-DMB receiver is searching for a T-DMB ensemble. The default value is SearchModeAutomatic which means it is searching according to a default method. The parameter is a flag field supporting the following flags which can be specified together
  - T-DMBConstants.SearchModeAutomatic: default method

  - T-DMBConstants.SearchMode16kHzSteps: The frequency range is searched in 16 kHz steps. This is a very intensive search which means that command execution can take a large amount of time.
  - T-DMBConstants.SearchModeUp: The search direction is from low to high frequencies.
  - T-DMBConstants.SearchModeDown: The search direction is from high to low frequencies.
  - T-DMBConstants.SearchModeUseTables: The search is based on the specified frequency tables.
  - T-DMBConstants.SearchModeUseFrequencyRange: The search is based on the specified frequency range.
  - T-DMBConstants.SearchModeContinuous: The search is looping over the specified frequency range until a T-DMB ensemble has been found. The default is to stop after the specified frequency range has been checked once.
- **tables** – This parameter specifies frequency tables the receiver uses in order to search for T-DMB ensembles. The parameter is a flag field supporting the following flags which can be specified together:
  - T-DMBConstants.searchCEPTFrequencyTableBandIII: The frequencies according to the CEPT frequency table for Band III are tested for T-DMB Ensembles.
  - T-DMBConstants.SearchCEPTFrequencyTableLBand: The frequencies according to the CEPT L-Band table are tested for T-DMB ensembles.
  - T-DMBConstants.SearchCanadaFrequencyTableLBand: The frequencies according to the Canadian L-Band table are tested for T-DMB ensembles.
- **transmissionModes** – This parameter specifies the transmission modes a T-DMB receiver tests for T-DMB ensembles. The default value is T-DMBConstants.transmissionModeAutomatic which means that the receiver is automatically detecting the transmission mode. The parameter is a flag field supporting the following flags which can be specified together:
  - transmissionModeAutomatic: The transmission mode is automatically detected. All other flags are ignored.
  - transmissionMode1: At the specified frequency it is tested if a T-DMB ensemble is sent in transmission mode 1.
  - transmissionMode2: At the specified frequency it is tested if a T-DMB ensemble is sent in transmission mode 2.
  - transmissionMode3: At the specified frequency it is tested if a T-DMB ensemble is sent in transmission mode 3.
  - transmissionMode4: At the specified frequency it is tested if a T-DMB ensemble is sent in transmission mode 4.
- **notifications** – This parameter specifies the type of notifications the client wants to get while the Seek command is performed. The parameter is a flag field supporting the following flags which can be specified together:
  - notificationsOff: No intermediate notifications are sent. Only a SearchNtf notification which informs about the start of searching is sent.
  - notifications16kHzSteps: With each 16 kHz step a notification is sent. This is used only if 16 kHz step searching is specified as search mode.
  - notificationsTableEntry: With each table entry frequency a notification is sent. This is the default value.

void searchCnf(SearchCnfEvent e)

The searchCnf method finalizes a Search command and provides information about the command status, currently selected T-DMB frequency and current reception conditions. The Search command is used in order to search for a T-DMB ensemble according to a specified search mode. Searching for a T-DMB ensemble can take a large amount of time. The start of searching is indicated by a 'Started' searchNtf message. Other searchNtf messages inform a

T-DMB client about search progress. It is finalized by delivery of the searchCnf message. It informs about the command status, the selected frequency and the synchronization state. No further searchNtf messages will be delivered after the searchCnf message.

void searchNtf(SearchNtfEvent e)

The SearchNtf event is sent after a search for a T-DMB Ensemble was started searchReq. It informs about the start of searching and about the progress of searching. A SearchCnf event finalizes a Search command. No more SearchNtf events are sent after a SearchCnf event was sent. The SearchNtf event is sent after the searching for a T-DMB ensemble has been started and while searching is in progress in order to provide information about the current status of searching. The 'Started' notification is sent in any case. Progress notifications are only sent if notifications have been requested with the related SearchReq message. No further notifications will be sent after a SearchCnf message is delivered.

5.2.5 Scanning

Figure 6 shows the scanning. The scanning means looking for ensembles in a specified range. Essentially, it is like searching except that the scanning process looks for all ensembles in the range. When the command has been issued, notification will be sent, after the scanning has been started. Further notifications are sent during the scan, which inform about the progress. When the scan is terminated, a confirmation is sent, which contains information about the scan and the state of the receiver.

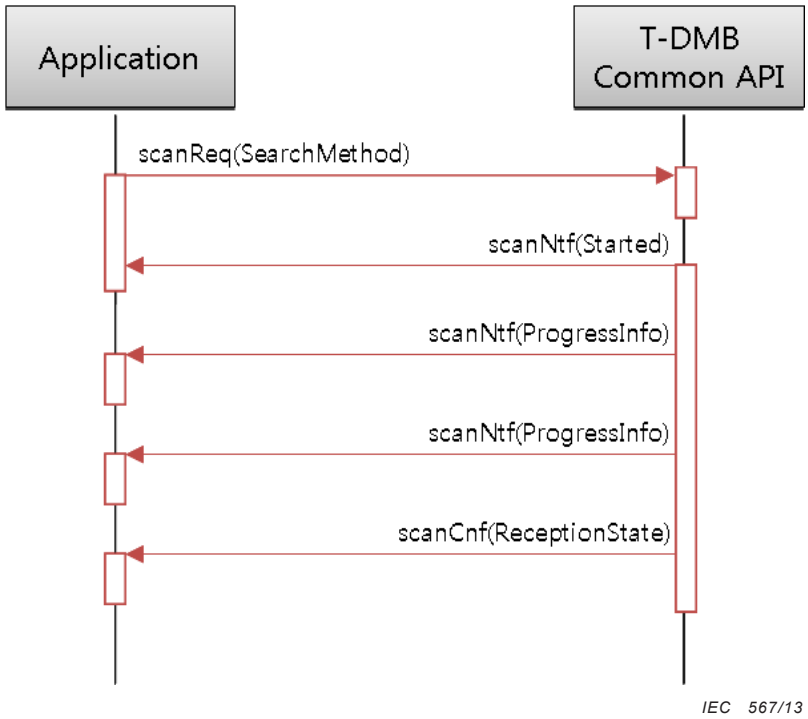


Figure 6 – Scanning

void scanReq(
int searchMode,
int tables,
int startFrequency,
int stopFrequency,
int tansmissionModes,
int notifications)

The ScanReq request initiates a Scan command. The Scan command is used in order to perform a search for all available T-DMB ensembles in a specified frequency range. Depending on the frequency range and the search mode this operation may require a substantial amount of time from only a second up to several minutes. The command is started by the ScanReq request and is finished with the ScanCnf confirmation. In between ScanNtf

notification are sent in order to inform about the current status of scanning if notifications are requested. In case of searching from lower to higher frequencies (searchMode= T-DMBConstants.searchModeUp).

The value of startFrequency is not allowed to be larger than the value of stopFrequency. In case of searching from higher to lower frequencies (searchMode= T-DMBConstants.searchModeDown) the value of startFrequency is not allowed to be smaller than the value of stopFrequency.

Parameters

- **searchMode** – This parameter specifies the way the T-DMB receiver is searching for a T-DMB ensemble. The default value is T-DMBConstants.searchModeAutomatic which means it is searching according to a default method. The parameter is a flag field supporting the following flags which can be specified together:
  - T-DMBConstants.searchModeAutomatic: default method
  - T-DMBConstants.searchMode16kHzSteps: The frequency range is searched in 16 kHz steps.
  - T-DMBConstants.searchModeUp: The search direction is from low to high frequencies.
  - T-DMBConstants.searchModeDown: The search direction is from high to low frequencies.
  - T-DMBConstants.searchModeUseTables: The search is based on the specified frequency tables.
  - T-DMBConstants.searchModeUseFrequencyRange: The search is based on the specified frequency range.
  - T-DMBConstants.searchModeContinuous: The search is looping over the specified frequency range until a T-DMB Ensemble has been found. The default is to stop after the specified frequency range has been checked once.
- **tables** – This parameter specifies frequency tables the receiver uses in order to search for T-DMB ensembles. The parameter is a flag field supporting the following flags which can be specified together:
  - T-DMBConstants.searchCEPTFrequencyTableBandIII: The frequencies according to the CEPT frequency table for Band III are tested for T-DMB ensembles.
  - T-DMBConstants.searchCEPTFrequencyTableLBand: The frequencies according to the CEPT L-Band table are tested for T-DMB ensembles.
  - T-DMBConstants.searchCanadaFrequencyTableLBand: The frequencies according to the Canadian L-Band table are tested for T-DMB ensembles.
- **startFrequency** – This parameter specifies the start frequency at which the T-DMB receiver starts its search for T-DMB ensembles.
- **stopFrequency** – This parameter specifies the stop frequency at which the T-DMB receiver stops its search for T-DMB ensembles.
- **transmissionModes** – This parameter specifies the transmission modes a T-DMB receiver should look for T-DMB ensembles. The default value is T-DMBConstants.transmissionModeAutomatic which means that the receiver is automatically detecting the transmission mode. The parameter is a flag field supporting the following flags which can be specified together:
  - T-DMBConstants.transmissionModeAutomatic: The transmission mode is automatically detected.
  - T-DMBConstants.transmissionMode1: At the specified frequency it is tested if a T-DMB ensemble is sent in transmission mode 1.
  - T-DMBConstants.transmissionMode2: At the specified frequency it is tested if a T-DMB ensemble is sent in transmission mode 2.

– T-DMBConstants.transmissionMode3: At the specified frequency it is tested if a T-DMB ensemble is sent in transmission mode 3.

– T-DMBConstants.transmissionMode4: At the specified frequency it is tested if a T-DMB ensemble is sent in transmission mode 4.

– notifications – This parameter specifies the type of notifications wanted by the application while the Seek command is performed. The parameter is a flag field supporting the following flags which can be specified together:

– T-DMBConstants.notificationsOff: No notifications are sent.

– T-DMBConstants.notifications16kHzSteps: With each 16 kHz step a notification is sent.

– T-DMBConstants.notificationsTableEntry: With each table entry frequency a notification is sent. This is the default value.

void scanCnf(ScanCnfEvent e)

The ScanCnf message finalizes a Scan command. It informs about the result of scanning and the current tune state. The Scan command is used in order to perform a search for all available T-DMB ensembles in a specified frequency range. Depending on the frequency range and the search mode this operation may require a substantial amount of time from only a second up to several minutes. The command is started by the ScanReq message and is finished with the ScanCnf message. In between ScanNtf messages are sent in order to inform about the current status of searching if notifications are requested.

The ScanCnf message indicates that the Scan command is finished and informs about the current tune state. As a result of performing the Scan command the service information database is filled with information. If a SI subscription is running several SINtf messages are delivered to the connected application.

void scanNtf(ScanNtfEvent e)

The ScanNtf message is sent after a search for all available T-DMB Ensembles in a specified frequency range is started by the ScanReq message. The ScanNtf message provides information about the current status of searching for all available T-DMB Ensembles in a specified frequency range. It is delivered to the connected application after the search has been started by the ScanReq message and notifications have been requested. No further notifications will be sent after a ScanCnf message is delivered.

5.2.6 Selecting a T-DMB service

Figure 7 shows the selecting a T-DMB service. An audio/video service is started with the SelectComponent command. The application calls selectComponentReq passing the identifier of the audio/video component. The T-DMB common APIs will start the audio/video service and send back a confirmation. To stop this audio/video service, the application calls selectComponentReq again now specifying that the component has to be removed. When the T-DMB common APIs respond with a confirmation, the audio/video service has been stopped.

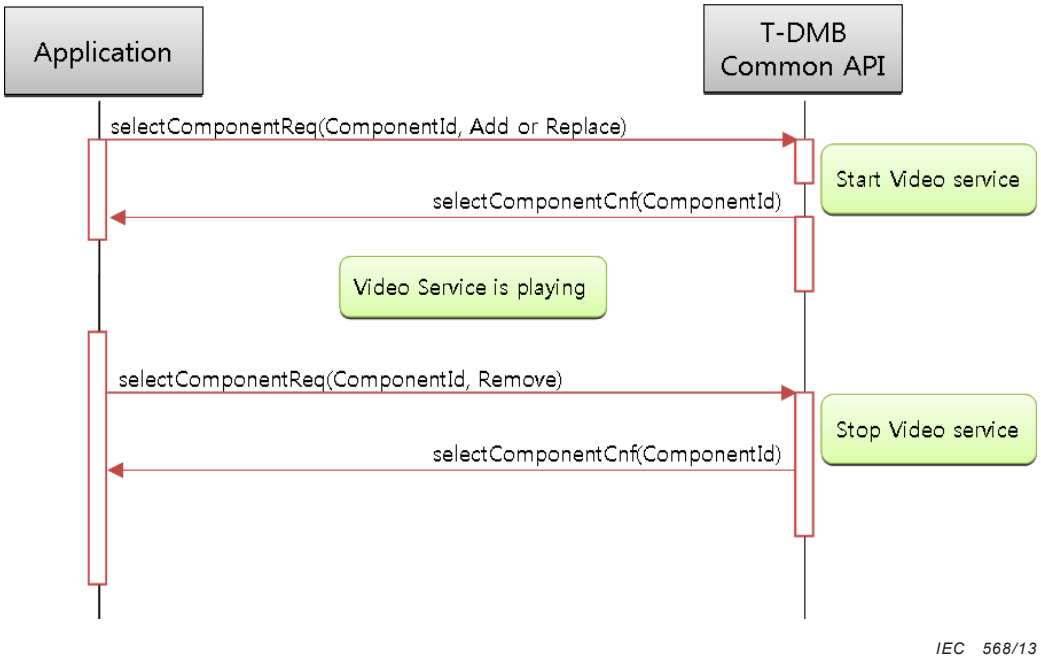


Figure 7 – Selecting a T-DMB service

void selectComponentReq(
ComponentId id,
int selectionMode)

The selectComponentReq request initiates the SelectComponent command. The SelectComponent command starts or stops an application delivered in a T-DMB component. The SelectComponent command allows to start or stop applications delivered in T-DMB components. In general, more than one component of the same T-DMB ensemble can be selected simultaneously. It is possible to select one audio component, all programme-associated data components of the selected audio component and more than one independent data component at the same time. The selection of a component is requested by the selectComponentReq request and is confirmed by the selectComponentCnf confirmation. It is possible that a component is removed from a T-DMB ensemble which means it is no longer broadcast and therefore no longer available. This is indicated by a SINtf call and means that the selection is removed automatically. If the selection of a component is removed also all existing object selections belonging to the component are removed.

If the user application is a slide show or a dynamic label, its objects are delivered automatically (using objectNtf notifications) after the SelectComponent confirmation was sent. If the selected component is an audio service, its PAD data services become available as well. This means service information is generated for all PAD services and they can be selected. If the selection of the audio service is stopped, also all PAD services are stopped and they are not available anymore. If the component is not in the current ensemble, it depends on the implementation whether it is selected nevertheless.

Parameters

- **id** – This parameter is a pointer to the identifier of the T-DMB component which is to be selected. If all component selections should be removed (set selectionMode to T-DMBConstants.selectionModeRemoveAll) this parameter is ignored and should be set to null.
- **selectionMode** – This parameter specifies the selection mode for the component. The following flags are supported:
  - T-DMBConstants.selectionModeReplace: All currently selected components of the same type are stopped and the specified component is to be started. The same type means an audio component replaces any other selected audio component, a data component replaces all other selected independent data components and a

programme-associated data component replaces all other selected programme-associated data components.

– T-DMBConstants.selectionModeAdd: The application delivered by the specified component is to be started. Other selected components are not affected.
– T-DMBConstants.selectionModeRemove: The selection of the specified component is stopped.
– T-DMBConstants.selectionModeRemoveAll: All existing component selections are removed. Set serviceId to null in this case.

void selectComponentCnf(SelectComponentCnfEvent e)

The SelectComponentCnf confirmation finalizes the SelectComponent command. It informs about the command status and the selection status of the specified component. The SelectComponent command allows to start or to stop applications delivered in T-DMB components. In general, more than one component of the same T-DMB ensemble can be selected simultaneously. It is possible to select one audio component, all programme-associated data components of the selected audio component, and more than one independent data component at the same time. The selection of a component is requested by the selectComponentReq message and is confirmed by a selectComponentCnf call. It is possible that a component is removed from a T-DMB ensemble which means it is no longer broadcast and therefore no longer available. This is indicated by a SINtf call and means that the selection is removed automatically.

5.2.7 Selecting a slideshow or a dynamic label service

Figure 8 shows the selection of a slideshow or a dynamic label service. An application selects a slideshow or a dynamic label service with the SelectComponent command. When the request selectComponentReq with the respective service identifier is issued, the service gets started and a confirmation is sent back. The application will then receive objectNtf notifications containing objects of the service. To stop the service, selectComponentReq is called again by setting selectionMode to selectionModeRemove. The removal of the service will be confirmed.

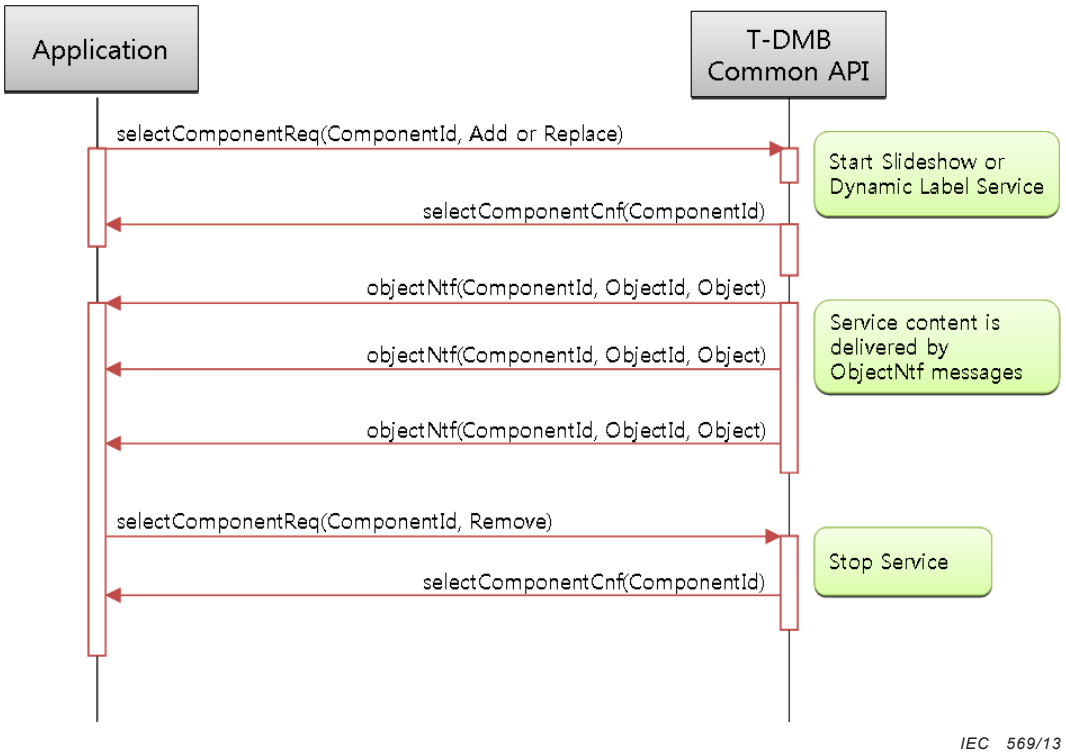


Figure 8 – Selecting a slideshow or a dynamic label service

void objectNtf(ObjectNtfEvent e)

The objectNtf method is called as a consequence of selecting objects from a data component by use of the SelectObject command. It delivers a selected object partially or complete to a T-DMB client. objectNtf is used to deliver a selected object to the connected T-DMB client. Depending on the request mode the object is delivered only once or more than once in case of updates. If the object cannot be delivered in-time as indicated by a call to selectObjectCnf, then objectNtf informs about the delay. If transmission of a selected object is stopped, objectNtf informs about the termination of the object transmission and the object selection. It is possible that a T-DMB component is removed from a T-DMB ensemble. This is indicated by a call to siNtf. In this case also the selected objects of the component are no longer selected. No termination messages are sent for terminated object selections resulting from termination of a component.

5.2.8 Selecting a broadcast website service

Figure 9 shows the selection of a broadcast website service. To run a broadcast website service the component has to be selected. This is accomplished by calling selectComponentReq with the respective service identifier. The start of the service is confirmed by the T-DMB common APIs. The actual objects of the service are retrieved with the selectObject command. Usually, the start object is demanded first. For that, a selectObjectReq request is issued with the service identifier of the component and the object identifier of the start object. The T-DMB common APIs will send back a confirmation including the likely access time. The actual object is received with an objectNtf notification. All other objects of the service are requested and delivered similarly. The service is stopped calling selectComponentReq specifying the removal of the service. Note, that the SelectComponent command can be used to improve the access time of the requested time (e.g. especially caching the objects of the service).

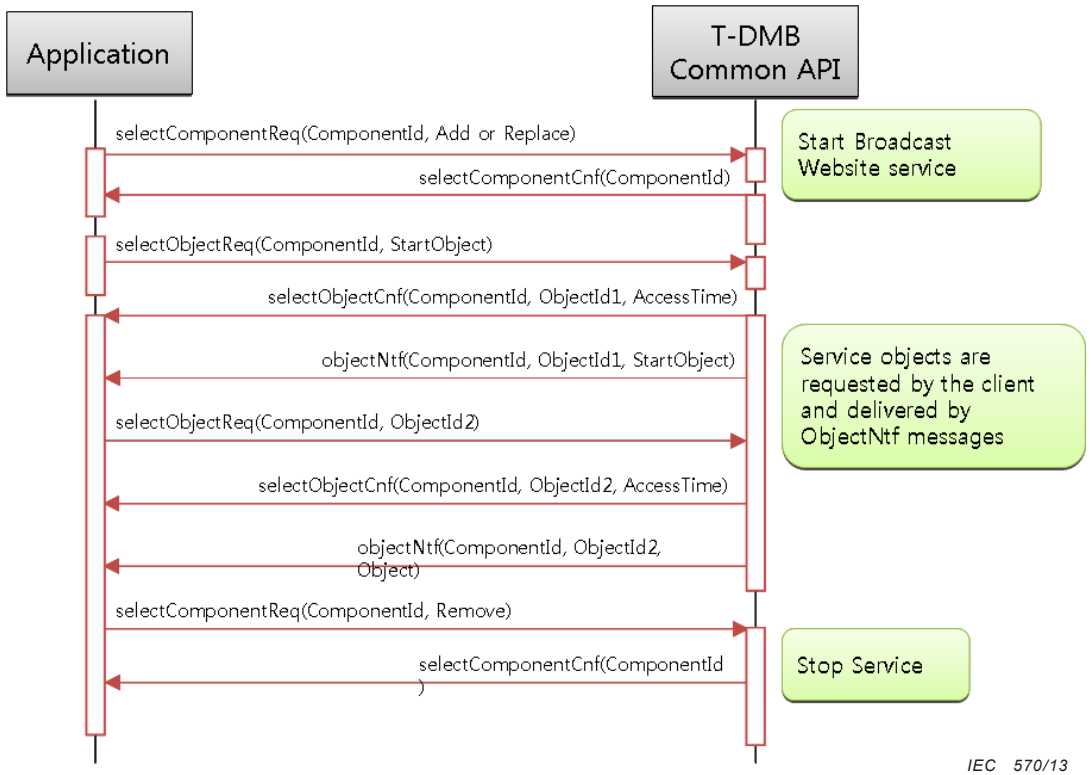


Figure 9 – Selecting a broadcast website service

void selectObjectReq(
ComponentId id,
ObjectId objectId,
int requestMode,
boolean replaceSelections,
int deliveryMode,

int cacheHint)

The selectObjectReq request initiates the SelectObject command. The SelectObject command selects an object from a selected T-DMB component. This includes requesting an object from a data component, delivery after reception and notification of updates as long as the object is selected. Selection means it is requested for delivery and if wanted also updates of the object are delivered. Additionally, it is possible to give some hints for caching. More than one object and also from more than one component can be selected simultaneously. The selection of an object is requested by the selectObjectReq request and is confirmed by the selectObjectCnf confirmation. The object is delivered using the objectNtf method. This includes first-time delivery and all updates. Beyond starting or stopping a selection, it is possible to remove all other selections belonging to the same component by setting parameter replaceSelections to true. It is possible to remove a component from a T-DMB ensemble. This is indicated by a serviceInfoNtf call. In this case also the selected objects of the service are no longer selected. It is possible to remove an object from a current on-air service. This is indicated by an objectNtf call. In this case the selections for this object are automatically disabled.

Currently, object selection makes only sense with applications of type BroadcastWebSite. Objects of applications like Slideshows or Dynamic Label are delivered automatically by objectNtf calls.

Parameters

- **id** – This parameter identifies the selected component the object is belonging to.
- **objectId** – This parameter identifies the object which is to be selected.
- **selectionMode** – This parameter specifies the selection mode of the object. The following values are supported:
  - T-DMBConstants.requestModeOff: This is used in order to stop the selection of objects which are requested with the request mode T-DMBConstants.requestModeUpdate. It is not needed for objects which are requested with the T-DMBConstants.requestModeOnce flag except when a SelectObjectReq is pending and the delivery is no longer wanted.
  - T-DMBConstants.requestModeOnce: The object is requested for one-time delivery. After the first reception from the broadcast channel the object is delivered to the connected T-DMB client. The client is not notified about new versions.
  - T-DMBConstants.requestModeUpdate: The object is requested for update delivery. After the first reception from the broadcast channel the object is delivered to the connected client. Additionally, each new version of the object is delivered.
- **replaceSelections** – This parameter specifies whether all current object selections belonging to the component identified by serviceId are replaced with this selection. If this parameter is set to true, then all selections are removed. If this parameter is set to false, then existing selections remain unchanged.
- **deliveryMode** – This parameter specifies the delivery mode of the object. The following values are supported:
  - T-DMBConstants.deliveryModeComplete: Only the complete object is delivered to the T-DMB client.
  - T-DMBConstants.deliveryModePartial: The object may be delivered in parts.
- **cacheHint** – This parameter specifies a hint for caching of the selected object.

void selectObjectCnf(SelectObjectCnfEvent e)

The SelectObjectCnf method finalizes the SelectObject command. The SelectObject command selects an object from a selected T-DMB component. This includes requesting an object from a data component, delivery after reception and notification of updates as long as the object is selected. Selection means, it is requested for delivery and if wanted also updates of the object are delivered. Additionally, it is possible to give some hints for caching. More than one object can be selected simultaneously as well as from more than one component. The selection of an object is requested by selectObjectReq and is confirmed by calling

selectObjectCnf. The object is delivered using objectNtf. This includes first-time delivery and all updates. Beyond starting or stopping a selection, it is possible to remove all other selections belonging to the same component by setting parameter replaceSelections to true. It is possible to remove a component from a T-DMB ensemble. In this case also the selected objects of the service are no longer selected. It is possible that an object is removed from current on-air service. This is indicated by calling objectNtf. In this case, the selections for this object are automatically disabled. Currently, object selection makes only sense with applications of type BroadcastWebSite. Objects of applications like Slideshows or Dynamic Label are delivered automatically using objectNtf.

5.2.9 Get T-DMB service information

Figure 10 shows the Get T-DMB service information. The application can also use the ServiceInfo command to retrieve the respective T-DMB service information objects. It has to specify the service identifier in the siReq request. The confirmation will then contain the requested object.

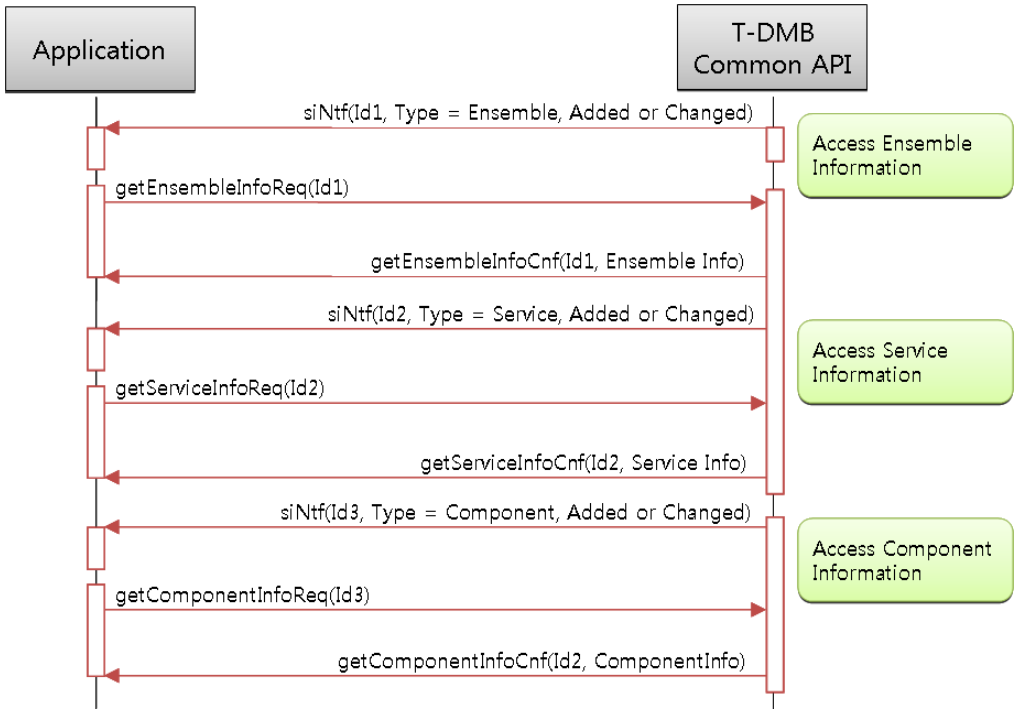


Figure 10 – Get T-DMB service information

void getEnsembleInfoReq(EnsembleId id)

The getEnsembleInfoReq method initiates a GetEnsembleInfo command. The GetEnsembleInfo command requests information about the specified T-DMB ensemble. The GetEnsembleInfo command provides a T-DMB client with information about a specified T-DMB ensemble, e.g. label, No of services, and so on. The command is initiated by a getEnsembleInfoReq request and is finished by a getEnsembleInfoCnf confirmation.

Parameters

- **id** – This parameter is a handle identifying the T-DMB ensemble.

void getServiceInfoReq(ServiceId id)

The getServiceInfoReq requests initiates a GetServiceInfo command. The GetServiceInfo command requests information about a specified T-DMB service. The GetServiceInfo command provides a T-DMB client with information about a specified T-DMB service, e.g. label, No of components, and so on. The command is initiated by a getServiceInfoReq request and is finished by a getServiceInfoCnf confirmation.

Parameters

- **id** – This parameter is a handle identifying the T-DMB service.

void getComponentInfoReq(ComponentId id)

The getComponentInfoReq request initiates a GetComponentInfo command. The GetComponentInfo command requests information about a specified T-DMB component. The GetComponentInfo command provides a T-DMB client with information about a specified T-DMB component, e.g. label, language, and so on. The command is initiated calling getComponentInfoReq and is finished by a call to getComponentInfoCnf.

Parameters

- **id**- This parameter is a handle identifying the T-DMB component.

void getEnsembleInfoCnf(GetEnsembleInfoCnfEvent e)

The GetEnsembleInfoCnf method finalizes the GetEnsembleInfo command and delivers information about a requested T-DMB ensemble to a T-DMB client. The GetEnsembleInfo command provides a T-DMB client with information about a specified T-DMB ensemble, e.g. label, No of services, and so on. The command is initiated by a getEnsembleInfoReq request and is finished by a getEnsembleInfoCnf call.

void getServiceInfoCnf(GetServiceInfoCnfEvent e)

A call to the getServiceInfoCnf method finalizes the GetServiceInfo command and delivers information about a requested T-DMB Service to a T-DMB client. The GetServiceInfo command provides a T-DMB client with information about a specified T-DMB Service, e.g. label, No of services, and so on. The command is initiated by a getServiceInfoReq message and is finished by a getServiceInfoCnf message.

void getComponentInfoCnf(GetComponentInfoCnfEvent e)

The GetComponentInfoCnf message finalizes the GetComponentInfo command and delivers information about a requested T-DMB component to a T-DMB client. The GetComponentInfo command provides a T-DMB client with information about a specified T-DMB component, e.g. label, language and so on. The command is initiated by a GetComponentInfoReq request and is finished by a call to getComponentInfoCnf message.

5.2.10 Monitoring reception qualities

Figure 11 shows the monitoring reception qualities. The reception quality can be monitored using the SelectReceptionInfo command. The application has to make a selectReceptionInfoReq request specifying what parameters are monitored. Then it receives receptionInfoNtf notifications as long as the monitoring is not stopped.

(selectionReceptionInfoReq (Off)).

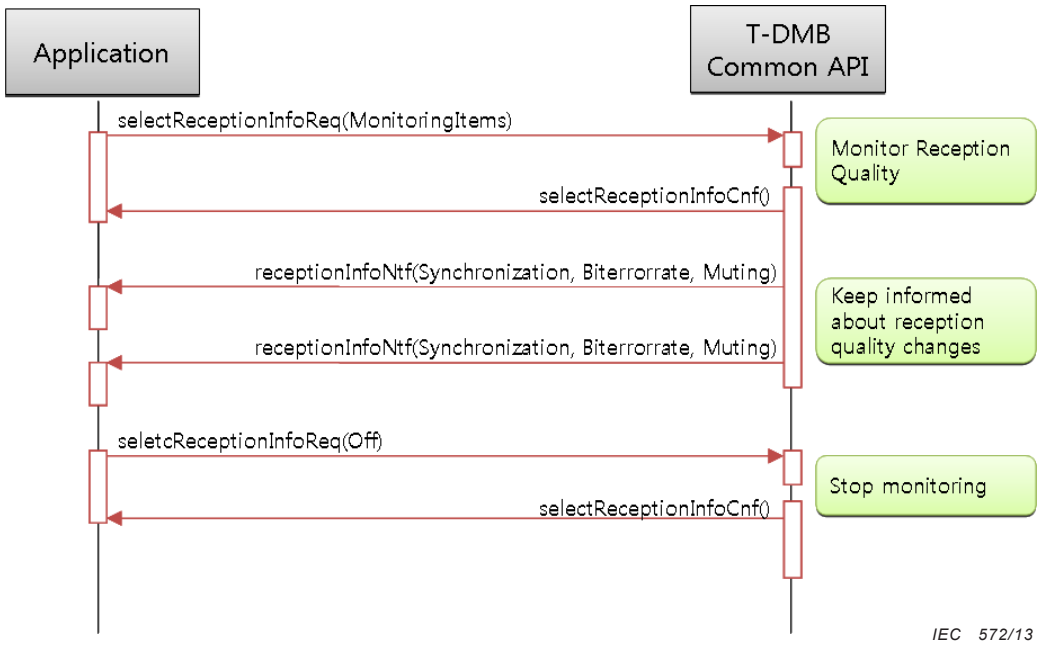


Figure 11 – Monitoring reception qualities

void selectReceptionInfoReq(
boolean synchronizationNotification,
boolean bitErrorRateNotifcations,
boolean muteStateNotifications,
boolean requestOnce)

The selectReceptionInfoReq request initiates the SelectReceptionInfo command. The SelectReceptionInfo command starts, stops or changes subscription to state change notifications concerning reception conditions. It is possible to monitor synchronization, bit-error-rate and audio decoder muting. The SelectReceptionInfo command allows a T-DMB client to subscribe for state change notifications concerning reception conditions in terms of synchronization, bit-error-rate and audio decoder muting. The subscription is requested by the selectReceptionInfoReq request and is confirmed with the selectReceptionInfoCnf confirmation. The subscription level can be changed by another SelectReceptionInfo command. This includes stopping of subscription. After a successful subscription the calling T-DMB client receives ReceptionInfoNtf notifications when state changes occur.

Parameters

- **synchronizationNotification** – This parameter specifies if the calling client is notified about state changes concerning T-DMB signal synchronization. If the parameter is set to true (default) notifications are provided, if it is set to false no notifications are provided.
- **bitErrorRateNotifications** – This parameter specifies if the calling client is notified about state changes concerning the bit-error-rate. If the parameter is set to true (default) notifications are provided, if it is set to false no notifications are provided.
- **muteStateNotifications** – This parameter specifies if the calling client is notified about state changes concerning the mute state of the audio decoder. If the parameter is set to true (default) notifications are provided, if it is set to false no notifications are provided.
- **requestOnce** – This parameter specifies if the reception condition information is wanted only once. In this case the reception condition is once detected and the T-DMB client informed by one and only one receptionInfoNtf call.

void selectReceptionInfoCnf(SelectReceptionInfoCnfEvent e)

The selectReceptionInfoCnf method finalizes the SelectReceptionInfo command. It informs about the command status and the current subscription level. The selectReceptionInfo method allows a T-DMB client to subscribe for state change notifications concerning reception conditions in terms of synchronization, bit-error-rate and audio decoder muting. The subscription is requested by selectReceptionInfoReq and is confirmed with

selectReceptionInfoCnf. The subscription level can be changed by another SelectReceptionInfo command. This includes stopping of subscription. After a successful subscription the calling T-DMB client receives receptionInfoNtf calls when state changes occur.

void receptionInfoNtf(ReceptionInfoNtfEvent e)

The receptionInfoNtf method is called as a consequence of subscribing to state changes in synchronization, bit-error-rate and audio decoder muting. receptionInfoNtf indicates that the synchronization state, bit-error-rate or mute state has changed (see ReceptionInfoNtfEvent). The ReceptionInfoNtf message is provided to a connected client as a result of subscription to state change notifications concerning reception conditions (selectReceptionInfoReq and selectReceptionInfoCnf messages).

Annex A
(informative)

Examples of the classes used in T-DMB APIs

In this annex, examples of the classes used in T-DMB APIs are described.

SearchCnfEvent(
T-DMBSource source,
int result,
int tuneState,
int tuneFrequency,
int transmissionMode,
int synchronizationState)

SearchNtfEvent(
T-DMBSource source,
int tuneFrequency,
int notifications)

ScanNtfEvent(
T-DMBSource source,
int tuneFrequency,
int notifications)

ScanCnfEvent(
T-DMBSource source,
int result,
int tuneState,
int tuneFrequency,
int transmissionModes,
int noOfEnsemblesFound)

GetEnsembleInfoCnfEvent(
T-DMBSource source,
int result,
EnsembleInfo ensembleInfo)

GetServiceInfoCnfEvent(
T-DMBSource source,
int result,
ServiceInfo serviceInfo)
GetComponentInfoCnfEvent(
T-DMBSource source,
int result,
ComponentInfo componentInfo)

SelectReceptionInfoCnfEvent(
T-DMBSource source,
int result,
boolean synchronizationNotifications,
boolean bitErrorRateNotifications,
boolean muteStateNotifications)

ReceptionInfoNtfEvent(
T-DMBSource source,
int updateFlags,
int synchronizationState,
int bitErrorRateState,

```
int muteState)

SelectComponentCnfEvent(
T-DMBSource source,
int result,
ComponentId componentId,
int selectionMode)

ObjectNtfEvent(
T-DMBSource source,
ComponentId componentId,
ObjectId objectId,
int selectionState,
T-DMBObject object)

SelectObjectCnfEvent(
T-DMBSource source,
int result,
ComponentId componentId,
ObjectId objectId,
int requestMode,
boolean replaceSelections,
Date accessTime)

ComponentInfo(
ComponentId id,
int type,
byte[] data,
boolean isPrimary,
ServiceId[] parentIds,
int accessControlSystem,
boolean hasLabel,
Label label,
boolean hasLanguage,
int language,
boolean hasStartObjectId,
ObjectId startObjectId,
boolean hasObjectDirectoryId,
ObjectId objectDirectoryId,
boolean hasAudioComponent,
ComponentId audioComponent,
boolean hasBitrate,
int bitrate)

ServiceInfo(
ServiceId id,
int type,
EnsembleId parent,
ComponentId[] componentIds,
boolean isLocal,
int accessControlSystem,
boolean hasLabel,
Label label,
boolean hasLanguage,
int language,
boolean hasIsPrimary,
boolean isPrimary,
boolean hasRegionId,
int regionId,
boolean hasRegionLabel,
Label regionLabel,
boolean hasStaticProgrammeType,
```

ProgrammeType staticProgrammeType,
boolean hasDynamicProgrammeType,
ProgrammeType dynamicProgrammeType,
boolean hasProgrammeNumber,
ProgrammeNumber programmeNumber,
boolean hasTimeOffset,
int timeOffset,
boolean hasAnnouncementSupport,
AnnouncementSupport announcementSupport,
boolean hasCountry,
int country)

EnsembleInfo(
EnsembleId id,
ServiceId[] serviceIds,
int frequency,
int transmissionMode,
boolean hasDate,
Date date,
boolean hasLabel,
Label label,
boolean hasCountry,
int country)

Bibliography

IEC 62104:2003, *Characteristics of DAB receivers*

ISO 10486:1992, *Passenger cars – Car radio identification number (CRIN)*

ETSI EN 50094:1992, *Access control system for the MAC/packet family: EurocryptETSI TS 101 993 v1.1.1 Digital Audio Broadcasting (DAB); A Virtual Machine for DAB: DAB Java Specification*

_____

This page deliberately left blank

British Standards Institution (BSI)

BSI is the independent national body responsible for preparing British Standards and other standards-related publications, information and services. It presents the UK view on standards in Europe and at the international level.

BSI is incorporated by Royal Charter. British Standards and other standardisation products are published by BSI Standards Limited.

Revisions

British Standards and PASs are periodically updated by amendment or revision. Users of British Standards and PASs should make sure that they possess the latest amendments or editions.

It is the constant aim of BSI to improve the quality of our products and services. We would be grateful if anyone finding an inaccuracy or ambiguity while using British Standards would inform the Secretary of the technical committee responsible, the identity of which can be found on the inside front cover. Similary for PASs, please notify BSI Customer Services.

Tel: +44 (0)20 8996 9001 Fax: +44 (0)20 8996 7001

BSI offers BSI Subscribing Members an individual updating service called PLUS which ensures that subscribers automatically receive the latest editions of British Standards and PASs.

Tel: +44 (0)20 8996 7669 Fax: +44 (0)20 8996 7001
Email: plus@bsigroup.com

Buying standards

You may buy PDF and hard copy versions of standards directly using a credit card from the BSI Shop on the website **www.bsigroup.com/shop.** In addition all orders for BSI, international and foreign standards publications can be addressed to BSI Customer Services.

Tel: +44 (0)20 8996 9001 Fax: +44 (0)20 8996 7001
Email: orders@bsigroup.com

In response to orders for international standards, BSI will supply the British Standard implementation of the relevant international standard, unless otherwise requested.

Information on standards

BSI provides a wide range of information on national, European and international standards through its Knowledge Centre.

Tel: +44 (0)20 8996 7004 Fax: +44 (0)20 8996 7005
Email: knowledgecentre@bsigroup.com

BSI Subscribing Members are kept up to date with standards developments and receive substantial discounts on the purchase price of standards. For details of these and other benefits contact Membership Administration.

Tel: +44 (0)20 8996 7002 Fax: +44 (0)20 8996 7001
Email: membership@bsigroup.com

Information regarding online access to British Standards and PASs via British Standards Online can be found at **www.bsigroup.com/BSOL**

Further information about British Standards is available on the BSI website at **www.bsi-group.com/standards**

Copyright

BSI

389 Chiswick High Road London W4 4AL UK

Tel +44 (0)20 8996 9001
Fax +44 (0)20 8996 7001
www.bsigroup.com/standards

raising standards worldwide™