MOST

Media **O**riented **S**ystems **T**ransport

Multimedia and Control Networking Technology

MOST FBlock DABTuner

Rev 4.0.1

11/2011

MOSTCO CONFIDENTIAL

See page 3 for the terms of disclosure

Legal Notice

COPYRIGHT

LICENSE DISCLAIMER

Nothing on any MOST Cooperation Web Site, or in any MOST Cooperation document, shall be construed as conferring any license under any of the MOST Cooperation or its members or any third party's intellectual property rights, whether by estoppel, implication, or otherwise.

CONTENT AND LIABILITY DISCLAIMER

MOST Cooperation or its members shall not be responsible for any errors or omissions contained at any MOST Cooperation Web Site, or in any MOST Cooperation document, and reserves the right to make changes without notice. Accordingly, all MOST Cooperation and third party information is provided "AS IS". In addition, MOST Cooperation or its members are not responsible for the content of any other Web Site linked to any MOST Cooperation Web Site. Links are provided as Internet navigation tools only.

MOST COOPERATION AND ITS MEMBERS DISCLAIM ALL WARRANTIES WITH REGARD TO THE INFORMATION (INCLUDING ANY SOFTWARE) PROVIDED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

In no event shall MOST Cooperation or its members be liable for any damages whatsoever, and in particular MOST Cooperation or its members shall not be liable for special, indirect, consequential, or incidental damages, or damages for lost profits, loss of revenue, or loss of use, arising out of or related to any MOST Cooperation Web Site, any MOST Cooperation document, or the information contained in it, whether such damages arise in contract, negligence, tort, under statute, in equity, at law or otherwise.

FEEDBACK INFORMATION

Any information provided to MOST Cooperation in connection with any MOST Cooperation Web Site, or any MOST Cooperation document, shall be provided by the submitter and received by MOST Cooperation on a non-confidential basis. MOST Cooperation shall be free to use such information on an unrestricted basis.

TRADEMARKS

MOST Cooperation and its members prohibit the unauthorized use of any of their trademarks. MOST Cooperation specifically prohibits the use of the MOST Cooperation LOGO unless the use is approved by the Steering Committee of MOST Cooperation.

SUPPORT AND FURTHER INFORMATION

For more information on the MOST technology, please contact:

MOST Cooperation
Administration
Bannwaldallee 48
D-76185 Karlsruhe
Germany

Tel: (+49) (0) 721 966 50 00

E-mail: contact@mostcooperation.com
Web: www.mostcooperation.com

This Specification is Confidential Information of the MOST Cooperation. It may only be disclosed to member companies. Member companies wishing to discuss these Specifications with suppliers or other third parties must ensure that a commercially standard form of non-disclosure agreement has been previously executed by the party receiving such Specifications. Use of these Specifications may only be for purposes for which they are intended by the MOST Cooperation. Unauthorized use or disclosure is a violation of law.

Contents

Bibliography

All documents, which this MOST document have references to, are listed here with the actual revision this document is referring to.

Number	Document	Revision
[1]	ETS 300 401: Radio broadcasting systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers	V1.3.3
[2]	TR 101 496: Digital Audio Broadcasting (DAB); Guidelines and Rules of Implementation and Operation	V1.1.1
[3]	TS 101 756: Digital Audio Broadcasting (DAB); Registered Tables	V1.2.1
[4]	EN 50248: Characteristics of DAB Receivers	08/03
[5]	EN 50067: Specification of the radio data system (RDS) for VHF/FM broadcasting	1992
[6]	ETS 300 250: Television systems; ETSI/EBU Joint Technical Committee (JTC) Specification of the D2-MAC/packet	12/93
[7]	TS 101 499: Digital Audio Broadcasting (DAB); Slide Show Application	V1.1.1
[8]	TS 101 498 - 1: Digital Audio Broadcasting (DAB); Broadcast Web Site Application, Part 1: User Application Specification	V1.1.1
[9]	TS 101 498 - 2: Digital Audio Broadcasting (DAB); Broadcast Web Site Application, Part 2: Basic Profile Specification	V1.1.1
[10]	ISO/IEC 11172-3: Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to 1,5 Mbit/s – Audio Part	1993
[11]	EN 50255: Digital Audio Broadcasting System – Specification of the Receiver Interface (RDI) IEC 62105: Digital Audio Broadcasting System – Specification of the Receiver Interface (RDI)	1997 2002
[12]	MOST Specification	2.5
[13]	MOST Specification	3.0
[14]	MOST FBlock GeneralFBlock	2.5.1
[15]	MOST FBlock GeneralFBlock	3.0.3

Document History

Changes DABTuner FBlock Rev 4.0 to DABTuner FBlock Rev 4.0.1

Change Ref.	Section	Changes
4V01_001	General	– Removed list of released FBlocks, which is of no relevance to this FBlock. – Added references to GeneralFBlock. – Corrected parameter position value, which was always off by one. – Correction of clerical errors. – Added SymbolicNames to all Enums. – Added Occurrence attribute to all functions. – Corrected data type information for ErrorCode and ErrorInfo parameters. – Corrected wrong type references. – Corrected data format – Removed "full range" entry from integer types.
4V01_002	Bibliography	Updated references to MOST Specification to Revision 2.5 and Revision 3.0 and GeneralFBlock Revision 2.5.1 and Revision 3.0.3.
4V01_003	0x211	CurrentAudioService: – Modelled parameter AnnouncementSupport as BoolField. – Modeled parameter PTYCodes as Classified Stream.
4V01_004	0x220	Learn: Changed from Sequence Method to Unclassified Method.
4V01_005	0x221	Seek: Changed from Sequence Method to Unclassified Method.
4V01_006	0x222	Tune: Corrected XML elements that belong to an Unclassified Method but reside within a Sequence Method.
4V01_007	0x223	SelectAudioService: Corrected XML elements that belong to an Unclassified Method but reside within a Sequence Method.
4V01_008	0x224	SelectAudioComponent: Corrected XML elements that belong to an Unclassified Method but reside within a Sequence Method.
4V01_009	0x400	AnnouncementFilter: – Corrected XML elements that belong to an Unclassified Method but reside within a Sequence Method. – Modeled parameter AnnouncementType as BoolField. – Modeled parameter NewFlag as BoolField. – Modeled parameter AnnouncementSource as BoolField.
4V01_010	0x402	AnnouncementStatus: Modeled parameter AnnouncementType as BoolField.
4V01_011	0x403	AnnouncementSwitch: Modeled parameter AnnouncementType as BoolField.
4V01_012	0x440	GetSinkHandle: – Changed from Sequence Method to Unclassified Method. – Removed OPTypes that do not use the SenderHandle parameter.
4V01_013	0x441	ReleaseSinkHandle: Removed OPTypes that do not use the SenderHandle parameter.
4V01_014	0x450	ServiceFollowingMode: Renamed parameter from “Tag” to “Pos” because Tag is not used in function class Record. Modified parameter description accordingly.
4V01_015	0x471	SeekStatus: Changed from Sequence Property to Unclassified Property.
4V01_016	0x472	TuneStatus: Changed from Sequence Property to Unclassified Property.

Change Ref.	Section	Changes
4V01_017	0x473	SelectAudioServiceStatus: Changed from Sequence Property to Unclassified Property.
4V01_018	0x474	SelectAudioComponentStatus: Changed from Sequence Property to Unclassified Property.
4V01_019	0x510	AudioServiceList: – Modeled parameter AnnouncementSupport as BoolField. – Modeled parameter PTYCodes as Classified Stream.
4V01_020	0x713	CurrentDataUserApp: Modeled parameter UserAppData as Classified Stream.
4V01_021	0x723	SelectDataService: Corrected XML elements that belong to an Unclassified Method but reside within a Sequence Method.
4V01_022	0x724	SelectDataComponent: Corrected XML elements that belong to an Unclassified Method but reside within a Sequence Method.
4V01_023	0x725	SelectUserApp: Corrected XML elements that belong to an Unclassified Method but reside within a Sequence Method.
4V01_024	0x800	CreateDataChannel: Corrected XML elements that belong to an Unclassified Method but reside within a Sequence Method.
4V01_025	0x810	MotherDataChannel: Changed parameter DataStreamType from BitField to BoolField.

Changes DABTuner FBlock Rev 3.0 to DABTuner FBlock Rev 4.0

Change Ref.	Section	Changes
4V0_001	2.1.62, 2.1.64	Function CreateDataChannel, DataChannelList, Parameter FktIDMotherDataChannel Datatype modified from Unsigned Byte to Unsigned Word
4V0_002	2.1.22, 2.1.23, 2.1.24, 2.1.25, 2.1.26, 2.1.52, 2.1.53, 2.1.54, 2.1.55, 2.1.56, 2.1.57, 2.1.58	FunctionID of Learn, Seek, Tune, SelectAudioService, SelectAudioComponent, SelectDataService, SelectDataComponent, SelectUserApp, DataServiceList, DataComponentList, DataUserAppList, DataDecoderCap modified
4V0_003	2.1.41, 2.1.42, 2.1.43	Functions LearnStatus. SeekStatus and TuneStatus added
4V0_004	2.1.55, 2.1.56, 2.1.57, 2.1.58	Functions DataServicList, DataComponentList, DataUserAppList and DataDecoderCapp changed FktIDs
4V0_005	3	New chapter Dynamic Specification added; Message Sequence Charts
4V0_006	3.2.1	DAB Tuner Notification Matrices added
4V0_007	3.4.1	DataServices Notification Matrices added
4V0_008	3.4.3	Transmit DataService List renamed to Mandatory DataService Status Messages

Changes DABTuner FBlock Rev 2.3 to DABTuner FBlock Rev 3.0

Change Ref.	Section	Changes
3V0_001	All	DABHandle renamed to Sinkhandle in every context
3V0_002	2.1.23	Function 'Seek' The Datatype of the SinkHandle from UByte to UWord modified. The Coding of the parameter SeekMode reordered to (up/down)
3V0_003	All	Parameter Label8CharAbbreviation description modified. Link to DAB-Spec added.
3V0_004	All	Datatype of the parameter Label changed from Classified stream to String. The Parameter LabelCharset deleted.
3V0_005	2.1.19	Function CurrentAudioService PosY renumbered
3V0_006	2.1.29, 2.1.30	Function AnnouncementStatus and AnnouncementSwitch Parameter AnnouncementType FunctionClass changed from Number to Boolfield.
3V0_007	2.1.47	Function AudioServiceList parameter AnnouncementSupport FunctionClass changed from Number to Boolfield.
3V0_008	2.1.21	Function CurrentTunerStatus Parameter EnsECC, ServiceType and SignalQuality description added/modified
3V0_009	All	The representation of the Datatype Boolean harmonized (e.g.,0x00 to false, 0x01 to true)
3V0_010	2.1.21	Function CurrentTunerStatus parameter renamed from FrequencyBand to FrequencyLabel and added a example 12C, 5E to description
3V0_011	2.1.23, 2.1.24, 2.1.25, 2.1.26, 2.1.27, 2.1.38, 2.1.39	Function Seek, Tune, SelectAudioService, SelectComponent, AnnouncementFilter, PresetSave, PresetSelect Functionclass changed to Unclassified Property
3V0_012	2.1.27	Function AnnouncementFilter description extended
3V0_013	2.1.32	Function CurrentRadioText Parameter Charset deleted
3V0_014	All	Definition changed for *SID* and *SCIDI*
3V0_015	2.1.22, 2.1.23, 2.1.24, 2.1.25, 2.1.26	Convert properties *Learn, Seek*, *Tune, SelectAudioService, SelectComponent* into methods.
3V0_016	2.1.49 – 2.1.65	Add chapter Data Services to this document
3V0_017	[11]	Add new reference for DAB RDI
3V0_018	All	*ServiceType* changed to *SIDType* (unsigned byte)
3V0_019	2.1.37	New function *SetPTYFilter*
3V0_020	All	Functionclass Unclassified Method replaced by Sequence Method
3V0_021	2.1.26	Function *SelectComponent* renamed to *SelectAudioComponent*
3V0_022	2.1.27, 2.1.38, 2.1.39	Function *AnnouncementFilter, PresetSave, PresetSelect* Functionclass changed to Sequence Property
3V0_023	2.1.23, 2.1.39	Parameter *SeekMode* modified
3V0_024	2.1.24	Function *Tune*, Parameter *SelectioMode* modified
3V0_025	2.1.25	FuntionFunction *SelectAudioService* Parameter *SelectionMode* modified
3V0_026	2.1.26	Function *SelectAudioComponent*, Parameter *SelectionMode* modified

1 Introduction

A MOST Function Catalog is a collection of MOST FunctionBlocks.
This document contains the specification of a FunctionBlock. MOST FunctionBlocks are standardized and maintained by MOST workgroup Device Architecture (WG_DA). In order to speed up the process of making new Function Blocks available, every Function Block will be updated individually as required.

1.1 Definitions

Note: In case of Classified Stream “application/octet-stream” is used as Media Type.

Term	Description
Available ensemble	Available ensemble(s) are tunable ensembles “in the air”.
Known ensemble	Known ensemble(s) are stored ensembles in a buffer or a list after e.g., a tune or learn procedure. What is considered to be known depends on the database strategy of each supplier (e.g., depending on single/double tuner implementation).
Tuned ensemble	A tuned ensemble is currently tuned ensemble which can be received at a frequency x. **Note:** A single tuner is able to tune only one ensemble at the same time; a double tuner is able to tune two different ensembles at the same time.
Service IDentifier (SID)	The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SID Type is needed for data services. The service identifier (SID) (see [1], [2]) Programme Service: Byte 0 (Bit 0...7): Programme Service Type (0x00 undef; 0x01 AudioService) Byte 1 (Bit 8...15): ExtendedCountryCode (ECC) Byte 2, 3 (Bit 16...31): ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id) Data Service: Byte 0-2 (Bit 0...23): DataServiceID (20 Bit Service-Reference + 4 Bit Country-id) Byte 3 (Bit 24...31): ExtendedCountryCode (ECC)
Service Component Identifier Internal (SCIDI)	In DAB several Ids are used to identify different components. The Subchannel ID (SubChId) is used for stream mode components. The Subchannel ID combined with packet address is used to identify packet mode components. The TCId and FIG extension (FIDCID) identifies components in FIC. To simplify the selection procedure for service components, the receiver internally generates a unique identifier for a service component. This unique identifier is referred to as Service Component Identifier Internal (SCIDI). If the HMI gets an SCIDI from the receiver, it is valid until the ensemble where it comes from is not tuned in anymore (e.g., if another ensemble is tuned in) or the service component is changed or removed from the ensemble during a reconfiguration.
SenderHandle	The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.
SinkHandle	A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

2 FBlock Definition

2.1 DABTuner (FBlockID=0x43)

In addition to the functions contained in this document, the following functions are also part of this FBlock. They exist in the GeneralFBlock template and are included here by reference.

For GeneralFBlock Rev. 2.5.1, the included functions are:

FktID	Function name
0x000	FktIDs
0x001	Notification
0x002	NotificationCheck
0x010	Version
0x090	CreateArrayWindow
0x091	DestroyArrayWindow
0x092	MoveArrayWindow
0x093	SearchArrayWindow
0x094	LongArrayInfo
0x100	SourceInfo
0x101	Allocate
0x102	DeAllocate
0x103	SourceActivity
0x104	SourceName
0x105	SourceConnect
0x106	SourceDisConnect
0x107	SourceRouting
0x116	SyncDataInfo

For GeneralFBlock Rev. 3.0.3, the included functions are:

FktID	Function name
0x000	FktIDs
0x001	Notification
0x002	NotificationCheck
0x011	FBlockInfo
0x090	CreateArrayWindow
0x091	DestroyArrayWindow
0x092	MoveArrayWindow
0x093	SearchArrayWindow
0x094	LongArrayInfo
0x100	SourceInfo
0x101	Allocate
0x102	DeAllocate
0x103	SourceActivity
0x104	SourceName
0x116	StreamDataInfo

Function Overview		
FktID	**Name**	**Occurrence**
0x210	CurrentEnsemble	Mandatory
0x211	CurrentAudioService	Mandatory
0x212	CurrentAudioComponent	Mandatory
0x213	CurrentTunerStatus	Mandatory
0x220	Learn	Mandatory
0x221	Seek	Mandatory
0x222	Tune	Mandatory
0x223	SelectAudioService	Mandatory
0x224	SelectAudioComponent	Mandatory
0x400	AnnouncementFilter	Optional
0x401	AnnouncementPriority	Optional
0x402	AnnouncementStatus	Optional
0x403	AnnouncementSwitch	Optional
0x410	DRCSwitch	Optional
0x420	CurrentRadiotext	Optional
0x430	FrequencyTable	Optional
0x440	GetSinkHandle	Optional
0x441	ReleaseSinkHandle	Optional
0x450	ServiceFollowingMode	Optional
0x451	SetPTYFilter	Optional
0x460	PresetSave	Optional
0x461	PresetSelect	Optional
0x462	AudioPresets	Optional
0x470	LearnStatus	Optional
0x471	SeekStatus	Optional
0x472	TuneStatus	Optional
0x473	SelectAudioServiceStatus	Optional
0x474	SelectAudioComponentStatus	Optional
0x500	EnsembleList	Optional
0x510	AudioServiceList	Optional
0x520	AudioComponentList	Optional
0x711	CurrentDataService	Optional
0x712	CurrentDataComponent	Optional
0x713	CurrentDataUserApp	Optional
0x723	SelectDataService	Optional
0x724	SelectDataComponent	Optional
0x725	SelectUserApp	Optional
0x730	DataServiceList	Optional
0x740	DataComponentList	Optional
0x750	DataUserAppList	Optional
0x760	DataDecoderCap	Optional
0x773	SelectDataServiceStatus	Optional
0x774	SelectDataComponentStatus	Optional
0x775	SelectUserAppStatus	Optional
0x800	CreateDataChannel	Optional
0x801	RemoveDataChannel	Optional
0x805	DataChannelList	Optional
0x810	MotherDataChannel	Optional

2.1.1 CurrentEnsemble (0x210)

Occurrence: Mandatory

List with general information (EnsID, EnsECC, Label) about the tuned ensemble with respect to the SinkHandle.

Feature: General, Information

2.1.1.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Boolean Number String } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	CurrentEnsemble (0x210)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.1.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, EnsID, EnsECC, LabelStatus, Label8CharAbbreviation, Label }
		{ x=Tag, y=0 }	Tag[Tag], EnsID[Tag], EnsECC[Tag], LabelStatus[Tag], Label8CharAbbreviation[Tag], Label[Tag]
		{ x=Tag, y=2 }	EnsID[Tag]
		{ x=Tag, y=3 }	EnsECC[Tag]
		{ x=Tag, y=4 }	LabelStatus[Tag]
		{ x=Tag, y=5 }	Label8CharAbbreviation[Tag]
		{ x=Tag, y=6 }	Label[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsID

Ensemble idendifier

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsECC

Extended country code of the Ensemble

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

LabelStatus

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	Label valid
		False	Label not valid

Label8CharAbbreviation

Bitfield which is added with the Label to get an abbreviation. This is a Bitmask from DAB please see Spec [1].

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Label

It shall be coded as a string of 16 characters

Basis data type	MaxSize
String	16

2.1.2 CurrentAudioService (0x211)

Occurrence: Mandatory

List with general information about the tuned audio service identified by the SinkHandle.

Feature: Audio, Information

2.1.2.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Boolean Number String Boolean Number Classified Stream } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	CurrentAudio Service (0x211)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.2.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, SID, LabelStatus, Label8CharAbbreviation, Label, AnnouncementSupport, PTYTypeMask, PTYCodes }
		{ x=Tag, y=0 }	Tag[Tag], SID[Tag], LabelStatus[Tag], Label8CharAbbreviation[Tag], Label[Tag], AnnouncementSupport[Tag], PTYTypeMask[Tag], PTYCodes[Tag]
		{ x=Tag, y=2 }	SID[Tag]
		{ x=Tag, y=3 }	LabelStatus[Tag]
		{ x=Tag, y=4 }	Label8CharAbbreviation[Tag]
		{ x=Tag, y=5 }	Label[Tag]
		{ x=Tag, y=6 }	AnnouncementSupport[Tag]

Basis data type	Length	Description	
		{ x=Tag, y=7 }	PTYTypeMask[Tag]
		{ x=Tag, y=8 }	PTYCodes[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

LabelStatus

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	Label valid
		False	Label not valid

Label8CharAbbreviation

Bitfield which is added with the Label to get an abbreviation. This is a Bitmask from DAB please see Spec [1].

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Label

It shall be coded as a string of 16 characters

Basis data type	MaxSize
String	16

AnnouncementSupport

Supported Announcements by the service.

Basis data type	Bit #	Code	Description
Unsigned Word	Bit 0	False	Alarm not supported
		True	Alarm supported
	Bit 1	False	Road Traffic Flash not supported
		True	Road Traffic Flash supported
	Bit 2	False	Transport Flash not supported
		True	Transport Flash supported
	Bit 3	False	Warning/service not supported
		True	Warning/service supported
	Bit 4	False	News flash not supported
		True	News flash supported
	Bit 5	False	Area Weather Flash not supported
		True	Area Weather Flash supported
	Bit 6	False	Event Announcement not supported
		True	Event Announcement supported
	Bit 7	False	Special event not supported
		True	Special event supported
	Bit 8	False	Program information supported
		True	Program information not supported
	Bit 9	False	Sport report not supported
		True	Sport report supported
	Bit 10	False	Financial report not supported
		True	Financial report supported
	Bit 11	False	Not used
		True	Not used
	Bit 12	False	Not used
		True	Not used
	Bit 13	False	Not used
		True	Not used
	Bit 14	False	Not used
		True	Not used
	Bit 15	False	Not used
		True	Not used

PTYTypeMask

The PTYTypeMask describes how to interpret the PTYCodes. Bit 0 corresponds to Byte 0 in the Parameter PTYCodes and so on.

Bit 0: International Code
Bit 1: International Code Dynamic
Bit 2: Coarse Code
Bit 3: Coarse Code Dynamic
Bit 4: Fine Code
Bit 5: Fine Code Dynamic
Bit 6: Second Fine Code
Bit 7: Second Fine Code Dynamic

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

PTYCodes

Stream of PTY codes in order of the PTYTypeMask. E.g. if only 2 PTYTypes are avialable, only this 2 codes will be transmitted. PtyCodes are regional dependent, regional information (country Id and ECC OR international table) can be derived from a property which the system master should contain (is not implemented in a protocol yet). Only the codes are transferred, not the labels.

Basis data type	Length	Media type
Classified Stream	8	application/octet-stream

2.1.3 CurrentAudioComponent (0x212)

Occurrence: Mandatory

List with general information about the tuned service component identified by the SinkHandle.

Feature: Audio, Information

2.1.3.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Boolean Number String Number Number Number } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	CurrentAudio Component (0x212)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.3.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, SID, SCIDI, LabelStatus, Label8CharAbbreviation, Label, Language, Flags, ASCTy }
		{ x=Tag, y=0 }	Tag[Tag], EnsID[Tag], EnsECC[Tag], SID[Tag], SCIDI[Tag], LabelStatus[Tag], Label8CharAbbreviation[Tag], Label[Tag], Language[Tag], Flags[Tag], ASCTy[Tag]
		{ x=Tag, y=2 }	SID[Tag]
		{ x=Tag, y=3 }	SCIDI[Tag]
		{ x=Tag, y=4 }	LabelStatus[Tag]
		{ x=Tag, y=5 }	Label8CharAbbreviation[Tag]
		{ x=Tag, y=6 }	Label[Tag]
		{ x=Tag, y=7 }	Language[Tag]

Basis data type	Length	Description	
		{ x=Tag, y=8 }	Flags[Tag]
		{ x=Tag, y=9 }	ASCTy[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

SCIDI

Data Service Component ID internal, virtual ID assigned by the tuner, unique within all known ensembles

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

LabelStatus

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	Label valid
		False	Label not valid

Label8CharAbbreviation

Bitfield which is added with the Label to get an abbreviation. This is a Bitmask from DAB please see Spec [1].

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Label

It shall be coded as a string of 16 characters

Basis data type	MaxSize
String	16

Language

Service component language.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Flags

Additional information on the service component.

Bit0 False = Primary component
True = Secondary component
Bit1 False = CA not valid (see [1], [2])
True = CA valid (see [1], [2])
Bit2..7 Not used

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

ASCTy

Audio Service Component Type (foreground, background, multi-channel audio).

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

2.1.4 CurrentTunerStatus (0x213)

Occurrence: Mandatory

Status of the DAB Tuner with respect to the SinkHandle.

Feature: General, Information

2.1.4.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Number Number Number Boolean Boolean Boolean String Number Number } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	CurrentTuner Status (0x213)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.4.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, EnsID, EnsECC, SIDType, SID, SCIDI, Sync, ServiceComponentValid, Mute, FrequencyLabel, Frequency, SignalQuality }
		{ x=Tag, y=0 }	Tag[Tag], EnsID[Tag], EnsECC[Tag], SIDType[Tag], SID[Tag], SCIDI[Tag], Sync[Tag], ServiceComponentValid[Tag], Mute[Tag], FrequencyLabel[Tag], Frequency[Tag], SignalQuality[Tag]
		{ x=Tag, y=2 }	EnsID[Tag]
		{ x=Tag, y=3 }	EnsECC[Tag]
		{ x=Tag, y=4 }	SIDType[Tag]
		{ x=Tag, y=5 }	SID[Tag]
		{ x=Tag, y=6 }	SCIDI[Tag]

Basis data type	Length	Description	
		{ x=Tag, y=7 }	Sync[Tag]
		{ x=Tag, y=8 }	ServiceComponentValid[Tag]
		{ x=Tag, y=9 }	Mute[Tag]
		{ x=Tag, y=10 }	FrequencyLabel[Tag]
		{ x=Tag, y=11 }	Frequency[Tag]
		{ x=Tag, y=12 }	SignalQuality[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsID

EnsID consists of the Ensemble identifier. (see [1], [2])

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsECC

Extended country code of the Ensemble

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

SIDType

Programme or data service structure of SID (e.g. derived from P/D-Flag of the FIG 0 field, see [1], [2]): SID-Type: 0x01 Programme Service 0x02 Data Service

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	0...23):	
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

SCIDI

Internal Service Component ID of currently tuned service component

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Sync

Reception status of the DAB Tuner (e.g. in a tunnel there may be no reception).

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	DAB signal detected
		False	no DAB Signal

ServiceComponentValid

E.g. if no service following is active; it may happen that the DABTuner stays on an ensemble where the selected service is not present. In this case the audio service component is invalid.

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	service component is valid
		False	service component is invalid (not in current ensemble)

Mute

Indicates, if the audio signal is muted by DABTuner, e.g due to bad signal quality

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	audio channel is muted, no audio output
		False	audio is not muted, audio is audible

FrequencyLabel

Currently tuned frequencylabel (e.g. 12C, 5E, ...). Complies to EN 50.248 ([4])

Basis data type	MaxSize
String	3

Frequency

Current tuned frequency.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		16	kHz

SignalQuality

The value describes the quality of the signal. **Note:** The interpretation of this value depends on the implementation.

Note: It is recommend to use only a few quality levels (e.g. 3 levels) which do not change very often (e.g. every 0.5 secondes).

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0	0...100	1	none

2.1.5 Learn (0x220)

Occurrence: Mandatory

Indicates the Learn mode of the DAB tuner. By this function, the internal lists of the DAB Tuner are filled.

This function replaces the old function Learn (FktID=0x200).

Feature: General, Administration

2.1.5.1 Format of Function

Function classes: Unclassified Method

FBlock	Function	OPType	Parameter
DABTuner (0x43)	Learn (0x220)	StartResultAck	SenderHandle, Mode
		AbortAck	SenderHandle
		ErrorAck	SenderHandle, ErrorCode, ErrorInfo
		ProcessingAck	SenderHandle
		ResultAck	SenderHandle

2.1.5.2 Parameter

SenderHandle

The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Mode

Normal Mode means normal operation mode or switch back to normal mode and stay on the last tuned frequency.
Learn Mode means learning operation mode by scanning the frequencies and add information to the internal lists.
Clear Mode means learning operation after clearing the internal lists.
Auto Learn Mode means the learning mode over a frequency range depending on the implementation of the DAB Tuner.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x03	0x00	NormalMode	Normal Mode
		0x01	LearnMode	Learn Mode (Add informations to internal lists)
		0x02	ClearMode	Clear Mode (Clear and Learn afterward
		0x03	AutoLearnMode	Auto Learn Mode

2.1.6 Seek (0x221)

Occurrence: Mandatory

This function replaces the old function Seek (0x201).

Feature: Audio, Administration

2.1.6.1 Format of Function

Function classes: Unclassified Method

FBlock	Function	OPType	Parameter
DABTuner (0x43)	Seek (0x221)	StartResultAck	SenderHandle, SinkHandle, SeekMode, Seekfilter, ScanPlayTime
		AbortAck	SenderHandle
		ErrorAck	SenderHandle, ErrorCode, ErrorInfo
		ProcessingAck	SenderHandle
		ResultAck	SenderHandle

2.1.6.2 Parameter

SenderHandle

The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SeekMode

The DAB-Tuner supports different modes for the seek functionality.

Manual Tune: The parameter "manual tune" allows stepping through the audio services of the current tuned ensemble or through the CEPT frequency grid. If the Seekfilter isn't restricted to the current tuned ensemble, the following behavior is defined: A step to the up/down audio service is performed, when an ensemble is received and the current tuned audio service isn't the last or first audio service of the ensemble. If the first/last audio service is tuned and a down/up manual step is set, than the next CEPT frequency is chosen by the tuner. When the tuner is set on a CEPT frequency without reception of a DAB ensemble, the up/down manual command initiates a step to the next/previous CEPT frequency.

Manual Tune Continuous: The DAB Tuner is running through the CEPT frequency band until an "off"

command is sent.

Manual Tune Preset: The DAB Tuner is changing to the next preset, depending on the direction up/down.

Auto Seek: The parameter "auto seek" allows to step through the audio services of the current tuned ensemble or to start an automatic search to the next tunable ensemble. If the Seekfilter isn't restricted to the current tuned ensemble, the following behavior is defined: A step to the up/down audio service is performed, when an ensemble is received and the current tuned audio service isn't the last or first audio service of the ensemble. If the first/last audio service is tuned and a down/up auto seek is made, then the DAB-Tuner performs an automatic search down/up in the CEPT frequency grid to the next ensemble.

Auto Seek TP: The function "Auto Seek" is extended with a filter criteria for programmes with traffic announcements. This enumeration type is only supported, if the respective optional announcements function is used.

Auto Scan: This function is based on auto seek. During an auto scan, an audio service is played for the playtime and after the playtime the next audio service is automatically tuned depending of the Seekfilter value. The auto scan is stopped by an abort command, than the current tuned audio service will be kept.

Auto Scan Audio Servicelist: The scan is only performed with the audio services which are part of the audio service list.

Auto Scan Audio Presetlist: The scan is only performed with the audio services which are part of the audio preset list.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x12	0x00	Off	Off (In conjunction with a method, please use the OPType "AbortAck")
		0x01	ManualTuneUp	Manual Tune up
		0x02	ManualTuneDown	Manual Tune down
		0x03	ManualTuneUpCont	Manual Tune up Continious, Seek filter not used
		0x04	ManualTuneDownCont	Manual Tune down Continious, Seek filter not used
		0x05	ManualTuneUpPreset	Manual Tune up Audio Preset List, Seek filter not used
		0x06	ManualTuneDownPreset	Manual Tune down Audio Preset List, Seek filter not used
		0x07	ManualTuneUpService	Manual Tune up Audio Servicelist, Seek filter not used
		0x08	ManualTuneDownService	Manual Tune down Audio Servicelist, Seek filter not used
		0x09	AutoSeekUp	Auto seek up
		0x0A	AutoSeekDown	Auto seek down
		0x0B	AutoSeekUpTraffic	Auto seek up Traffic programme
		0x0C	AutoSeekDownTraffic	Auto seek down Traffic programme
		0x0D	AutoSeekScanUp	Auto Scan up, playtime parameter is also set
		0x0E	AutoSeekScanDown	Auto Scan down,

Basis data type	Range of values	Code	Symbolic Name	Description
				playtime parameter is also set
		0x0F	AutoSeekScanUpService	Auto Scan up Audio Servicelist., playtime parameter is also set
		0x10	AutoSeekScanDownService	Auto Scan down Audio Servicelist, playtime parameter is also set
		0x11	AutoSeekScanUpPreset	Auto Scan up Audio Presetlist, playtime parameter is also set
		0x12	AutoSeekScanDownPreset	Auto Scan down Audio Presetlist, playtime parameter is also set

Seekfilter

The Seekfilter defines which audio service component type shall be selected by manual tune, auto seek or auto scan.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x04	0x00	NotUsed	Not used
		0x01	CurrentPrimaryComponents	Primary audio components in the current tuned ensemble
		0x02	CurrentPrimaryAndSecondaryComponents	Primary and secondary audio components in the current tuned ensemble
		0x03	TunablePrimaryComponents	Primary audio components in all tunable ensembles of the supported frequency bands
		0x04	TunablePrimaryAndSecondaryComponents	Primary and secondary audio components in all tunable ensembles of the supported frequency bands

ScanPlayTime

This parameter sets the playtime time at each station. 0xFF value indicates infinite playtime or is set for all other options.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	s

2.1.7 Tune (0x222)

Occurrence: Mandatory

With this function, the frequency or the ensemble is selected. It depends on the implementation, if the first audio service and its primary audio service component are automatically played when an ensemble is selected and an audio service component is available.

Note: The EnsembleState is only changed by the DAB Tuner as long as the SelectionMode Off is not reported by the DAB Tuner.

This function replaces the old function Tune (0x202).

Feature: General, Administration

2.1.7.1 Format of Function

Function classes: Sequence Method

FBlock	Function	OPType	Parameter
DABTuner (0x43)	Tune (0x222)	StartResultAck	SenderHandle, SinkHandle, SelectionMode, Frequency, EnsID, EnsECC
		AbortAck	SenderHandle
		ErrorAck	SenderHandle, ErrorCode, ErrorInfo
		ProcessingAck	SenderHandle
		ResultAck	SenderHandle, EnsembleState

2.1.7.2 Parameter

SenderHandle

The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SelectionMode

The Selection Mode defines the action of the DAB receiver.
Up Frequency, Down Frequency: The DAB receiver steps to the next frequency in the current frequency table.
Select Frequency: The DAB receiver tunes to the frequency which is given by the parameter

Frequency.
Ensemble next, Ensemble previous: The DAB receiver tunes to the next/previous ensemble in the internal ensemble list.
Select Ensemble: The DAB receiver tunes to the ensemble which is given by the parameters EnsID and EnsECC.
Search up, Search down: The DAB receiver performs a frequency search and stops when an ensemble is found.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x08	0x00	Off	Off
		0x01	UpFrequency	Up Frequency
		0x02	DownFrequency	Down Frequency
		0x03	SelectFrequency	Select Frequency
		0x04	EnsembleNext	Ensemble next
		0x05	EnsemblePrevious	Ensemble previous
		0x06	SelectEnsemble	Select Ensemble
		0x07	SearchUp	Search up
		0x08	SearchDown	Search down

Frequency

The frequency is the current frequency based on the unit of one kHz and a step of 16kHz (based on the DAB specification).

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		16	kHz

EnsID

The ensemble identifier (based on the DAB specification).

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsECC

The extended country code of the Ensemble (based on the DAB specification).

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

EnsembleState

The information about the status of the function. In combination of set, the DAB Tuner has to ignore the EnsembleState.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x06	0x00	NotDefined	not defined
		0x01	EnsembleInvalid	Ensemble invalid
		0x02	EnsembleSelectionDone	Ensemble selection done
		0x03	NoNextEnsemble	No next ensemble
		0x04	NoPreviousEnsemble	No previous ensemble
		0x05	WaitingForEnsemble	Waiting for ensemble
		0x06	NoEnsemble	No ensemble

2.1.8 SelectAudioService (0x223)

Occurrence: Mandatory

This function allows the selection of an audio service. As default, the primary audio service component is selected by the DAB Tuner.
Note: For selection of secondary components, use function SelectComponent.
Note: The ServiceState is only changed by the DAB Tuner as long as the SelectionMode Off is not reported by the DAB Tuner.

This function replaces the old function SelectAudioService (0x203).

Feature: Audio, Administration

2.1.8.1 Format of Function

Function classes: Sequence Method

FBlock	Function	OPType	Parameter
DABTuner (0x43)	SelectAudio Service (0x223)	StartResultAck	SenderHandle, SinkHandle, SelectionMode, SID
		AbortAck	SenderHandle
		ErrorAck	SenderHandle, ErrorCode, ErrorInfo
		ProcessingAck	SenderHandle
		ResultAck	SenderHandle, ServiceState

2.1.8.2 Parameter

SenderHandle

The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SelectionMode

The Selection Mode defines what the DABTuner should select.
"Off": If selection is done by the DAB Tuner, "Off" is given as new status.
"First audio service" (resp. "Last audio service") selects the first (resp. last) service of the currently tuned ensemble.
"Next audio service" and "Previous audio service" operate on the currently selected audio service,

which is described by the SinkHandle and is based upon receiver internal service list of all services in the currently tuned ensemble.
"Select SID" selects the service identified by the parameter SID.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x05	0x00	Off	Off
		0x01	FirstAudioService	First audio Service
		0x02	NextAudioService	Next audio Service
		0x03	PreviousAudioService	Previous audio Service
		0x04	LastAudioService	Last audio Service
		0x05	SelectSID	Select SID

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

ServiceState

The information about the result of the function.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x06	0x00	NotDefined	not defined
		0x01	ServiceInvalid	Service invalid
		0x02	ServiceSelectionDone	Service selection done
		0x03	NoNextService	No next service
		0x04	NoPreviousService	No previous service
		0x05	WaitingForService	Waiting for service
		0x06	NoService	No service

2.1.9 SelectAudioComponent (0x224)

Occurrence: Mandatory

This function allows the selection of the primary- and secondary- service components within the currently tuned service. It is also possible to select a service component by its SCIDI. In this case, the service component needs not to be part of the currently tuned service of the tuned ensemble.

This function replaces the old function SelectAudioComponent (0x204).

Note: In the moment the function is used to select audio service components.

Feature: General, Administration

2.1.9.1 Format of Function

Function classes: Sequence Method

FBlock	Function	OPType	Parameter
DABTuner (0x43)	SelectAudio Component (0x224)	StartResultAck	SenderHandle, SinkHandle, SelectionMode, SCIDI
		AbortAck	SenderHandle
		ErrorAck	SenderHandle, ErrorCode, ErrorInfo
		ProcessingAck	SenderHandle
		ResultAck	SenderHandle, ComponentState

2.1.9.2 Parameter

SenderHandle

The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SelectionMode

The SelectionMode defines which component the DABTuner should select.

"Off": If selection is done by the DAB Tuner, "Off" is given as new status.
"Primary Component" selects the primary service component of the currently tuned service.
"Next Component" and "Previous Component" operate on the currently selected service component.
"Select SCIDI" selects the component identified by the parameter SCIDI.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x04	0x00	Off	Off (In conjunction with a method, please use the OPType "AbortAck")
		0x01	PrimaryComponent	Primary Component
		0x02	NextComponent	Next Component
		0x03	PreviousComponent	Previous Component
		0x04	SelectSCIDI	Select SCIDI

SCIDI

Data Service Component ID internal, virtual ID assigned by the tuner, unique within all known ensembles

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

ComponentState

The information about the result of the function.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x07	0x00	NotDefined	not defined
		0x01	ComponentInvalid	Component invalid
		0x02	PrimaryComponent OK	Primary component OK
		0x03	NoPrimary	No primary
		0x04	SecondaryComponent OK	Secondary component OK
		0x05	NoNextComponent	No next component
		0x06	NoPreviousComponent	No previous component
		0x07	WaitingForComponent	Waiting for component

2.1.10 AnnouncementFilter (0x400)

Occurrence: Optional

The function is used to indicate which types of Announcements should be sent (announcement filter) from the DAB Tuner to the MMI (controller) by the function AnnouncementStatus. The function is used in case of the DAB Tuner handles the routing of announcements. The priority management is handled by the function AnnouncementPriority.

Feature: Audio, Announcement

2.1.10.1 Format of Function

Function classes: Sequence Property

FBlock	Function	OPType	Parameter
DABTuner (0x43)	AnnouncementFilter (0x400)	Set	AnnouncementType, NewFlag, AnnouncementSource
		Get	-
		SetGet	AnnouncementType, NewFlag, AnnouncementSource
		Status	AnnouncementType, NewFlag, AnnouncementSource
		Error	ErrorCode, ErrorInfo

2.1.10.2 Parameter

AnnouncementType

The AnnounementTypes which should be transmitted to the MMI is bitcoded by the following table

Basis data type	Bit #	Code	Description
Unsigned Word	Bit 0	False	Alarm off
		True	Alarm on
	Bit 1	False	Road Traffic Flash off
		True	Road Traffic Flash on
	Bit 2	False	Transport flash off
		True	Transport flash on
	Bit 3	False	Warning/service off
		True	Warning/service on
	Bit 4	False	News flash off
		True	News flash on
	Bit 5	False	Area Weather Flash off
		True	Area Weather Flash on
	Bit 6	False	Event Announcement off
		True	Event Announcement on
	Bit 7	False	Special event off
		True	Special event on
	Bit 8	False	Program information on
		True	Program information off
	Bit 9	False	Sport report off
		True	Sport report on
	Bit 10	False	Financial report off
		True	Financial report on
	Bit 11	False	Reserved for future definition

Basis data type	Bit #	Code	Description
		True	Reserved for future definition
	Bit 12	False	Reserved for future definition
		True	Reserved for future definition
	Bit 13	False	Reserved for future definition
		True	Reserved for future definition
	Bit 14	False	Reserved for future definition
		True	Reserved for future definition
	Bit 15	False	Reserved for future definition
		True	Reserved for future definition

NewFlag

Sets the filter with respect to newly introduced announcements.

Basis data type	Bit #	Code	Description
Unsigned Word	Bit 0	False	NewFlag for Alarm off
		True	NewFlag for Alarm on
	Bit 1	False	NewFlag for Road Traffic Flash off
		True	NewFlag for Road Traffic Flash on
	Bit 2	False	NewFlag for Transport flash off
		True	NewFlag for Transport flash on
	Bit 3	False	NewFlag for Warning/service off
		True	NewFlag for Warning/service on
	Bit 4	False	NewFlag for News flash off
		True	NewFlag for News flash on
	Bit 5	False	NewFlag for Area Weather Flash off
		True	NewFlag for Area Weather Flash on
	Bit 6	False	NewFlag for Event Announcement off
		True	NewFlag for Event Announcement on
	Bit 7	False	NewFlag for Special event off
		True	NewFlag for Special event on
	Bit 8	False	NewFlag for Program information on
		True	NewFlag for Program information off
	Bit 9	False	NewFlag for Sport report off
		True	NewFlag for Sport report on
	Bit 10	False	NewFlag for Financial report off
		True	NewFlag for Financial report on
	Bit 11	False	NewFlag for Reserved for future definition
		True	NewFlag for Reserved for future definition
	Bit 12	False	NewFlag for Reserved for future definition
		True	NewFlag for Reserved for future definition
	Bit 13	False	NewFlag for Reserved for future definition
		True	NewFlag for Reserved for future definition
	Bit 14	False	NewFlag for Reserved for future definition
		True	NewFlag for Reserved for future definition
	Bit 15	False	NewFlag for Reserved for future definition
		True	NewFlag for Reserved for future definition

AnnouncementSource

Only Announcements from this AnnounementSources should be transmitted to the MMI. The AnnounementSource is bitcoded by the following table.

Basis data type	Bit #	Code	Description
Unsigned Word	Bit 0	False	Announcement in the current ensemble off
		True	Announcement in the current ensemble on
	Bit 1	False	Announcement in the other ensemble off
		True	Announcement in the other ensemble on
	Bit 2	False	Announcement to FM/RDS off
		True	Announcement to FM/RDS on
	Bit 3	False	Only Announcement from the selected service off
		True	Only Announcement from the selected service on
	Bit 4	False	Reserved for future definition
		True	Reserved for future definition
	Bit 5	False	Reserved for future definition
		True	Reserved for future definition
	Bit 6	False	Reserved for future definition
		True	Reserved for future definition
	Bit 7	False	Reserved for future definition
		True	Reserved for future definition
	Bit 8	False	Reserved for future definition
		True	Reserved for future definition
	Bit 9	False	Reserved for future definition
		True	Reserved for future definition
	Bit 10	False	Reserved for future definition
		True	Reserved for future definition
	Bit 11	False	Reserved for future definition
		True	Reserved for future definition
	Bit 12	False	Reserved for future definition
		True	Reserved for future definition
	Bit 13	False	Reserved for future definition
		True	Reserved for future definition
	Bit 14	False	Reserved for future definition
		True	Reserved for future definition
	Bit 15	False	Reserved for future definition
		True	Reserved for future definition

2.1.11 AnnouncementPriority (0x401)

Occurrence: Optional

The function allows setting a priority for each announcement type. The priority shall be used for the internal priority management of the DAB tuner to interrupt a running announcement with a new incoming announcement that has a higher priority. This function is only needed, if the DAB Tuner performs priority handling. In this case, when a higher priority announcement is selected, a lower priority announcement is not reported to the controller.

Feature: Audio, Announcement

2.1.11.1 Format of Function

Function classes: Array of { Number }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	Announcement Priority (0x401)	Set	Pos, Data
		Get	Pos
		SetGet	Pos, Data
		Status	Pos, Data
		Error	ErrorCode, ErrorInfo

2.1.11.2 Parameter

Pos

The parameter Pos={x,y} consists of two byte x and y and shows which parameter shall be set, inquired or read. Since this is a unidimensional construction, the second byte is unused (y = 0 = const.) and the simplified notation Pos = x is valid. Valid range x = 0 ... 11, y = 0.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Ann_Priority_Alarm, Ann_Priority_RoadTraffic, Ann_Priority_TransportFlash, Ann_Priority_WarningService, Ann_Priority_News, Ann_Priority_AreaWeather, Ann_Priority_Event, Ann_Priority_SpecialEvent, Ann_Priority_ProgramInformation, Ann_Priority_SportReport, Ann_Priority_FinancialReport }
		{ x=1, y=0 }	Ann_Priority_Alarm
		{ x=2, y=0 }	Ann_Priority_RoadTraffic
		{ x=3, y=0 }	Ann_Priority_TransportFlash
		{ x=4, y=0 }	Ann_Priority_WarningService
		{ x=5, y=0 }	Ann_Priority_News

Basis data type	Length	Description	
		{ x=6, y=0 }	Ann_Priority_AreaWeather
		{ x=7, y=0 }	Ann_Priority_Event
		{ x=8, y=0 }	Ann_Priority_SpecialEvent
		{ x=9, y=0 }	Ann_Priority_ProgrammInformation
		{ x=10, y=0 }	Ann_Priority_Sportreport
		{ x=11, y=0 }	Ann_Priority_Financialreport
		{ x=12-16, y=0 }	For future use

Ann_Priority_XX

Priority for an Announcement Type: 0 highest priority value, 255 lowest priority value.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

2.1.12 AnnouncementStatus (0x402)

Occurrence: Optional

The function signals that a specific Announcement is available.

Feature: Audio, Announcement

2.1.12.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Boolean Enumeration Number } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	Announcement Status (0x402)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.12.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, SID, AnnoucementType, Status, PI-Code }
		{ x=Tag, y=0 }	Tag[Tag], SID[Tag], AnnoucementType[Tag], Status[Tag], PI-Code[Tag]
		{ x=Tag, y=2 }	SID[Tag]
		{ x=Tag, y=3 }	AnnoucementType[Tag]
		{ x=Tag, y=4 }	Status[Tag]
		{ x=Tag, y=5 }	PI-Code[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

AnnouncementType

The AnnounementTypes which should be transmitted to the MMI is bitcoded by the following table.

Basis data type	Bit #	Code	Description
Unsigned Word	Bit 0	False	Alarm off
		True	Alarm on
	Bit 1	False	Road Traffic Flash off
		True	Road Traffic Flash on
	Bit 2	False	Transport flash off
		True	Transport flash on
	Bit 3	False	Warning/service off
		True	Warning/service on
	Bit 4	False	News flash off
		True	News flash on
	Bit 5	False	Area Weather Flash off
		True	Area Weather Flash on
	Bit 6	False	Event Announcement off
		True	Event Announcement on
	Bit 7	False	Special event off
		True	Special event on
	Bit 8	False	Program information on
		True	Program information off
	Bit 9	False	Sport report off
		True	Sport report on
	Bit 10	False	Financial report off
		True	Financial report on

Basis data type	Bit #	Code	Description
	Bit 11	False	Reserved for future definition
		True	Reserved for future definition
	Bit 12	False	Reserved for future definition
		True	Reserved for future definition
	Bit 13	False	Reserved for future definition
		True	Reserved for future definition
	Bit 14	False	Reserved for future definition
		True	Reserved for future definition
	Bit 15	False	Reserved for future definition
		True	Reserved for future definition

Status

Refers to all announcements which are signalled by the parameters SID and AnnouncementType before.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x03	0x00	AnnouncementOff	Announcement off
		0x01	AnnouncementCurrentEnsembleOn	Announcement in the Current Ensemble on
		0x02	AnnouncementOtherEnsembleOn	Announcement to Other Ensemble on (references to the other ensemble which delivers the announcement)
		0x03	AnnouncementFMRDSOn	Announcement to FM/RDS on

PICode

PI-Code of the linked FM-RDS Programme. PI-Code = 0x0000 shall be invalid. Switching to FM/RDS and back to DAB is controlled over the Status.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

2.1.13 AnnouncementSwitch (0x403)

Occurrence: Optional

The function is used to switch on/off an announcement or to request the status of the selected announcement in the DAB Tuner with respect to the SinkHandle. For replacing an active announcement by a new announcement with a higher priority, it shall be sufficient to switch on the new announcement type for the required SinkHandle.

Feature: Audio, Announcement

2.1.13.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Boolean Enumeration } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	Announcement Switch (0x403)	Set	Tag, PosY, Data
		Get	Tag, PosY
		SetGet	Tag, PosY, Data
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.13.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, SID, AnnouncementType, Status }
		{ x=Tag, y=0 }	Tag[Tag], SID[Tag], AnnouncementType[Tag], Status[Tag]
		{ x=Tag, y=2 }	SID[Tag]
		{ x=Tag, y=3 }	AnnouncementType[Tag]
		{ x=Tag, y=4 }	Status[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

AnnouncementType

The AnnounementType is bitcoded by the following table.

Basis data type	Bit #	Code	Description
Unsigned Word	Bit 0	False	Alarm off
		True	Alarm on
	Bit 1	False	Road Traffic Flash off
		True	Road Traffic Flash on
	Bit 2	False	Transport flash off
		True	Transport flash on
	Bit 3	False	Warning/service off
		True	Warning/service on
	Bit 4	False	News flash off
		True	News flash on
	Bit 5	False	Area Weather Flash off
		True	Area Weather Flash on
	Bit 6	False	Event Announcement off
		True	Event Announcement on
	Bit 7	False	Special event off
		True	Special event on
	Bit 8	False	Program information on
		True	Program information off
	Bit 9	False	Sport report off
		True	Sport report on
	Bit 10	False	Financial report off

Basis data type	Bit #	Code	Description
		True	Financial report on
	Bit 11	False	Reserved for future definition
		True	Reserved for future definition
	Bit 12	False	Reserved for future definition
		True	Reserved for future definition
	Bit 13	False	Reserved for future definition
		True	Reserved for future definition
	Bit 14	False	Reserved for future definition
		True	Reserved for future definition
	Bit 15	False	Reserved for future definition
		True	Reserved for future definition

Status

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x01	0x00	AnnouncementOff	Announcement off
		0x01	AnnouncementOn	Announcement on

2.1.14 DRCSwitch (0x410)

Occurrence: Optional

Switching between DRC States.

Feature: Audio, Dynamic Range Control

2.1.14.1 Format of Function

Function classes: DynamicArray of { Record of { Number Enumeration } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	DRCSwitch (0x410)	Set	Tag, PosY, Data
		Get	Tag, PosY
		SetGet	Tag, PosY, Data
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.14.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, DRCOnOff }
		{ x=Tag, y=0 }	Tag[Tag], DRCOnOff[Tag]
		{ x=Tag, y=2 }	DRCOnOff[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

DRCOnOff

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x01	0x00	DRCSwitchedOff	DRC switched off
		0x01	DRCSwitchedOn	DRC switched on

2.1.15 CurrentRadiotext (0x420)

Occurrence: Optional

Radiotext of the current selected ServiceComponent(s). An empty string deletes the current displayed radiotext.

Feature: Audio, Radiotext

2.1.15.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number String } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	CurrentRadiotext (0x420)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.15.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, SCIDI, Radiotext }
		{ x=Tag, y=0 }	Tag[Tag], SCIDI[Tag], Radiotext[Tag]
		{ x=Tag, y=2 }	SCIDI
		{ x=Tag, y=3 }	Radiotext

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SCIDI

Data Service Component ID internal, virtual ID assigned by the tuner, unique within all known ensembles

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

RadioText

Radiotext. Symbolic charcters are allowed.

Basis data type	MaxSize
String	128

2.1.16 FrequencyTable (0x430)

Occurrence: Optional

Determines the used frequency table.

Feature: General, Adminstration

2.1.16.1 Format of Function

Function classes: Enumeration

FBlock	Function	OPType	Parameter
DABTuner (0x43)	FrequencyTable (0x430)	Set	FrequencyTable
		Get	-
		SetGet	FrequencyTable
		Status	FrequencyTable
		Error	ErrorCode, ErrorInfo

2.1.16.2 Parameter

FrequencyTable

Definition of the frequency table to be used.

0x00 correponds to a mode whereby the DAB receiver determines automatically which frequency table to be used
0x01 corresponds to frequencies 5A to 13F and LA to LW, as defined in [4], table A.1
0x02 corresponds to frequencies LA to LW, as defined in [4], table A.1
0x03 corresponds to frequencies 5A to 13F, as defined in [4], table A.1
0x04 corresponds to frequencies 1 to 23, as defined in [4], table A.2

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x04	0x00	FrequencyTableAuto	Frequency Table Auto
		0x01	FrequencyTableEuropeBoth	Frequency Table Europe (both)
		0x02	FrequencyTableEuropeLBand	Frequency Table Europe L-Band
		0x03	FrequencyTableEuropeVHFBand	Frequency Table Europe VHF-Band
		0x04	FrequencyTableCanada	Frequency Table Canada

2.1.17 GetSinkHandle (0x440)

Occurrence: Optional

With this method, a controller (e.g. a MMI) asks for the SourceNr the DAB tuner will use to stream synchronous data onto the MOST network. This SourceNr is referred to as SinkHandle.

Background: Consider a car multimedia system which consists of several MMIs and one DAB receiver which has several source ports with different SourceNrs. Each MMI is able to control the DAB receiver. Let's say, that the audio stream (e.g. program Bayern 1) which is selected by the MMI 1 is output on the SourceNr 0x01 and the audio stream (e.g. program Bayern 2) which is selected by the MMI 2 is output on the SourceNr 0x03. If the user selects another audio stream (e.g. program Bayern 3) on the MMI 1, the DAB receiver does not know the SourceNr on which it has to output the audio stream, unless the MMI 1 tells the DAB receiver what SourceNr to use. Therefore, the MMI 1 has to carry the relevant SourceNr in the parameter list of the function which initiates the program change. A MMI first asks for the respective SourceNr which is in the following carried in the parameter list of the functions which refer to the output of synchronous data.

Note: This method is used to implement a dynamic assignment of the SourceNrs to the MMIs. However, a fixed assignment of SourceNrs and MMIs can be used instead of the dynamic administration, thus avoiding the usage of the method GetSinkHandle.

Feature: General, Administration

2.1.17.1 Format of Function

Function classes: Unclassified Method

FBlock	Function	OPType	Parameter
DABTuner (0x43)	GetSinkHandle (0x440)	StartResultAck	SenderHandle
		AbortAck	SenderHandle
		ErrorAck	SenderHandle, ErrorCode, ErrorInfo
		ProcessingAck	SenderHandle
		ResultAck	SenderHandle, SinkHandle
		Error	ErrorCode, ErrorInfo

2.1.17.2 Parameter

SenderHandle

The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

2.1.18 ReleaseSinkHandle (0x441)

Occurrence: Optional

With this method, a controller (e.g. a MMI) releases the SinkHandle

Feature: General, Administration

2.1.18.1 Format of Function

Function classes: Unclassified Method

FBlock	Function	OPType	Parameter
DABTuner (0x43)	ReleaseSink Handle (0x441)	StartResultAck	SenderHandle, SinkHandle
		AbortAck	SenderHandle, SinkHandle
		ErrorAck	SenderHandle, ErrorCode, ErrorInfo
		ProcessingAck	SenderHandle
		ResultAck	SenderHandle, SinkHandle
		Error	ErrorCode, ErrorInfo

2.1.18.2 Parameter

SenderHandle

The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

2.1.19 ServiceFollowingMode (0x450)

Occurrence: Optional

Function to enable the automatic switching between DAB services. The DAB Tuner will try to follow the DAB service with respect to the enabled strategies.

2.1.19.1 Format of Function

Function classes: Record of { Boolean Boolean Boolean Boolean Boolean Boolean }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	ServiceFollowing Mode (0x450)	Get	Pos
		Status	Pos, Data
		Error	ErrorCode, ErrorInfo

2.1.19.2 Parameter

Pos

The parameter Pos consists of one byte that shows what is to be set or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Data

Basis data type	Length	Description	
Record	-	Pos	Data
		{ x=0, y=0 }	AltFrequency, AltEnsemble, AltHardLinking, AltSearchSID, AltSoftLinking, AltSearch

AltFrequency

Use alternative frequencies to find the service in identically ensembles (switch frequency, same audio)

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	on
		False	off

AltEnsemble

Use alternative ensembles to find the service in another ensemble (go to another ensemble, same audio)

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	on
		False	off

AltHardLinking

Use hard linked services e.g. different broadcast stations send the equal service at night (go to another ensemble, play service with different Sid, same audio)

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	on
		False	off

AltSearchSID

Use searching (initiate searches to find the SID), the result is the same audio.

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	on
		False	off

AltSoftLinking

Use soft linked services which results in an other audio channel/programme.

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	on
		False	off

AltSearch

Use searching to find a programme with the same characteristics like announcement support or programme type (other audio).

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	on
		False	off

2.1.20 SetPTYFilter (0x451)

Occurrence: Optional

This function is used to set up a general PTY filter. This filter can be used for a search (Tune, Seek, Scan). One record entry describes one filter.

Feature: Audio, Information

2.1.20.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Classified Stream } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	SetPTYFilter (0x451)	Set	Tag, PosY, Data
		Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.20.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, PTYTypeMask, PTyCodes }
		{ x=Tag, y=0 }	Tag[Tag], PTYTypeMask[Tag], PTyCodes[Tag]
		{ x=Tag, y=2 }	PTYTypeMask [Tag]
		{ x=Tag, y=3 }	PTyCodes [Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PTYTypeMask

The PTYTypeMask describes how to interpret the PTYCodes. Bit 0 corresponds to Byte 0 in the Parameter PTYCodes and so on.

Bit 0: International Code
Bit 1: International Code Dynamic
Bit 2: Coarse Code
Bit 3: Coarse Code Dynamic
Bit 4: Fine Code
Bit 5: Fine Code Dynamic
Bit 6: Second Fine Code
Bit 7: Second Fine Code Dynamic

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

PTYCodes

Stream of PTY codes in order of the PTYTypeMask. E.g. if only 2 PTYTypes are avialable, only this 2 codes will be transmitted. PtyCodes are regional dependent, regional information (country Id and ECC OR international table) can be derived from a property which the system master should contain (is not implemented in a protocol yet). Only the codes are transferred, not the labels.

Basis data type	Length	Media type
Classified Stream	8	application/octet-stream

2.1.21 PresetSave (0x460)

Occurrence: Optional

Save the current service on the position PresetNumber in the list which is given by the FktID. The PresetNumber corresponds to the Tag in the preset list.

Feature: General, Adminstration

2.1.21.1 Format of Function

Function classes: Sequence Property

FBlock	Function	OPType	Parameter
DABTuner (0x43)	PresetSave (0x460)	Set	SinkHandle, FktID, PresetNumber
		Error	ErrorCode, ErrorInfo

2.1.21.2 Parameter

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

FktID

Function ID.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0	0...4095	1	none

PresetNumber

PresetNumber corresponds to the Tag in the Function which is given by the FktID.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

2.1.22 PresetSelect (0x461)

Occurrence: Optional

Select the service which is stored on the position PresetNumber in the list which is given by the FktID. The PresetNumber is corresponding to the Tag in the preset list.

Feature: General, Adminstration

2.1.22.1 Format of Function

Function classes: Sequence Property

FBlock	Function	OPType	Parameter
DABTuner (0x43)	PresetSelect (0x461)	Set	SinkHandle, FktID, PresetNumber
		Error	ErrorCode, ErrorInfo

2.1.22.2 Parameter

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

FktID

Function ID.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0	0...4095	1	none

PresetNumber

PresetNumber corresponds to the Tag in the Function which is given by the FktID.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

2.1.23 AudioPresets (0x462)

Occurrence: Optional

List with information about the presets.

Feature: General, Administration

2.1.23.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Boolean Number String Enumeration } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	AudioPresets (0x462)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.23.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag is corresponding to the PresetNumber.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, SID, LabelStatus, Label8CharAbbreviation, Label, SelectionStatus }
		{ x=Tag, y=0 }	Tag[Tag], SID[Tag], LabelStatus[Tag], Label8CharAbbreviation[Tag], Label[Tag], SelectionStatus[Tag]
		{ x=Tag, y=2 }	SID[Tag]
		{ x=Tag, y=3 }	LabelStatus[Tag]
		{ x=Tag, y=4 }	Label8CharAbbreviation[Tag]
		{ x=Tag, y=5 }	Label[Tag]
		{ x=Tag, y=6 }	SelectionStatus[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag is corresponding to the PresetNumber.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

LabelStatus

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	Label valid
		False	Label not valid

Label8CharAbbreviation

Bitfield which is added with the Label to get an abbreviation. This is a Bitmask from DAB please see Spec [1].

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Label

It shall be coded as a string of 16 characters

Basis data type	MaxSize
String	16

SelectionStatus

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x01	0x00	NotActive	not active
		0x01	Active	active

2.1.24 LearnStatus (0x470)

Occurrence: Optional

With this function the current Learn status can be retrieved.

Feature: General, Administration

2.1.24.1 Format of Function

Function classes: Enumeration

FBlock	Function	OPType	Parameter
DABTuner (0x43)	LearnStatus (0x470)	Get	-
		Status	Mode
		Error	ErrorCode, ErrorInfo

2.1.24.2 Parameter

Mode

Normal Mode means normal operation mode or switch back to normal mode and stay on the last tuned frequency.
Learn Mode means learning operation mode by scanning the frequencies and add information to the internal lists.
Clear Mode means learning operation after clearing the internal lists.
Auto Learn Mode means the learning mode over a frequency range depending on the implementation of the DAB Tuner.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x03	0x00	NormalMode	Normal Mode
		0x01	LearnMode	Learn Mode (Add information to internal lists)
		0x02	ClearMode	Clear Mode (Clear and Learn afterwards)
		0x03	AutoLearnMode	Auto Learn Mode

2.1.25 SeekStatus (0x471)

Occurrence: Optional

With this function the current SeekStatus can be retrieved.

Feature: Audio, Administration

2.1.25.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
DABTuner (0x43)	SeekStatus (0x471)	Get	SinkHandle
		Status	SinkHandle, SeekMode
		Error	ErrorCode, ErrorInfo

2.1.25.2 Parameter

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SeekMode

The DAB-Tuner supports different modes for the seek functionality.

Manual Tune: The parameter "manual tune" allows stepping through the audio services of the current tuned ensemble or through the CEPT frequency grid. If the Seekfilter isn't restricted to the current tuned ensemble, the following behavior is defined: A step to the up/down audio service is performed, when an ensemble is received and the current tuned audio service isn't the last or first audio service of the ensemble. If the first/last audio service is tuned and a down/up manual step is set, than the next CEPT frequency is chosen by the tuner. When the tuner is set on a CEPT frequency without reception of a DAB ensemble, the up/down manual command initiates a step to the next/previous CEPT frequency.

Manual Tune Continuous: The DAB Tuner is running through the CEPT frequency band until an "off" command is sent.

Manual Tune Preset: The DAB Tuner is changing to the next preset, depending on the direction up/down.

Auto Seek: The parameter "auto seek" allows to step through the audio services of the current tuned ensemble or to start an automatic search to the next tunable ensemble. If the Seekfilter isn't restricted to the current tuned ensemble, the following behavior is defined: A step to the up/down audio service is performed, when an ensemble is received and the current tuned audio service isn't the last or first audio service of the ensemble. If the first/last audio service is tuned and a down/up auto seek is made, then the DAB-Tuner performs an automatic search down/up in the CEPT frequency grid to the next ensemble.

Auto Seek TP: The function "Auto Seek" is extended with a filter criteria for programmes with traffic announcements. This enumeration type is only supported, if the respective optional announcements function is used.

Auto Scan: This function is based on auto seek. During an auto scan, an audio service is played for the playtime and after the playtime the next audio service is automatically tuned depending of the Seekfilter value. The auto scan is stopped by an abort command, than the current tuned audio service will be kept.

Auto Scan Audio Servicelist: The scan is only performed with the audio services which are part of the audio service list.

Auto Scan Audio Presetlist: The scan is only performed with the audio services which are part of the audio preset list.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x12	0x00	Off	Off (In conjunction with a method, please use the OPType "AbortAck")
		0x01	ManualTuneUp	Manual Tune up
		0x02	ManualTuneDown	Manual Tune down
		0x03	ManualTuneUpCont	Manual Tune up Continuous, Seekfilter not used
		0x04	ManualTuneDownCont	Manual Tune down Continuous, Seekfilter not used
		0x05	ManualTuneUpPreset	Manual Tune up Audio Preset List, Seekfilter not used
		0x06	ManualTuneDownPreset	Manual Tune down Audio Preset List, Seekfilter not used
		0x07	ManualTuneUpService	Manual Tune up Audio Servicelist, Seek filter not used
		0x08	ManualTuneDownService	Manual Tune down Audio Servicelist, Seekfilter not used
		0x09	AutoSeekUp	Auto seek up
		0x0A	AutoSeekDown	Auto seek down
		0x0B	AutoSeekUpTP	Auto seek up Traffic programme
		0x0C	AutoSeekDownTP	Auto seek down Traffic programme
		0x0D	AutoScanUp	Auto Scan up, playtime parameter is also set
		0x0E	AutoScanDown	Auto Scan down, playtime parameter is also set
		0x0F	AutoScanUpService	Auto Scan up Audio Servicelist., playtime parameter is also set
		0x10	AutoScanDownService	Auto Scan down Audio Servicelist, playtime parameter is also set
		0x11	AutoScanUpPreset	Auto Scan up Audio Presetlist, playtime parameter is also set
		0x12	AutoScanDownPreset	Auto Scan down Audio Presetlist, playtime parameter is also set

2.1.26 TuneStatus (0x472)

Occurrence: Optional

With this function the current TuneStatus can be retrieved.

Feature: General, Administration

2.1.26.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
DABTuner (0x43)	TuneStatus (0x472)	Get	SinkHandle
		Status	SinkHandle, SelectionMode
		Error	ErrorCode, ErrorInfo

2.1.26.2 Parameter

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SelectionMode

The Selection Mode defines the action of the DAB receiver.

Up Frequency, Down Frequency: The DAB receiver steps to the next frequency in the current frequency table.
Select Frequency: The DAB receiver tunes to the frequency which is given by the parameter Frequency.
Ensemble next, Ensemble previous: The DAB receiver tunes to the next/previous ensemble in the internal ensemble list.
Select Ensemble: The DAB receiver tunes to the ensemble which is given by the parameters EnsID and EnsECC.
Search up, Search down: The DAB receiver performs a frequency search and stops when an ensemble is found.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x08	0x00	Off	Off (In conjunction with a method, please use the OPType "AbortAck")
		0x01	UpFrequency	Up Frequency
		0x02	DownFrequency	Down Frequency
		0x03	SelectFrequency	Select Frequency
		0x04	EnsembleNext	Ensemble next
		0x05	EnsemblePrevious	Ensemble previous
		0x06	SelectEnsamble	Select Ensamble
		0x07	SearchUp	Search up
		0x08	SearchDown	Search down

2.1.27 SelectAudioServiceStatus (0x473)

Occurrence: Optional

With this function the current SelectAudioServiceStatus can be retrieved

Feature: Audio, Administration

2.1.27.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
DABTuner (0x43)	SelectAudio ServiceStatus (0x473)	Get	SinkHandle
		Status	SinkHandle, SelectionMode
		Error	ErrorCode, ErrorInfo

2.1.27.2 Parameter

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SelectionMode

The Selection Mode defines what the DABTuner should select.

"Off": If selection is done by the DAB Tuner, "Off" is given as new status.
"First audio service" (resp. "Last audio service") selects the first (resp. last) service of the currently tuned ensemble.
"Next audio service" and "Previous audio service" operate on the currently selected audio service, which is described by the SinkHandle and is based upon receiver internal service list of all services in the currently tuned ensemble.
"Select SID" selects the service identified by the parameter SID.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x05	0x00	Off	Off (In conjunction with a method, please use the OPType "Abort")
		0x01	FirstAudioService	First audio Service
		0x02	NextAudioService	Next audio Service
		0x03	PreviousAudioService	Previous audio Service
		0x04	LastAudioService	Last audio Service
		0x05	SelectSID	Select SID

2.1.28 SelectAudioComponentStatus (0x474)

Occurrence: Optional

With this function the current SelectAudioComponentStatus can be retrieved

Feature: General, Administration

2.1.28.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
DABTuner (0x43)	SelectAudio ComponentStatus (0x474)	Get	SinkHandle
		Status	SinkHandle, SelectionMode
		Error	ErrorCode, ErrorInfo

2.1.28.2 Parameter

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SelectionMode

The SelectionModeComponent defines what the DABDevice should select.
"Off": If selection is done by the DAB Tuner, "Off" is given as new status.
"Primary Component" selects the primary service component of the currently tuned service.
"Next Component" and "Previous Component" operate on the currently selected service component.
"Select SCIDI" selects the component identified by the parameter SCIDI.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x04	0x00	Off	Off (In conjunction with a method, please use the OPType "Abort")
		0x01	PrimaryComponent	Primary Component
		0x02	NextComponent	Next Component
		0x03	PreviousComponent	Previous Component
		0x04	SelectSCIDI	Select SCIDI

2.1.29 EnsembleList (0x500)

Occurrence: Optional

List with general informations (Name, Shortname, Number of Audio and DataServices, etc.) about known ensembles.

The function can be used as a DynamicArray or as a MotherArray for an ArrayWindow. For this case the FktIDs 0x501 to 0x50F are reserved for possible instance-handles.

Feature: General, Information

2.1.29.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Boolean Number String Boolean Number Number } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	EnsembleList (0x500)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.29.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, EnsID, EnsECC, LabelStatus, Label8CharAbbreviation, Label, AlarmFlag, AudioServices, DataServices }
		{ x=Tag, y=0 }	Tag[Tag], EnsID[Tag], EnsECC[Tag], LabelStatus[Tag], Label8CharAbbreviation [Tag], Label[Tag], AlarmFlag[Tag], AudioServices[Tag], DataServices[Tag]
		{ x=Tag, y=2 }	EnsID[Tag]
		{ x=Tag, y=3 }	EnsECC[Tag]
		{ x=Tag, y=4 }	LabelStatus[Tag]

Basis data type	Length	Description	
		{ x=Tag, y=5 }	Label8CharAbbreviation[Tag]
		{ x=Tag, y=6 }	Label[Tag]
		{ x=Tag, y=7 }	AlarmFlag[Tag]
		{ x=Tag, y=8 }	AudioServices[Tag]
		{ x=Tag, y=9 }	DataServices[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsID

Ensemble identifier

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsECC

Extended country code of the Ensemble.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

LabelStatus

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	Label valid
		False	Label not valid

Label8CharAbbreviation

Bitfield which is added with the Label to get an abbreviation. This is a Bitmask from DAB please see Spec [1].

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Label

It shall be coded as a string of 16 characters

Basis data type	MaxSize
String	16

AlarmFlag

Alarm messages accessible in this ensemble (see [1], [2]).

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	AlarmFlag on
		False	AlarmFlag off

AudioServices

Number of audio services in the ensemble; 0xFF if the number is unknown.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

DataServices

Number of data services in the ensamble; 0xFF if the number is unknow.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

2.1.30 AudioServiceList (0x510)

Occurrence: Optional

List with general information about the known audio services. The function can be used as a DynamicArray or as a MotherArray for an ArrayWindow. For this case the FktIDs 0x511 to 0x51F are reserved for possible instance-handles.

Feature: Audio, Information

2.1.30.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Number Boolean Number String Boolean Number Classified Stream } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	AudioServiceList (0x510)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.30.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, EnsID, EnsECC, SID, LabelStatus, Label8CharAbbreviation, Label, AnnouncementSupport, PTYTypeMask, PTYCodes }
		{ x=Tag, y=0 }	Tag[Tag], EnsID[Tag], EnsECC[Tag], SID[Tag], LabelStatus[Tag], Label8CharAbbreviation[Tag], Label[Tag], AnnouncementSupport[Tag], PTYTypeMask[Tag], PTYCodes[Tag]
		{ x=Tag, y=2 }	EnsID[Tag]
		{ x=Tag, y=3 }	EnsECC[Tag]

Basis data type	Length	Description	
		{ x=Tag, y=4 }	SID[Tag]
		{ x=Tag, y=5 }	LabelStatus[Tag]
		{ x=Tag, y=6 }	Label8CharAbbreviation[Tag]
		{ x=Tag, y=7 }	Label[Tag]
		{ x=Tag, y=8 }	AnnouncementSupport[Tag]
		{ x=Tag, y=9 }	PTYTypeMask[Tag]
		{ x=Tag, y=10 }	PTYCodes[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsID

Ensemble identifier

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsECC

Extended country code of the Ensemble

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

LabelStatus

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	Label valid
		False	Label not valid

Label8CharAbbreviation

Bitfield which is added with the Label to get an abbreviation. This is a Bitmask from DAB please see Spec [1].

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Label

It shall be coded as a string of 16 characters

Basis data type	MaxSize
String	16

AnnouncementSupport

Supported Announcements by the service.

Basis data type	Bit #	Code	Description
Unsigned Word	Bit 0	False	Alarm not supported
		True	Alarm supported
	Bit 1	False	Road Traffic Flash not supported
		True	Road Traffic Flash supported
	Bit 2	False	Transport Flash not supported
		True	Transport Flash supported
	Bit 3	False	Warning/service not supported
		True	Warning/service supported
	Bit 4	False	News flash not supported
		True	News flash supported
	Bit 5	False	Area Weather Flash not supported
		True	Area Weather Flash supported
	Bit 6	False	Event Announcement not supported
		True	Event Announcement supported
	Bit 7	False	Special event not supported
		True	Special event supported
	Bit 8	False	Program information supported
		True	Program information not supported
	Bit 9	False	Sport report not supported
		True	Sport report supported
	Bit 10	False	Financial report not supported
		True	Financial report supported
	Bit 11	False	Not used
		True	Not used
	Bit 12	False	Not used
		True	Not used
	Bit 13	False	Not used
		True	Not used
	Bit 14	False	Not used
		True	Not used

Basis data type	Bit #	Code	Description
	Bit 15	False	Not used
		True	Not used

PTYTypeMask

The PTYTypeMask describes how to interpret the PTYCodes. Bit 0 corresponds to Byte 0 in the Parameter PTYCodes and so on.

Bit 0: International Code
Bit 1: International Code Dynamic
Bit 2: Coarse Code
Bit 3: Coarse Code Dynamic
Bit 4: Fine Code
Bit 5: Fine Code Dynamic
Bit 6: Second Fine Code
Bit 7: Second Fine Code Dynamic

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

PTYCodes

Stream of PTY codes in order of the PTYTypeMask. E.g. if only 2 PTYTypes are avialable, only this 2 codes will be transmitted. PtyCodes are regional dependent, regional information (country Id and ECC OR international table) can be derived from a property which the system master should contain (is not implemented in a protocol yet). Only the codes are transferred, not the labels.

Basis data type	Length	Media type
Classified Stream	8	application/octet-stream

2.1.31 AudioComponentList (0x520)

Occurrence: Optional

List with general information about the currently known audio service components. The function can be used as a DynamicArray or as a MotherArray for an ArrayWindow. For this case the FktIDs 0x521 to 0x52F are reserved for possible instance-handles.

Feature: Audio, Information

2.1.31.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Number Number Boolean Number String Number Number Number } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	AudioComponentList (0x520)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.31.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, EnsID, EnsECC, SID, SCIDI, LabelStatus, Label8CharAbbreviation, Label, Language, Flags, ASCTy }
		{ x=Tag, y=0 }	Tag[Tag], EnsID[Tag], EnsECC[Tag],SID[Tag], SCIDI[Tag], LabelStatus[Tag], Label8CharAbbreviation[Tag], Label[Tag], Language[Tag], Flags[Tag], ASCTy[Tag]
		{ x=Tag, y=2 }	EnsID[Tag]
		{ x=Tag, y=3 }	EnsECC[Tag]
		{ x=Tag, y=4 }	SID[Tag]
		{ x=Tag, y=5 }	SCIDI[Tag]

Basis data type	Length	Description	
		{ x=Tag, y=6 }	LabelStatus[Tag]
		{ x=Tag, y=7 }	Label8CharAbbreviation[Tag]
		{ x=Tag, y=8 }	Label[Tag]
		{ x=Tag, y=9 }	Language[Tag]
		{ x=Tag, y=10 }	Flags[Tag]
		{ x=Tag, y=11 }	ASCTy[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsID

Ensemble identifier

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsECC

Extended country code of the Ensemble

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

SCIDI

Data Service Component ID internal, virtual ID assigned by the tuner, unique within all known ensembles

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

LabelStatus

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	Label valid
		False	Label not valid

Label8CharAbbreviation

Bitfield which is added with the Label to get an abbreviation. This is a Bitmask from DAB please see Spec [1].

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Label

It shall be coded as a string of 16 characters

Basis data type	MaxSize
String	16

Language

Service component language.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Flags

Additional information on the service component.
Bit0 False = Primary component
True = Secondary component
Bit1 False = CA not valid (see [1], [2])
True = CA valid (see [1], [2])
Bit2-7 Not used

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

ASCTy

Audio Service Component Type (foreground, background, multi-channel audio).

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

2.1.32 CurrentDataService (0x711)

Occurrence: Optional

List with general information about the tuned data service identified by the SinkHandle.

Feature: Data, Information

2.1.32.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Boolean Number String Number } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	CurrentDataService (0x711)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.32.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, SIDType, SID, LabelStatus, Label8CharAbbreviation, Label, SCAID }
		{ x=Tag, y=0 }	Tag[Tag], SIDType[Tag], SID[Tag], LabelStatus[Tag], Label8CharAbbreviation[Tag], Label[Tag], SCAID[Tag]
		{ x=Tag, y=2 }	SIDType[Tag]
		{ x=Tag, y=3 }	SID[Tag]
		{ x=Tag, y=4 }	LabelStatus[Tag]
		{ x=Tag, y=5 }	Label8CharAbbreviation[Tag]
		{ x=Tag, y=6 }	Label[Tag]
		{ x=Tag, y=7 }	SCAID[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SIDType

Programme or data service structure of SID (e.g., derived from P/D-Flag of the FIG 0 field, see [1], [2]): SID-Type: 0x01 Programme Service 0x02 Data Service

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

LabelStatus

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	Label valid
		False	Label not valid

Label8CharAbbreviation

Bitfield which is added with the Label to get an abbreviation. This is a Bitmask from DAB please see Spec [1].

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Label

It shall be coded as a string of 16 characters

Basis data type	MaxSize
String	16

SCAID

Conditional Access Indentifier (see [1], [2]).

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

2.1.33 CurrentDataComponent (0x712)

Occurrence: Optional

List with general information about the selected data service component.

Feature: Data Information

2.1.33.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Number Boolean Number String Number Number Enumeration Number Boolean } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	CurrentDataComponent (0x712)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.33.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, SIDType, SID, SCIDI, LabelStatus, Label8CharAbbreviation, Label, Language, DSCTy, TMID, SCIDS, SCCA-Flag }
		{ x=Tag, y=0 }	Tag[Tag], SIDType[Tag], SID[Tag], SCIDI[Tag], LabelStatus[Tag], Label8CharAbbreviation[Tag], Label[Tag], Language[Tag], DSCTy[Tag], TMID[Tag], SCIDS[Tag], SCCA-Flag[Tag]
		{ x=Tag, y=2 }	SIDType[Tag]
		{ x=Tag, y=3 }	SID[Tag]
		{ x=Tag, y=4 }	SCIDI[Tag]
		{ x=Tag, y=5 }	LabelStatus[Tag]
		{ x=Tag, y=6 }	Label8CharAbbreviation[Tag]

Basis data type	Length	Description	
		{ x=Tag, y=7 }	Label[Tag]
		{ x=Tag, y=8 }	Language[Tag]
		{ x=Tag, y=9 }	DSCTy[Tag]
		{ x=Tag, y=10 }	TMID[Tag]
		{ x=Tag, y=11 }	SCIDS[Tag]
		{ x=Tag, y=12 }	SCCA-Flag[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SIDType

Programme or data service structure of SID (e.g. derived from P/D-Flag of the FIG 0 field, see [1], [2]): SID-Type: 0x01 Programme Service 0x02 Data Service

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

SCIDI

Data Service Component ID internal, virtual ID assigned by the tuner, unique within all known ensembles

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

LabelStatus

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	Label valid
		False	Label not valid

Label8CharAbbreviation

Bitfield which is added with the Label to get an abbreviation. This is a Bitmask from DAB please see Spec [1].

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Label

It shall be coded as a string of 16 characters

Basis data type	MaxSize
String	16

Language

ServiceComponentLanguage

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

DSCTy

Combines Data Service Component Type (DSCTy) and extension (ExtDSCTy) (see [1], [2]) Bit 0-5: Service Component Type Bit 6-15: Service Component Type extension

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

TmID

Transport mechanism identifier.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x03	0x00	StreamModeAudio	stream mode audio
		0x01	StreamModeData	stream mode data
		0x02	FIDC	FIDC
		0x03	PacketModeData	packet mode data

SCIDS

Combines Service Component ID (packet mode only) and Service component ID within the service (see [1], [2])
Bit 0-3: Service component ID within the service, range 0-15 (all modes)
Bit 4-15: Service Component ID (packet mode only), unique within the ensemble

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SCCA-Flag

This Flag indicates wether or not Service Component Conditional Access (SCCA) is applied.

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	SCCA applied
		False	SCCA not applied

2.1.34 CurrentDataUserApp (0x713)

Occurrence: Optional

List with general information about the selected user applications of the currently tuned data service.

Feature: Data Information

2.1.34.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Classified Stream } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	CurrentDataUserApp (0x713)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.34.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, SCIDI, UserAppType, UserAppData }
		{ x=Tag, y=0 }	Tag[Tag], SCIDI[Tag], UserAppType[Tag], UserAppData[Tag]
		{ x=Tag, y=2 }	SCIDI[Tag]
		{ x=Tag, y=3 }	UserAppType[Tag]
		{ x=Tag, y=4 }	UserAppData[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SCIDI

Data Service Component ID internal, virtual ID assigned by the tuner, unique within all known ensembles

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

UserAppType

User Application Type of service component (DLS, MOT Slideshow, TPEG, MOT BWS, ...;see [1], [2]), up to 6 application types per service component may be signaled.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0	0...2047	1	none

UserAppData

For User Application Data Field (see [1],[2], see also User Application Specification [7],[8],[9]). X-PAD applications of the same UserAppType are identified via the XPADAppType, coded in the UserAppData. The stream length is coded in this data type, too.

Basis data type	Length	Media type
Classified Stream	23	application/octet-stream

2.1.35 SelectDataService (0x723)

Occurrence: Optional

This function is not essential but optional. It is used to select a data service, but this does not imply the automatic selection of the corresponding primary component (or user application) like in audio. A data application is first selected (and that means started) with the function SelectDataUserApp.

Feature: Data Administration

2.1.35.1 Format of Function

Function classes: Sequence Method

FBlock	Function	OPType	Parameter
DABTuner (0x43)	SelectDataService (0x723)	StartResultAck	SenderHandle, SinkHandle, SelectionMode, SIDType, SID
		AbortAck	SenderHandle
		ErrorAck	SenderHandle, ErrorCode, ErrorInfo
		ProcessingAck	SenderHandle
		ResultAck	SenderHandle, ServiceState

2.1.35.2 Parameter

SenderHandle

The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SelectionMode

The Selection Mode defines what the DABTuner should select.
"First data service" (resp. "Last data service") selects the first (resp. last) service of the currently tuned ensemble.
"Next data service" and "Previous data service" operate on the currently selected data service, which is described by the SinkHandle and is based upon receiver internal service list of all services in the currently tuned ensemble.
"Select SID" selects the service identified by the parameter SID.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x05	0x00	Off	Off (In conjuction with a

Basis data type	Range of values	Code	Symbolic Name	Description
				method, please use the OPType "Abort")
		0x01	FirstDataService	First data service
		0x02	NextDataService	Next data service
		0x03	PreviousDataService	Previous data service
		0x04	LastDataService	Last data service
		0x05	SelectSID	Select SID

SIDType

Programme or data service structure of SID (e.g. derived from P/D-Flag of the FIG 0 field, see [1], [2]): SID-Type: 0x01 Programme Service 0x02 Data Service

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

ServiceState

The information about the result of the function.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x06	0x00	NotDefined	not defined
		0x01	ServiceInvalid	Service invalid
		0x02	ServiceSelectionDone	Service selection done
		0x03	NoNextService	No next service
		0x04	NoPreviousService	No previous service
		0x05	ServiceNotAvailable	Service not available
		0x06	NoService	No service

2.1.36 SelectDataComponent (0x724)

Occurrence: Optional

This function is not essential but optional. It is used to select a data component, but this does not imply the automatic selection of the corresponding user application a data application is first selected (and that means started) with the function SelectDataUserApp.

Feature: Data, Administration

2.1.36.1 Format of Function

Function classes: Sequence Method

FBlock	Function	OPType	Parameter
DABTuner (0x43)	SelectDataComponent (0x724)	StartResultAck	SenderHandle, SinkHandle, SelectionMode, SCIDI
		AbortAck	SenderHandle
		ErrorAck	SenderHandle, ErrorCode, ErrorInfo
		ProcessingAck	SenderHandle
		ResultAck	SenderHandle, ComponentState

2.1.36.2 Parameter

SenderHandle

The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SelectionMode

The SelectionMode defines which component the DABTuner should select.

"Off": If selection is done by the DAB Tuner, "Off" is given as new status.
"Primary Component" selects the primary service component of the currently tuned service.
"Next Component" and "Previous Component" operate on the currently selected service component.
"Select SCIDI" selects the component identified by the parameter SCIDI.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x04	0x00	Off	Off (in conjunction with a method, please use the Op Type "Abort")
		0x01	PrimaryComponent	Primary Component
		0x02	NextComponent	Next Component
		0x03	PreviousComponent	Previous Component
		0x04	SelectSCIDI	Select SCIDI

SCIDI

Data Service Component ID internal, virtual ID assigned by the tuner, unique within all known ensembles

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

ComponentState

The information about the result of the function.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x07	0x00	NotDefined	not defined
		0x01	ComponentInvalid	Component invalid
		0x02	PrimaryComponent OK	Primary component OK
		0x03	NoPrimary	No primary
		0x04	SecondaryComponent OK	Secondary component OK
		0x05	NoNextComponent	No next component
		0x06	NoPreviousComponent	No previous component
		0x07	ComponentNotAvailable	Component not available

2.1.37 SelectUserApp (0x725)

Occurrence: Optional

This function allows the selection of a user application depending on the favoured (and available) decoding level (DCAP). The applications can either be selected from a list or directly via SCIDI and UserAppType.

Feature: Data, Administration

2.1.37.1 Format of Function

Function classes: Sequence Method

FBlock	Function	OPType	Parameter
DABTuner (0x43)	SelectUserApp (0x725)	StartResultAck	SenderHandle, SinkHandle, SelectionMode, SCIDI, UserAppType, UserAppData, DCAP
		AbortAck	SenderHandle
		ErrorAck	SenderHandle, ErrorCode, ErrorInfo
		ProcessingAck	SenderHandle
		ResultAck	SenderHandle, DCAP, UserAppState

2.1.37.2 Parameter

SenderHandle

The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SelectionMode

The Selection Mode defines what the DABTuner should select.
"Off": If selection is done by the DAB Tuner, "Off" is given as new status.
"First user application" (resp. "Last user application") selects the first (resp. last) user application of the currently tuned component.
"Next user application" and "Previous user application" operate on the currently selected user application, which is described by the SinkHandle and is based upon receiver internal user application list of all services in the currently tuned component.
"Select UserAppType" selects the user application identified by the parameter SCIDI and UserAppType.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x05	0x00	Off	Off (in conjunction with a method, please use the Op Type "Abort")
		0x01	FirstUserApplication	First user application
		0x02	NextUserApplication	Next user application
		0x03	PreviousUserApplication	Previous user application
		0x04	LastUserApplication	Last user application
		0x05	SelectUserAppType	Select UserAppType

SCIDI

Data Service Component ID internal, virtual ID assigned by the tuner, unique within all known ensembles

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

UserAppType

User Application Type of service component (DLS, MOT Slideshow, TPEG, MOT BWS, ...;see [1], [2]), up to 6 application types per service component may be signaled

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

UserAppData

For User Application Data Field (see [1],[2], see also User Application Specification [7],[8],[9]). X-PAD applications of the same UserAppType are identified via the XPADAppType, coded in the UserAppData. The stream length is coded in this data type, too.

Basis data type	Length	Media type
Classified Stream	23	application/octet-stream

DCAP

Decoding capabilities. Information about the DataServices decoder.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

UserAppState

The information about the result of the function.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x05	0x00	NotDefined	not defined
		0x01	UserApplicationInvalid	User application invalid
		0x02	UserApplicationOk	User application OK
		0x03	NoNextUserApplication	No next user application
		0x04	NoPreviousUserApplication	No previous user application
		0x05	UserApplicationNotAvailable	User application not available

2.1.38 DataServiceList (0x730)

Occurrence: Optional

List with general information about the known data services.

The function can be used as a DynamicArray or as a MotherArray for an ArrayWindow. For this case the FktIDs 0x731 to 0x73F are reserved for possible instance-handles.

Feature: Data, Information

2.1.38.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Number Number Boolean Number String Number } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	DataServiceList (0x730)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.38.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, EnsID, EnsECC, SIDType, SID, LabelStatus, Label8CharAbbreviation, Label, SCAID }
		{ x=Tag, y=0 }	Tag[Tag], EnsID[Tag], EnsECC[Tag], SIDType[Tag], SID[Tag], LabelStatus[Tag], Label8CharAbbreviation[Tag], Label[Tag], SCAID[Tag]
		{ x=Tag, y=2 }	EnsID[Tag]
		{ x=Tag, y=3 }	EnsECC[Tag]
		{ x=Tag, y=4 }	SIDType[Tag]
		{ x=Tag, y=5 }	SID[Tag]

Basis data type	Length	Description	
		{ x=Tag, y=6 }	LabelStatus[Tag]
		{ x=Tag, y=7 }	Label8CharAbbreviation[Tag]
		{ x=Tag, y=8 }	Label[Tag]
		{ x=Tag, y=9 }	SCAID[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsID

Ensemble identifier

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsECC

Extended country code of the Ensemble

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

SIDType

Programme or data service structure of SID (e.g. derived from P/D-Flag of the FIG 0 field, see [1], [2]):
SID-Type: 0x01 Programme Service 0x02 Data Service

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

LabelStatus

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	Label valid
		False	Label not valid

Label8CharAbbreviation

Bitfield which is added with the Label to get an abbreviation. This is a Bitmask from DAB please see Spec [1].

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Label

It shall be coded as a string of 16 characters

Basis data type	MaxSize
String	16

SCAID

Conditional Access Indentifier (see [1], [2]): 0x00 = no access control or don't care, 0x01 = NR MSK, 0x02 = Eurocrypt EN50094

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

2.1.39 DataComponentList (0x740)

Occurrence: Optional

List with general information about the currently known data service components.

The function can be used as a DynamicArray or as a MotherArray for an ArrayWindow. For this case the FktIDs 0x741 to 0x74F are reserved for possible instance-handles.

Feature: Data, Information

2.1.39.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Number Number Number Boolean Number String Number Number Enumeration Number Boolean } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	DataComponentList (0x740)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.39.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, EnsID, EnsECC, SIDType, SID, SCIDI, LabelStatus, Label8CharAbbreviation, Label, Language, DSCTy, TMID, SCIDS, SCCA-Flag }
		{ x=Tag, y=0 }	Tag[Tag], EnsID[Tag], EnsECC[Tag], SIDType[Tag], SID[Tag], SCIDI[Tag], LabelStatus[Tag], Label8CharAbbreviation[Tag], Label[Tag], Language[Tag], DSCTy[Tag], TMID[Tag], SCIDS[Tag], SCCA-Flag[Tag]
		{ x=Tag, y=2 }	EnsID[Tag]

Basis data type	Length	Description	
		{ x=Tag, y=3 }	EnsECC[Tag]
		{ x=Tag, y=4 }	SIDType[Tag]
		{ x=Tag, y=5 }	SID[Tag]
		{ x=Tag, y=6 }	SCIDI[Tag]
		{ x=Tag, y=7 }	LabelStatus[Tag]
		{ x=Tag, y=8 }	Label8CharAbbreviation[Tag]
		{ x=Tag, y=9 }	Label[Tag]
		{ x=Tag, y=10 }	Language[Tag]
		{ x=Tag, y=11 }	DSCTy[Tag]
		{ x=Tag, y=12 }	TMID[Tag]
		{ x=Tag, y=13 }	SCIDS[Tag]
		{ x=Tag, y=14 }	SCCA-Flag[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsID

Ensemble identifier

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

EnsECC

Extended country code of the Ensemble

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

SIDType

Programme or data service structure of SID (e.g. derived from P/D-Flag of the FIG 0 field, see [1], [2]):
SID-Type: 0x01 Programme Service 0x02 Data Service

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

SID

The SID can be either programme service id or data service id (for DAB data only). Even though the service has a programme service id, the content of the service can be data, too. In the data service ID the first Byte is not used to define the type, so the additional parameter SIDType is needed for data services.

The service identifier (SID) (see [1], [2])

Programme Service:	Byte 0 (Bit 0...7):	Programm Service Type (0x00 undef; 0x01 DABAudioService)
	Byte 1 (Bit 8...15):	ExtendedCountryCode (ECC)
	Byte 2, 3 (Bit 16...31):	ProgrammeServiceIdentifier (12 Bit Service-Reference + 4 Bit Country-id)
Data Service:	Byte 0...2 (Bit 0...23):	DataServiceID (20 Bit Service-Reference + 4 Bit Country-id)
	Byte 3 (Bit 24...31):	ExtendedCountryCode (ECC)

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

SCIDI

Data Service Component ID internal, virtual ID assigned by the tuner, unique within all known ensembles

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

LabelStatus

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	Label valid
		False	Label not valid

Label8CharAbbreviation

Bitfield which is added with the Label to get an abbreviation. This is a Bitmask from DAB please see Spec [1].

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

Label

It shall be coded as a string of 16 characters

Basis data type	MaxSize
String	16

Language

Service component language

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

DSCTy

Combines Data Service Component Type (DSCTy) and extension (ExtDSCTy) (see [1], [2])
Bit 0-5: Service Component Type
Bit 6-15: Service Component Type extension

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

TmID

Transport mechanism identifier

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x03	0x00	AudioStream	Audio Stream
		0x01	DataStream	Data Stream
		0x02	FIDC	FIDC
		0x03	PacketMode	Packet Mode

SCIDS

Combines Service Component ID (packet mode only) and Service component ID within the service (see [1], [2])
Bit 0-3: Service component ID within the service, range 0..15 (all modes)
Bit 4-15: Service Component ID (packet mode only), unique within the ensemble

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SCCA-Flag

This Flag indicates wether or not Service Component Conditional Access (SCCA) is applied.

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	SCCA applied
		False	SCCA not applied

2.1.40 DataUserAppList (0x750)

Occurrence: Optional

List with information about user applications related to tuned ensembles.
The function can be used as a DynamicArray or as a MotherArray for an ArrayWindow. For this case the FktIDs 0x751 to 0x75F are reserved for possible instance-handles.

Feature: Data, Information

2.1.40.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Classified Stream } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	DataUserAppList (0x750)	Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.40.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, SCIDI, UserAppType, UserAppData }
		{ x=Tag, y=0 }	Tag[Tag], SCIDI[Tag], UserAppType[Tag], UserAppData[Tag]
		{ x=Tag, y=2 }	SCIDI[Tag]
		{ x=Tag, y=3 }	UserAppType[Tag]
		{ x=Tag, y=4 }	UserAppData[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SCIDI

Data Service Component ID internal, virtual ID assigned by the tuner, unique within all known ensembles

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

UserAppType

User Application Type of service component (DLS, MOT Slideshow, TPEG, MOT BWS, ...;see [1], [2]), up to 6 application types per service component may be signaled.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0	0...2047	1	none

UserAppData

For User Application Data Field (see [1],[2], see also User Application Specification [7],[8],[9]). X-PAD applications of the same UserAppType are identified via the XPADAppType, coded in the UserAppData. The stream length is coded in this data type, too.

Basis data type	Length	Media type
Classified Stream	23	application/octet-stream

2.1.41 DataDecoderCap (0x760)

Occurrence: Optional

Information about the capability of the DAB tuner to decode data services.

Feature: Data, Information

2.1.41.1 Format of Function

Function classes: Number

FBlock	Function	OPType	Parameter
DABTuner (0x43)	DataDecoderCap (0x760)	Get	-
		Status	DCAP
		Error	ErrorCode, ErrorInfo

2.1.41.2 Parameter

DCAP

Decoding capabilities. Information about the DataServices decoder

Annotations:

- The decoding on the lower levels of DAB (fields, groups, blocks) shouldn't separate the header from the data field of the object. For example the decoding of Packet Mode to the level of data groups means that the complete data groups with data group header are sent through the MOST channel.
- Also MOT-objects are sent together with its header if the decoding level is on MOT-layer.
- If decoded MOT objects are sent the header of the containing MOT might be sent separately before the object (e.g., a jpeg or a tiff-file) to transmit the overall header information. The MOT directory will also be send as a separate object.

The following table describes the bit positions of the parameter 'DCAP'

b31	b30 ... 27	b26 ... b0
1 bit	4 bits	27 bits
CA	supported transport mode	supported decoding level

0	0 0 0 0: raw	The tuner is not able to resolve the data services of the DAB ensemble (e.g., just an audio-tuner). Therefore just a raw bit stream is transmitted. b2 b1 b0 0 0 0: format of bit stream not defined 0 0 1: RDI format is supported [11]
	x x x 1: XPAD	b11 ... b3 are reserved for definitions of XPAD-decoding levels (e.g., stream of XPAD-fields, XPAD data groups, XPAD objects, user group stream data channel, user group stream packet data channel) combination b11...b3 = 1 signals decoding of XPAD service components according DSCTY
	x x 1 x: stream data	b14 ... b12

		are reserved for definitions of stream data decoding levels combination b14...b12 = 1 signals decoding of stream data service components according DSCTY **Note:** Stream data and isochronous data is not specified further here, consider in particular the results of the 'Multimedia Streaming Group'
	x 1 x x: FIDC	b20 ... b15 are reserved for definitions of FIDC decoding levels (e.g., stream of fast information blocks, fast information groups ...) combination b20...b15 = 1 signals decoding of FIDC service components according DSCTY
	1 x x x: packet mode	b24 b23 b22 b21 x 0 0 0: stream of MSC data packets x 0 0 1: stream of MSC data groups x 0 1 0: stream of objects (MOT, embedded IP, ...) x 0 1 1: stream of decoded objects (TIFF, HTML, embedded TPEG ...., no MOTs or IPs)
1	Reserved for conditional access: tuner is able to perform descrambling of encrypted data-objects (=1) or not (=0)	
All other combinations reserved for future definitions		

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Long	0		1	none

2.1.42 SelectDataServiceStatus (0x773)

Occurrence: Optional

With this function the current SelectDataServiceStatus can be retrieved

Feature: Data, Administration

2.1.42.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
DABTuner (0x43)	SelectDataService Status (0x773)	Get	SinkHandle
		Status	SinkHandle, SelectionMode
		Error	ErrorCode, ErrorInfo

2.1.42.2 Parameter

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SelectionMode

The Selection Mode defines what the DABTuner should select.
"First data service" (resp. "Last data service") selects the first (resp. last) service of the currently tuned ensemble.
"Next data service" and "Previous data service" operate on the currently selected data service, which is described by the SinkHandle and is based upon receiver internal service list of all services in the currently tuned ensemble.
"Select SID" selects the service identified by the parameter SID.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x05	0x00	Off	Off (In conjuction with a method, please use the OPType "Abort")
		0x01	FirstDataService	First data service
		0x02	NextDataService	Next data service
		0x03	PreviousDataService	Previous data service
		0x04	LastDataService	Last data service
		0x05	SelectSID	Select SID

2.1.43 SelectDataComponentStatus (0x774)

Occurrence: Optional

With this function the current SelectDataComponentStatus can be retrieved.

Feature: Data, Administration

2.1.43.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
DABTuner (0x43)	SelectData ComponentStatus (0x774)	Get	SinkHandle
		Status	SinkHandle, SelectionMode
		Error	ErrorCode, ErrorInfo

2.1.43.2 Parameter

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SelectionMode

The SelectionMode defines which component the DABTuner should select.

"Off": If selection is done by the DAB Tuner, "Off" is given as new status.
"Primary Component" selects the primary service component of the currently tuned service.
"Next Component" and "Previous Component" operate on the currently selected service component.
"Select SCIDI" selects the component identified by the parameter SCIDI.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x04	0x00	Off	Off (In conjuction with a method, please use the OPType "Abort")
		0x01	PrimaryComponent	Primary Component
		0x02	NextComponent	Next Component
		0x03	PreviousComponent	Previous Component
		0x04	SelectSCIDI	Select SCIDI

2.1.44 SelectUserAppStatus (0x775)

Occurrence: Optional

With this function the current SelectUserAppStatus can be retrieved.

Feature: Data, Administration

2.1.44.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
DABTuner (0x43)	SelectUser AppStatus (0x775)	Get	SinkHandle
		Status	SinkHandle, SelectionMode
		Error	ErrorCode, ErrorInfo

2.1.44.2 Parameter

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SelectionMode

The Selection Mode defines what the DABTuner should select.
"Off": If selection is done by the DAB Tuner, "Off" is given as new status.
"First user application" (resp. "Last user application") selects the first (resp. last) user application of the currently tuned component.
"Next user application" and "Previous user application" operate on the currently selected user application, which is described by the SinkHandle and is based upon receiver internal user application list of all services in the currently tuned component.
"Select UserAppType" selects the user application identified by the parameter SCIDI and UserAppType.

Basis data type	Range of values	Code	Symbolic Name	Description
Enum	0x00...0x05	0x00	Off	Off (In conjuction with a method, please use the OPType "Abort")
		0x01	FirstUserApplication	First user application
		0x02	NextUserApplication	Next user application
		0x03	PreviousUserApplication	Previous user application
		0x04	LastUserApplication	Last user application
		0x05	SelectUserAppType	Select UserAppType

2.1.45 CreateDataChannel (0x800)

Occurrence: Optional

An application device requests a data channel from the DAB tuner. The dynamically created data channel function is identified by the SinkHandle (=FktID). This function is used to transfer the requested data from the DAB tuner to the application. The FktIDMotherDataChannel identifies the kind of data channel to use (different parameter list). By default, the channel status is set to "not active".

2.1.45.1 Format of Function

Function classes: Sequence Method

FBlock	Function	OPType	Parameter
DABTuner (0x43)	CreateData Channel (0x800)	StartResultAck	SenderHandle, FktIDMotherDataChannel
		AbortAck	SenderHandle
		ErrorAck	SenderHandle, ErrorCode, ErrorInfo
		ProcessingAck	SenderHandle
		ResultAck	SenderHandle, SinkHandle

2.1.45.2 Parameter

SenderHandle

The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

FktIDMotherDataChannel

FunctionID of the MotherDataChannel Function.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

2.1.46 RemoveDataChannel (0x801)

Occurrence: Optional

This function removes an existing data channel, identified by the SinkHandle.

2.1.46.1 Format of Function

Function classes: Sequence Method

FBlock	Function	OPType	Parameter
DABTuner (0x43)	RemoveData Channel (0x801)	StartResultAck	SenderHandle, SinkHandle
		AbortAck	SenderHandle
		ErrorAck	SenderHandle, ErrorCode, ErrorInfo
		ProcessingAck	SenderHandle
		ResultAck	SenderHandle, SinkHandle

2.1.46.2 Parameter

SenderHandle

The SenderHandle is a unique identifier of the "send"-task within the device. It is used to distinguish between different tasks of the device.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

2.1.47 DataChannelList (0x805)

Occurrence: Optional

This function is used to change or to report the status of the data channels. They are identified by the SinkHandle and the respective DeviceID of the sink application. By default, the channel status of a new created function is set to "not active".

2.1.47.1 Format of Function

Function classes: DynamicArray of { Record of { Number Number Number Boolean Number } }

FBlock	Function	OPType	Parameter
DABTuner (0x43)	DataChannelList (0x805)	Set	Tag, PosY, Data
		Get	Tag, PosY
		Status	Tag, PosY, Data
		Error	ErrorCode, ErrorInfo

2.1.47.2 Parameter

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

PosY

The parameter PosY consists of one byte and shows what is to be set, requested or read in the record.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Byte	0		1	none

Data

Basis data type	Length	Description	
Array	-	Pos	Data
		{ x=0, y=0 }	{ Tag, SinkHandle, FktIDMotherDataChannel, ChannelStatus, DeviceIDApp }
		{ x=Tag, y=0 }	Tag[Tag], SinkHandle[Tag], FktIDMotherDataChannel[Tag], ChannelStatus[Tag], DeviceIDApp[Tag]
		{ x=Tag, y=2 }	SinkHandle[Tag]
		{ x=Tag, y=3 }	FktIDMotherDataChannel[Tag],
		{ x=Tag, y=4 }	ChannelStatus[Tag]
		{ x=Tag, y=5 }	DeviceIDApp[Tag]

Tag

The Tag is unique in the DABTuner and shows what is to be requested or read in the array. The Tag corresponds to the SinkHandle.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

SinkHandle

A SinkHandle is referred to as SourceNr that the DABTuner will use to stream synchronous data onto the MOST Network.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

FktIDMotherDataChannel

FunctionID of the MotherDataChannel Function

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

ChannelStatus

Status of the channel connection. If the channel is removed or just paused the status is "Not active".

Basis data type	Bit #	Code	Description
Boolean	Bit 0	True	Active
		False	Not active

DeviceIDApp

DeviceID of the application which requested the data channel.

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

2.1.48 MotherDataChannel (0x810)

Occurrence: Optional

This is the sample function for dynamically created data channel functions. They are used to transfer data streams of different data services from the DAB tuner to the corresponding application devices. The data is transferred in the DataStream parameter. The type and length of the data package are also added as parameters. The receiving application is then able to decode the requested data from the channel function. A new function is created for every user application or PAD. By default, the channel status of a new created function is set to "not active".

Note: this function is only a sample function and NOT to be used in the real system, notification is not possible, this FktID is never used!!!

2.1.48.1 Format of Function

Function classes: Unclassified Property

FBlock	Function	OPType	Parameter
DABTuner (0x43)	MotherDataChannel (0x810)	Status	DataStreamType, DataStreamLength, DataStream
		Error	ErrorCode, ErrorInfo

2.1.48.2 Parameter

DataStreamType

Indicates the type structure of the data stream object (e.g. XPAD Datatypes etc.)

Basis data type	Bit #	Code	Description
Unsigned Word	Bit 0	False	D_Type is not present in Parameter DataStream
		True	D_Type is present in Parameter DataStream
	Bit 1	False	Tr_ID is not present in Parameter DataStream
		True	Tr_ID is present in Parameter DataStream
	Bit 2	False	Usr_Adr and Usr_Adr_length is not present in Parameter DataStream
		True	Usr_Adr and Usr_Adr_length is present in Parameter DataStream
	Bit 3	False	CA_Param is not present in Parameter DataStream
		True	CA_Param is present in Parameter DataStream
	Bit 4	False	CA_IM is not present in Parameter DataStream
		True	CA_IM is present in Parameter DataStream
	Bit 5	False	Reserved for future definitions
		True	Reserved for future definitions
	Bit 6	False	Reserved for future definitions
		True	Reserved for future definitions
	Bit 7	False	Reserved for future definitions
		True	Reserved for future definitions
	Bit 8	False	Reserved for future definitions
		True	Reserved for future definitions

Basis data type	Bit #	Code	Description
	Bit 9	False	Reserved for future definitions
		True	Reserved for future definitions
	Bit 10	False	Reserved for future definitions
		True	Reserved for future definitions
	Bit 11	False	Reserved for future definitions
		True	Reserved for future definitions
	Bit 12	False	Reserved for future definitions
		True	Reserved for future definitions
	Bit 13	False	Reserved for future definitions
		True	Reserved for future definitions
	Bit 14	False	Reserved for future definitions
		True	Reserved for future definitions
	Bit 15	False	Reserved for future definitions
		True	Reserved for future definitions

DataStreamLength

Indicates the length of the DataStream

Basis data type	Exp.	Range of values	Step	Unit
Unsigned Word	0		1	none

DataStream

Source data stream
Definition of the first bytes of the parameter "DataStream" depending on the parameter "DataStreamType"

Attributename	DataStreamType	Description	DataType
D_type	Extension 0 (Bit0 of DataStreamType)	Data type of the transmitted object: in particular used if type can't be derived from service component definition but is related to the particular object (see table below)	Unsigned Long
Tr_Id	Extension 1 (Bit1 of DataStreamType)	Transport-Id of the data object: used if several transmitted data blocks belong to one data object with this id and the object has to be cut into pieces for transport (e.g., from session header of MSC data groups)	Unsigned Word
Usr_Adr	Extension 2 (Bit2 of DataStreamType)	Length of user address field (Bytes)	Unsigned Byte
Usr_Adr_length	End user address field (from session header of MSC data-group)	Array [0...13 Byte]	
CA_Param	Extension 3 (Bit3 of DataStreamType) only necessary if transmitted objects are scrambled for conditional access	Descrambling parameters for conditional access; Dependent from transport mode this field contents the parameters from the SCCA/DGCA/FIDCCA/FIDCCA_Ext field (except initialization modifier which is signaled in 'CA_IM' below): - b5...0: ECM identifier from FIDCCA_Ext or SCCA - b7...b6: Transport flags from FIDCCA_Ext	Unsigned Word

	and CA is not performed by tuner	or SCCA - b8: update ECM flag for all modes - b10...b9: update mode from SCCA - b11: update access from SCCA - b13...b12: scrambling mode (all modes) - b14: parity flag (DGCA, FIDCCA, FIDCCA_ext) - b15: replacement flag(all modes) See DAB-Standard [1] of Conditional Access (in particular chapter 9.2)	
CA_IM	Extension 4 (Bit4 of DataStreamType) only necessary if transmitted objects are scrambled for conditional access and CA is not performed by tuner	Initialization Modifier for CA-descrambling: 24 bit used for - packet mode with CA-configuration 1/2/3 - FIDC with CA-configuration 1 16 bit used for - packet mode with CA-configuration 2/3 - FIDC with CA-configuration 2 - stream mode with CA-configuration 1 (only this) b31...b24 not used see DAB-Standard [1] of Conditional Access (in particular chapter 9.2)	Unsigned Long

Definition parameter D_type
The following table defines the parameter D_Type, the data type of the transmitted object, which is added to the data stream (parameter Data) if the extension 0 of the type structure is set on true.

Transport mode	Decoding level
All modes	For decoded objects this parameter is used with the following type definitions (decimal numbers): 0 = Undefined 1 = Traffic Message Channel (TMC), user message 2 = Traffic Message Channel (TMC), system message 3 = Emergency Warning System (EWS), control information 4 = Emergency Warning System (EWS), message 5 = Paging message 6 = Paging pointer 7 = Dynamic Label (DLS) 8 = Interactive Text Transmission (ITTS) 9 = In-house Data 10 = Multimedia Object Transport (MOT), header + contents * 11 = MOT header information (MOT type/subtype/extended header...) ** 12 = MOT data (transmitted object contains the content of a MOT object) ** 13 = Transparent Data Channel (TDC) 14 = Embedded IP Packet 15 = CA-message (EMM/ECM) 16...32767 are reserved for future definitions of decoded objects > 32768...65535 are used for type definitions on lower protocol levels (e.g., MSC-data-groups)
Special definitions XPAD	For decoded objects the types of the table above are restricted to definitions 0 and 7 ... 15 For operation of closed user-group packet channel use definitions of packet mode for closed user-group stream data channel no further definitions are specified here (see stream data mode below)
Special definitions stream data	The stream data is a transparent transport mechanism for any data object. If any data-type is known it is signaled by the DSCTY-parameter of the related service component so generally it shouldn't be necessary to transmit the D_type. A

	more common type definition (e.g., MIME type) might be used in future. In particular the specification of stream data and isochronous data is actually done by the 'Multimedia Streaming Group'. A type definition of the stream data should comply with the definitions of this group. A general type definition like this will require an additional extension in the data-transmission-object e.g., a string for the MIME-type-definition.
Special definitions FIDC	For decoded objects the types of table above are restricted to definitions 0 ... 6 and 15
Special definitions packet mode	For decoded objects the types of table above are restricted to definitions 0 and 10 ... 15

*Used if a complete or segmented MOT object is transmitted; if segmented use transport_id to indicate that transmitted objects belong to one MOT
**used if a MOT object has been de-segmented and the content (a TIFF, a TPEG, ...) is transmitted: the header is sent as a additional transmission-object in front of the embedded file. Header and content/file have the same transport_id 0

Some annotations:

- It is only necessary to transmit this parameter if it's not possible to derive the type from the definition of the service component (parameter DSCTY) and the tuner decoding capabilities (parameter DCAP of the function TunerCapabilites and as part of the user-application-data).
Some Examples:
  * D_Type is redundant if the related service component is of 'packet mode' and the tuner is just able to assemble MSC-packets. Then no further information is required because the transmitted objects can only be packets and nothing else.
  * An entry D_Type is always necessary for XPAD transport mode because in this case the type is not fixed within the service component but may vary with each transmitted XPAD-data-group.
- This parameter should only be used for types of objects which are defined within the DAB-standard (e.g., MSC-data-groups, MSC-packets, EMC/EMM-messages, TMC-objects) or related documents (e.g., MOT objects, TPEG messages ...). It is not suited to signal objects; which are tunneled via DAB by a higher level protocol such as objects within Embedded IP, Transparent Data Channel (TDC), In-House Data or undefined stream data.
-> Actually the receiving application has to deal with this problem. A more general type definition which is feasible also for higher level objects should be defined in future by the 'MOST Multimedia Streaming Group'. A potential candidate might be the MIME-type.
- If MOT-objects are decoded by the tuner and the contained objects (TPEG, TIFF...) are transmitted the parameter should contain the definition according the MOT standard.

Basis data type	Length	Description
Stream	65535	

3 Dynamic Specification

3.1 Introduction

The following section deals with application specific communication sequences. All described sequences have to be viewed together with the function catalogues for MOST. The current document concentrates on the exemplary sequences; it does not claim to be a complete description of the device communication over MOST.

3.2 Mandatory

3.2.1 DAB Tuner Notification Matrices

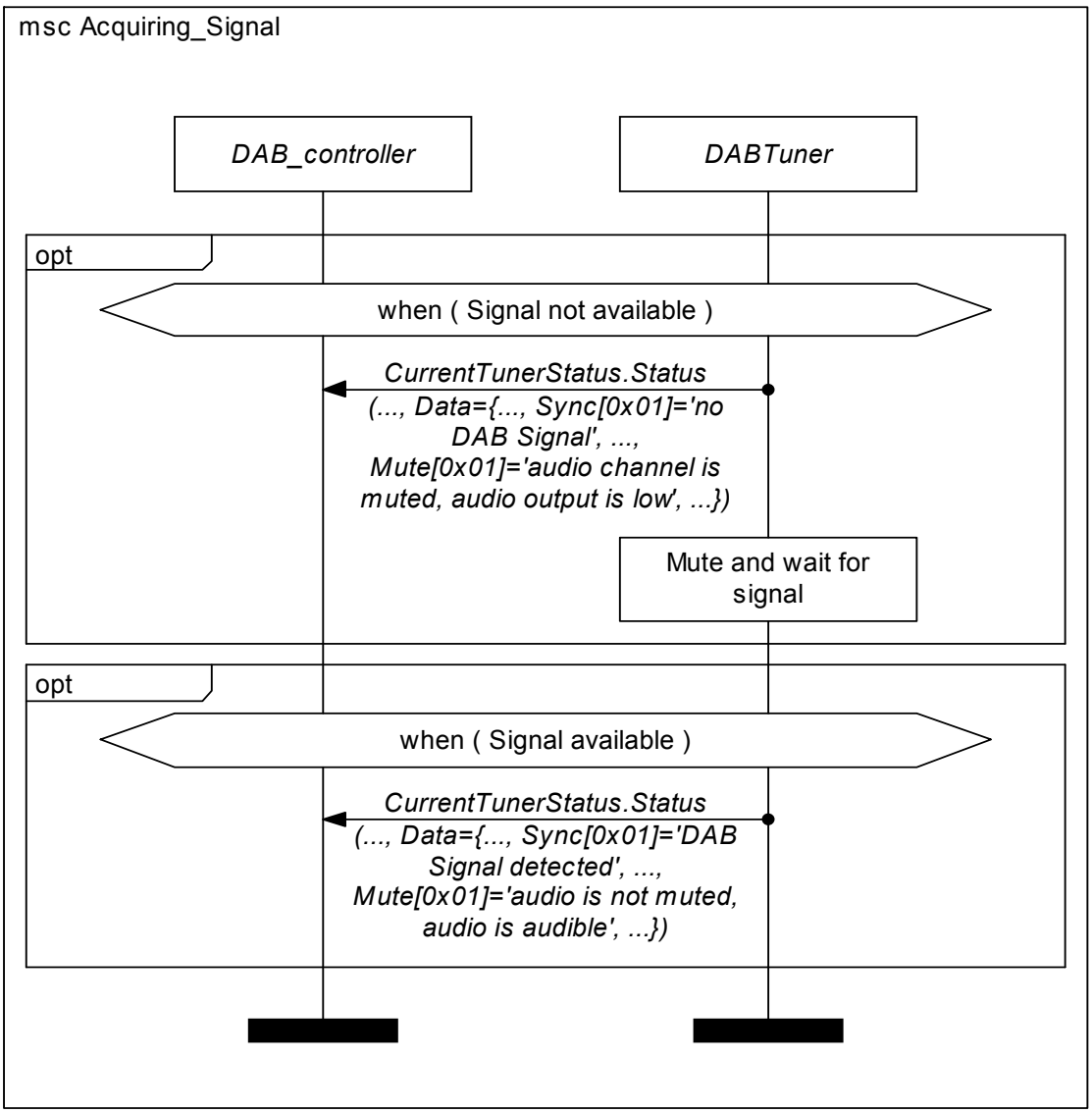
Use Case:	DAB Tuner Notification Matrices		
Description:	For the functionality DAB Tuner it's required that the listed devices are registered in the Notification matrices for the following functions.		
	DABTuner.0x00.AudioPresets	DAB_controller	
	DABTuner.0x00.CurrentEnsemble	DAB_controller	
	DABTuner.0x00.CurrentAudioService	DAB_controller	
	DABTuner.0x00.CurrentAudioComponent	DAB_controller	
	DABTuner.0x00.CurrentDataService	DAB_controller	
	DABTuner.0x00.CurrentDataComponent	DAB_controller	
	DABTuner.0x00.CurrentDataUserApp	DAB_controller	
	DABTuner.0x00.CurrentTunerStatus	DAB_controller	
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:			

3.2.2 Acquiring Signal

Use Case:	Acquiring Signal		
Description:	The DAB module is not synchronized on a DAB signal for one of the following reasons:		
		Startup DAB signal has been lost (when leaving the coverage area of a DAB transmitter) DAB module is performing a seek procedure	
	The DAB module monitors the availability of the DAB signal, and reports the status to the DAB_controller.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X		
Remarks:			

3.2.2.1 General MSCs

DABTuner



MSC 1: Acquiring Signal

3.2.2.2 Example MSCs

Not Available

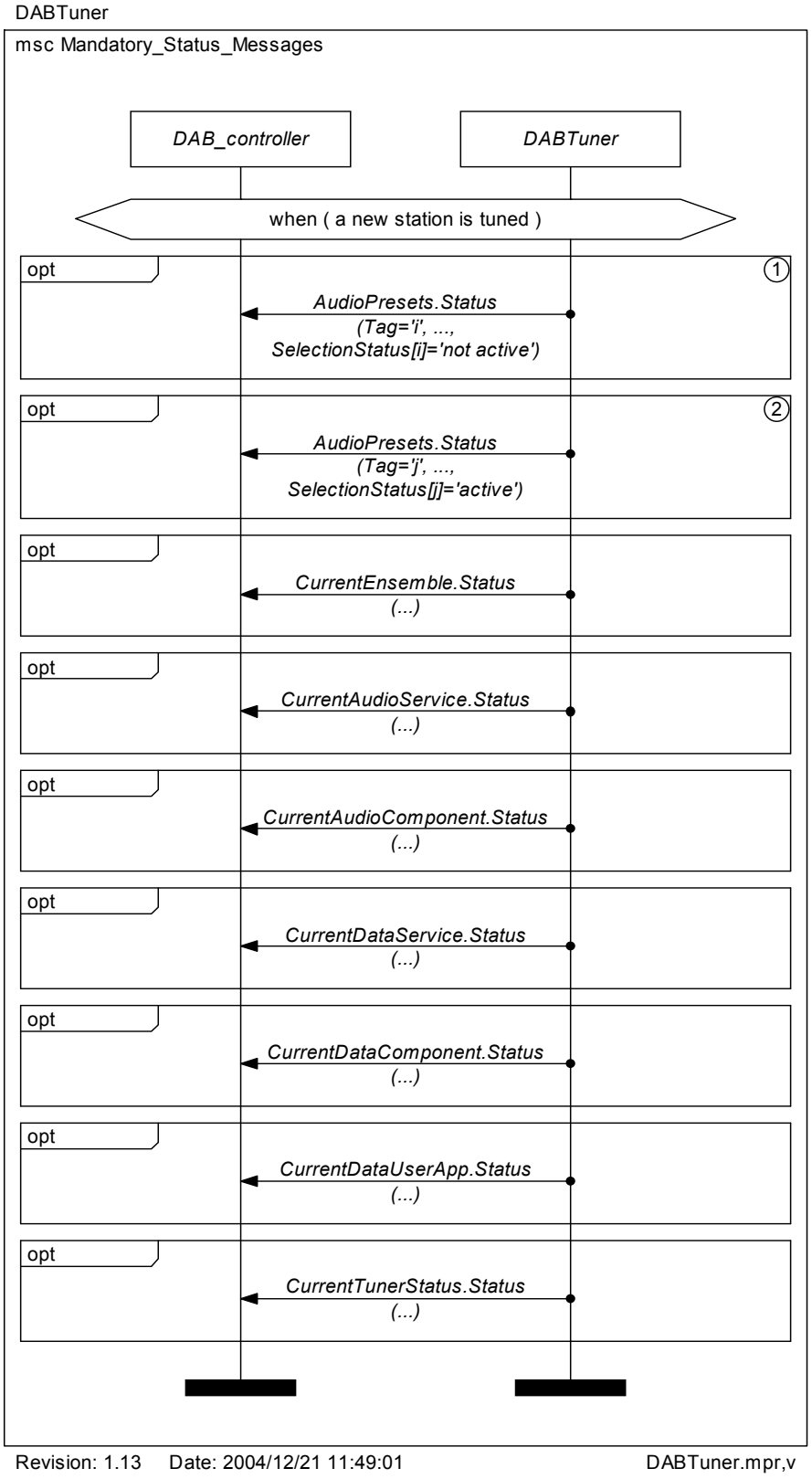
3.2.2.3 Error MSCs

Not Available

3.2.3 Mandatory Status Messages

Use Case:	Mandatory status messages		
Description:	If a new station is tuned the status of these functions will change as well.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:			

3.2.3.1 General MSCs



MSC 2: Mandatory status messages

1 If this preset has been the former station
2 If this preset is the current station

3.2.3.2 Example MSCs

Not Available

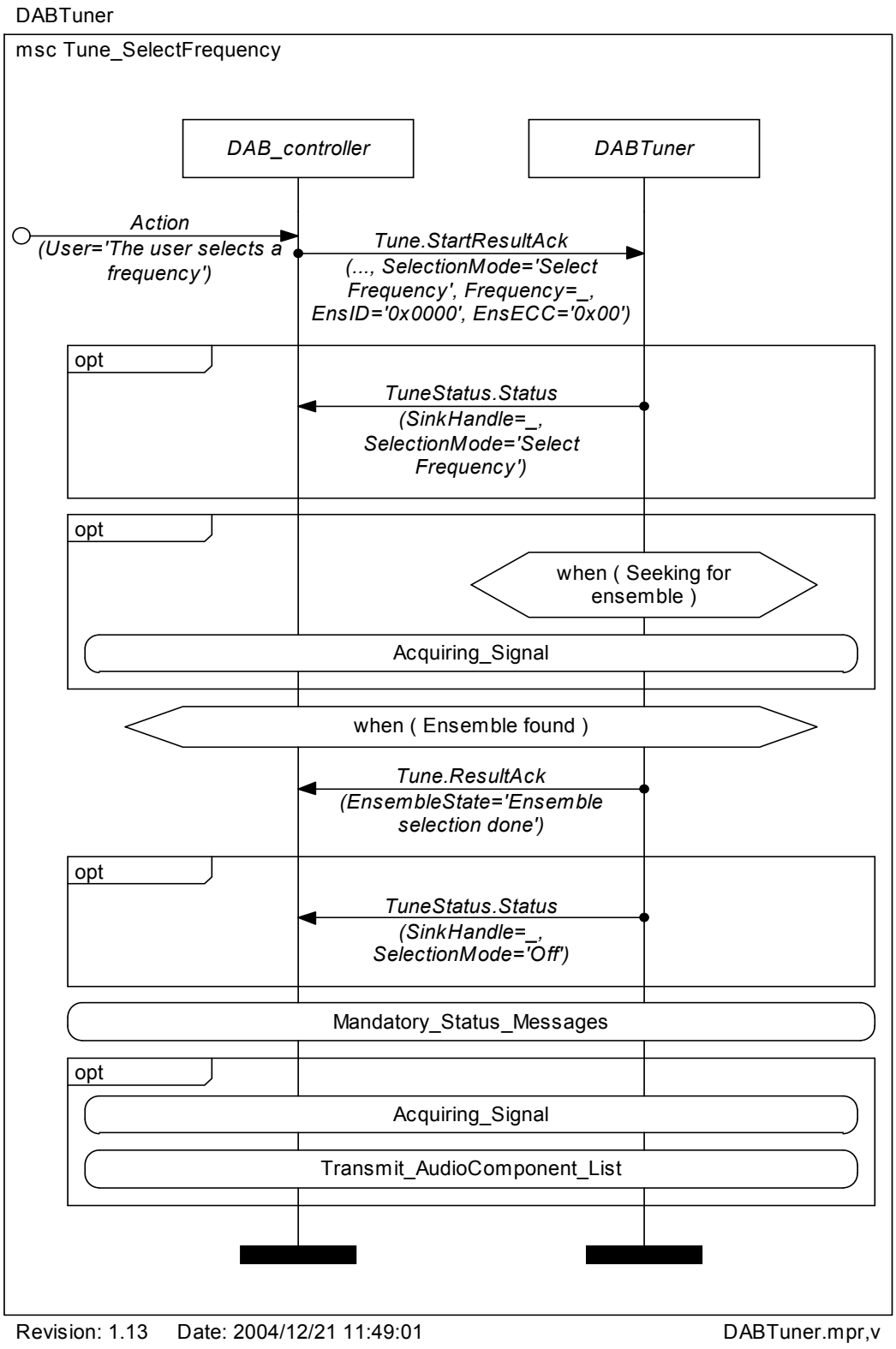
3.2.3.3 Error MSCs

Not Available

3.2.4 Tune SelectFrequency

Use Case:	Tune SelectFrequency		
Description:	The user selects a frequency. The DAB module tunes then to the frequency and plays the audio.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X		
Remarks:			

3.2.4.1 General MSCs



MSC 3: Tune SelectFrequency

3.2.4.2 Example MSCs

Not Available

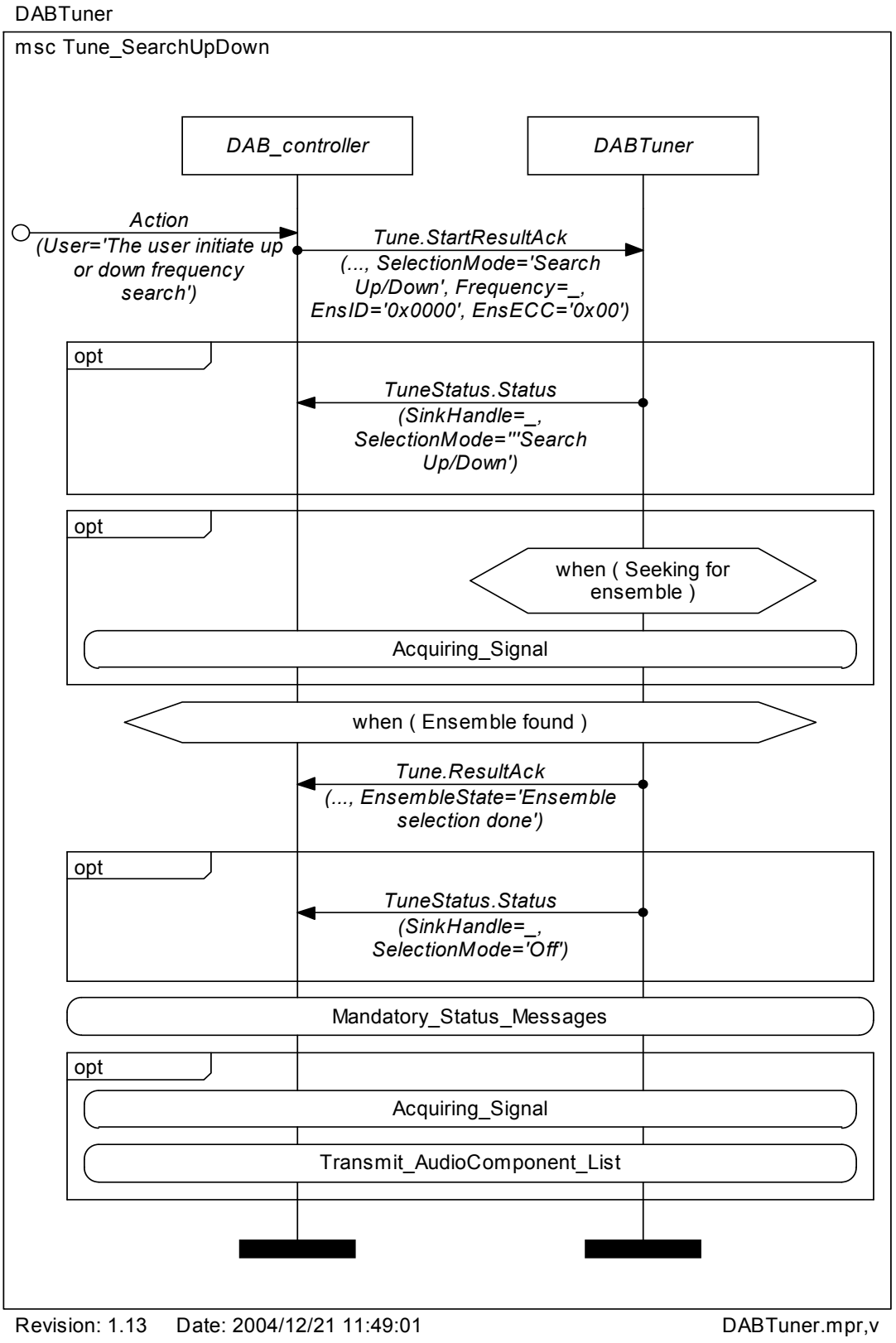
3.2.4.3 Error MSCs

Not Available

3.2.5 Tune SearchUpDown

Use Case:	Tune SearchUpDown		
Description:	The user initiate up or down frequency search.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X		
Remarks:			

3.2.5.1 General MSCs



MSC 4: Tune SearchUpDown

3.2.5.2 Example MSCs

Not Available

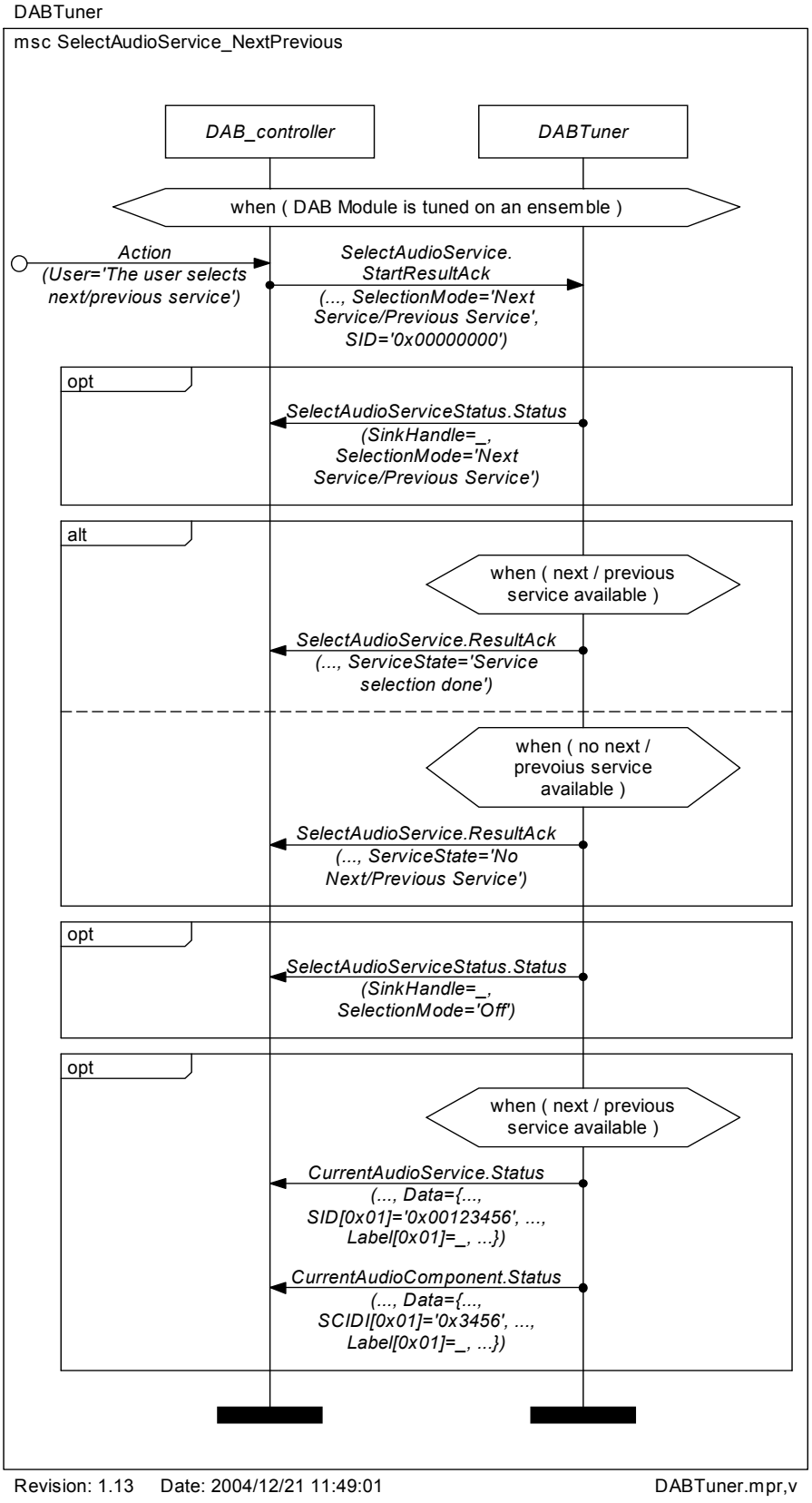
3.2.5.3 Error MSCs

Not Available

3.2.6 Select AudioService Next Previous

Use Case:	Select AudioService Next Previous		
Description:	The user selects the next or previous audio service within the currently tuned ensemble. As default, the primary audio service components selected by the DAB Tuner.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X		
Remarks:			

3.2.6.1 General MSCs



MSC 5: Select AudioService Next Previous

3.2.6.2 Example MSCs

Not Available

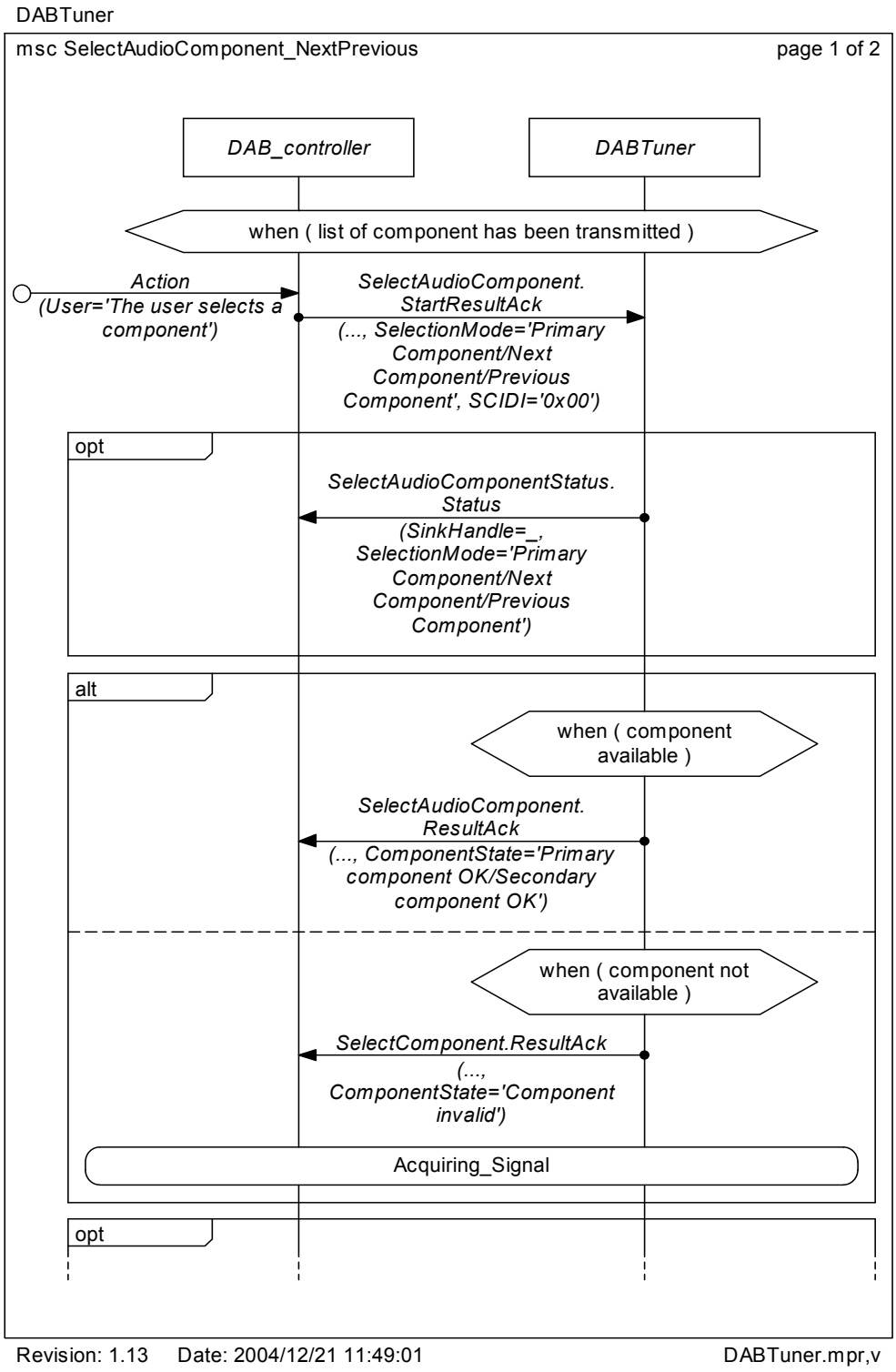
3.2.6.3 Error MSCs

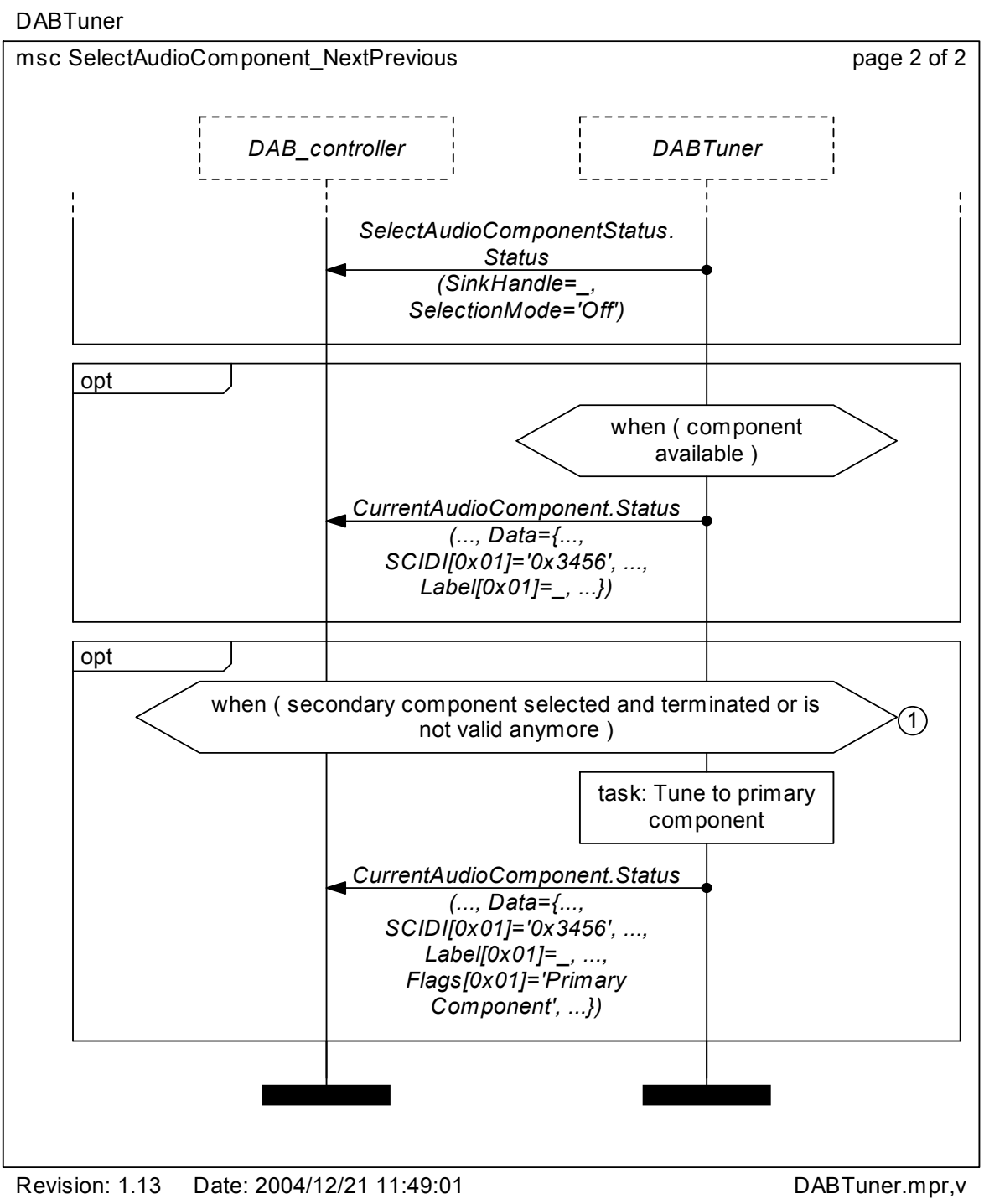
Not Available

3.2.7 Select Component Next Previous

Use Case:	Select Component Next Previous		
Description:	The user selects a secondary component within the currently tuned service or the primary component while being tuned to a secondary component.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X		
Remarks:			

3.2.7.1 General MSCs





MSC 6: Select Component Next Previous

1 The DABTuner should store the SID related to the primary component. If the DABTuner does not store this information, it is not possible to tune back to the primary component.

3.2.7.2 Example MSCs

Not Available

3.2.7.3 Error MSCs

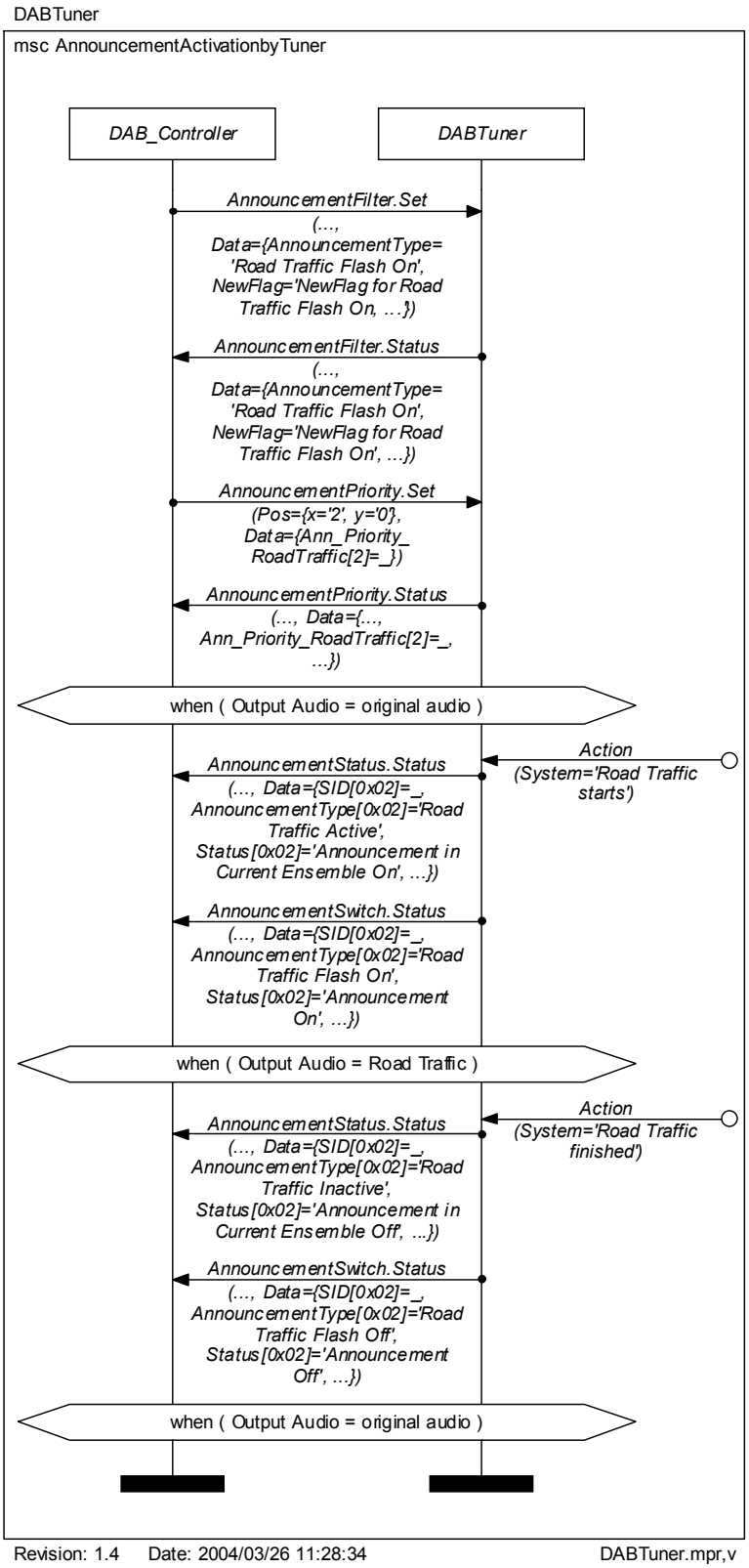
Not Available

3.3 Extensions

3.3.1 AnnouncementActivationbyTuner

Use Case:	AnnouncementActivationbyTuner		
Description:	This use case describes the configuration for the priority management of Announcements in the DAB tuner.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X		
Remarks:			

3.3.1.1 General MSCs



MSC 7: AnnouncementActivationbyTuner

3.3.1.2 Example MSCs

Not Available

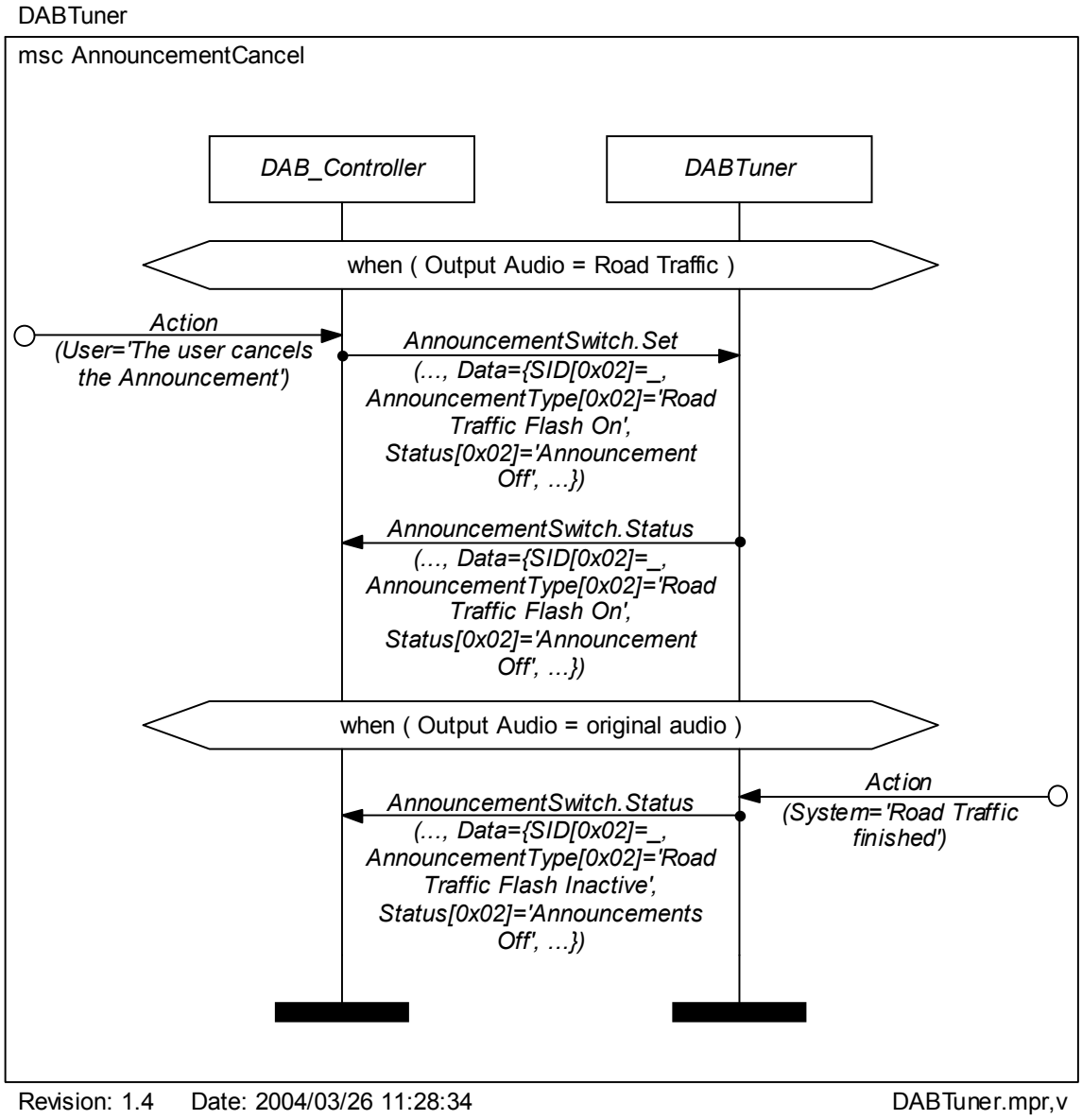
3.3.1.3 Error MSCs

Not Available

3.3.2 Announcement Cancel

Use Case:	Announcement Cancel		
Description:	The user cancels a running announcement.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X		
Remarks:			

3.3.2.1 General MSCs



MSC 8: Announcement Cancel

3.3.2.2 Example MSCs

Not Available

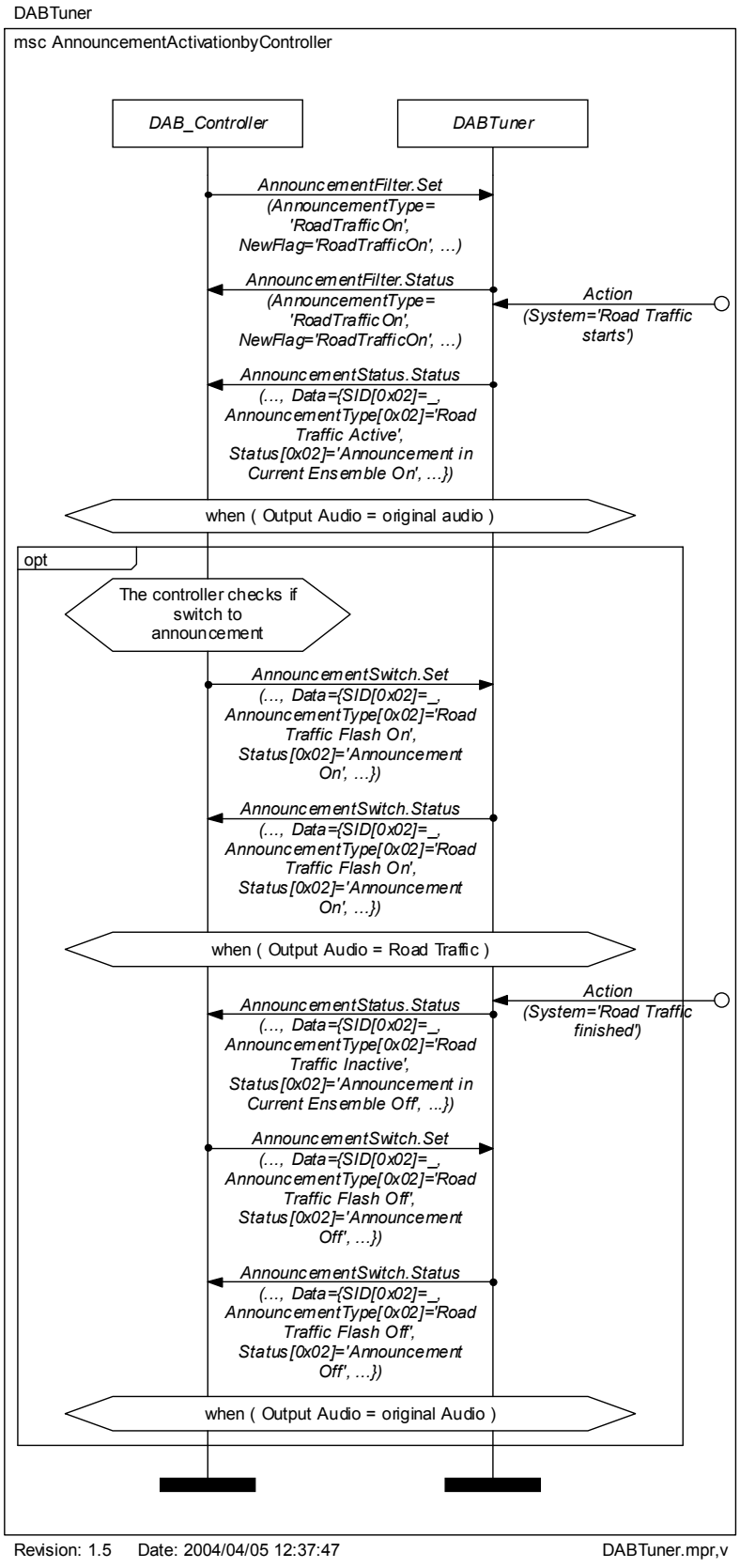
3.3.2.3 Error MSCs

Not Available

3.3.3 AnnouncementActivationbyController

Use Case:	AnnouncementActivationbyController		
Description:	This use case describes the configuration for the priority management of Announcements in the controller.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X		
Remarks:			

3.3.3.1 General MSCs



MSC 9: AnnouncementActivationbyController

3.3.3.2 Example MSCs

Not Available

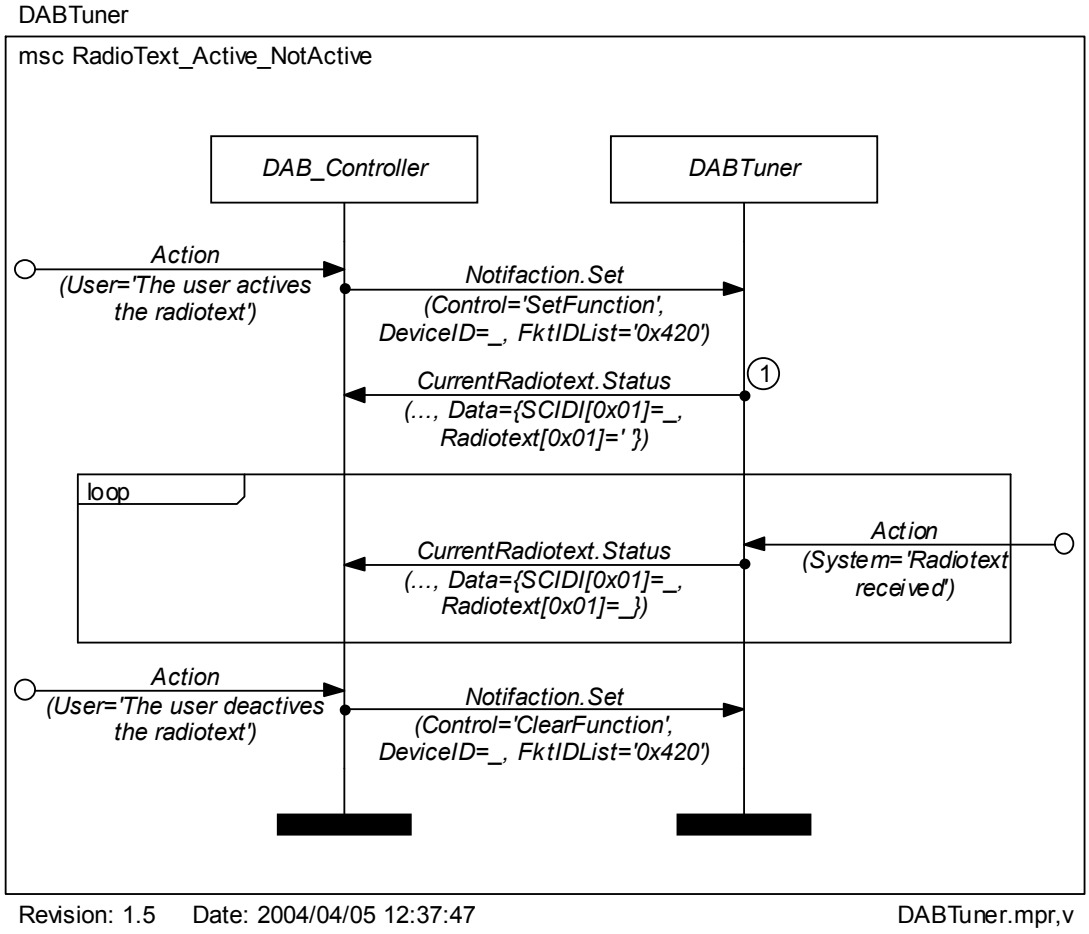
3.3.3.3 Error MSCs

Not Available

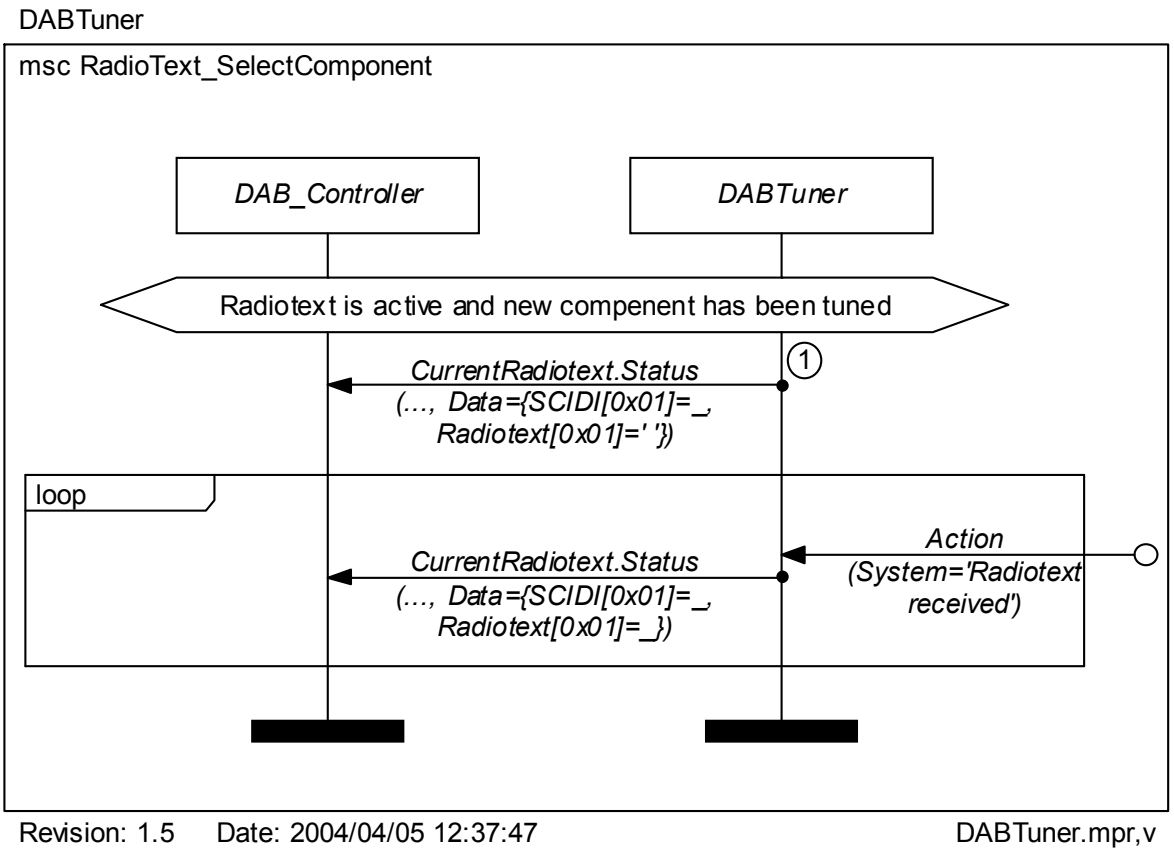
3.3.4 Radiotext

Use Case:	Using Radiotext		
Description:	Radiotext of the current selected ServiceComponent(s). An empty string deletes the current displayed radiotext.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X		
Remarks:			

3.3.4.1 General MSCs



MSC 10: Radiotext Active Not Active



MSC 11: Radiotext New Component

3.3.4.2 Example MSCs

Not Available

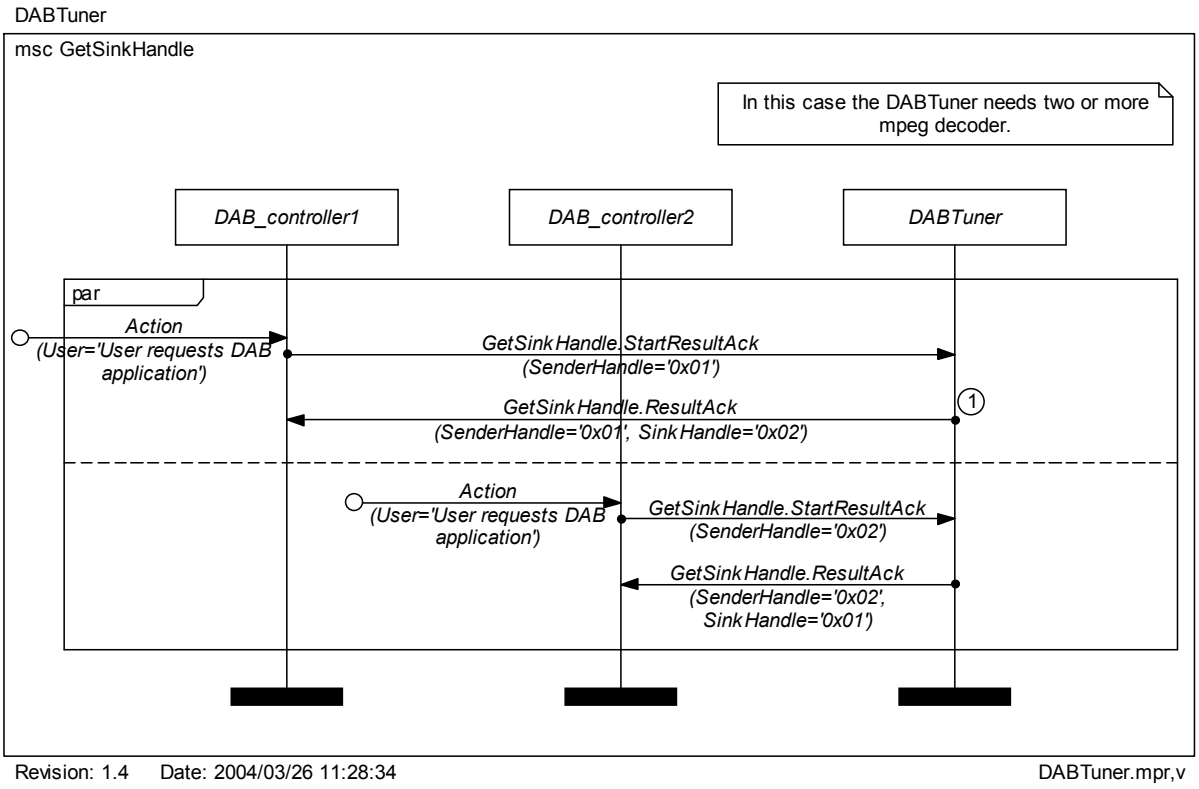
3.3.4.3 Error MSCs

Not Available

3.3.5 GetSinkHandle

Use Case:	GetSinkHandle		
Description:	The DAB_controller (e.g., MMI) asks for the SourceNr, the DAB Tuner will use to stream synchronous data onto the MOST Network.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:			

3.3.5.1 General MSCs



MSC 12: GetSinkHandle

1 The DABTuner gives the Controller a unique identifier.

3.3.5.2 Example MSCs

Not Available

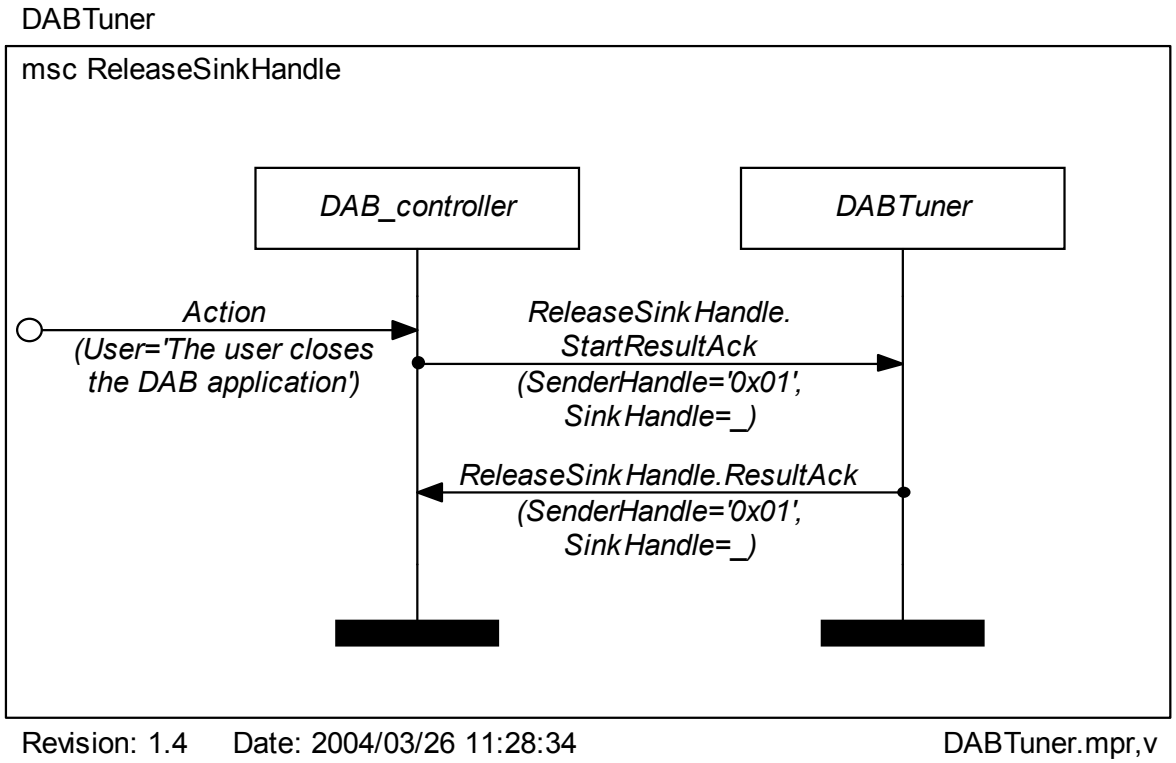
3.3.5.3 Error MSCs

Not Available

3.3.6 ReleaseSinkHandle

Use Case:	ReleaseSinkHandle		
Description:	The DAB_controller (e.g., MMI) releases the SinkHandle.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:			

3.3.6.1 General MSCs



MSC 13: ReleaseSinkHandle

3.3.6.2 Example MSCs

Not Available

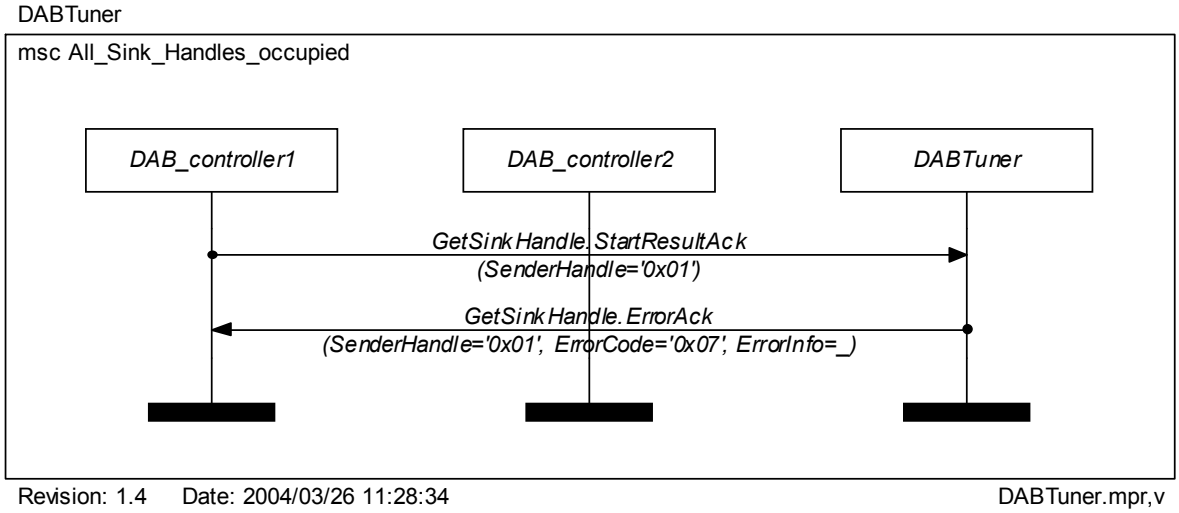
3.3.6.3 Error MSCs

Not Available

3.3.7 All Sink Handles occupied

Use Case:	All Sink Handles occupied		
Description:	If no more Handles are available, the DABTuner answers GetSinkHandle.ErrorAck.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:			

3.3.7.1 General MSCs



MSC 14: All Sink Handles occupied

3.3.7.2 Example MSCs

Not Available

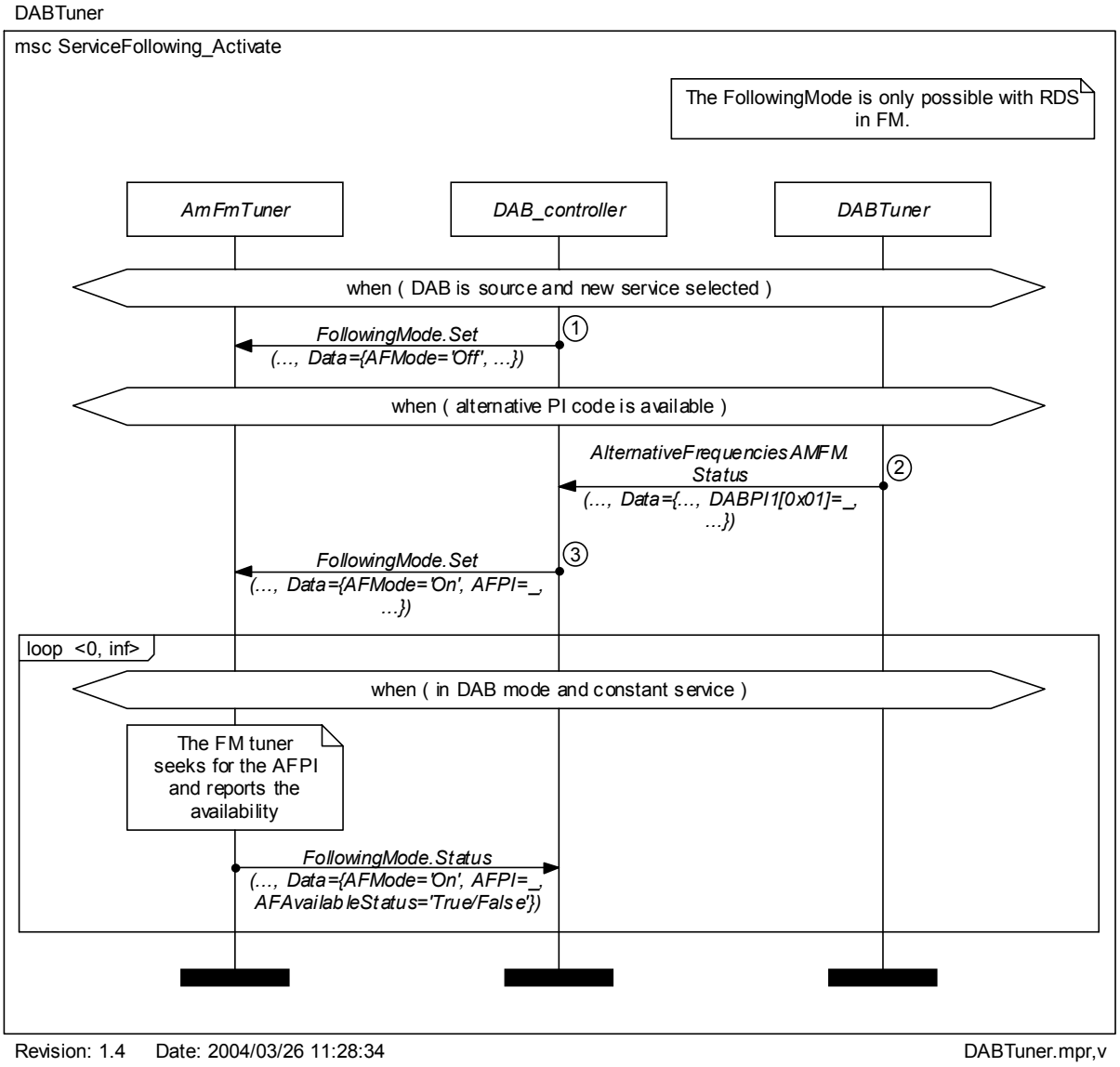
3.3.7.3 Error MSCs

Not Available

3.3.8 ServiceFollowing Activate

Use Case:	ServiceFollowing Activate		
Description:	The DAB_controller activates the ServiceFollowing in the tuner.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:			

3.3.8.1 General MSCs



MSC 15: ServiceFollowing Activate

1 After a new service has been selected, the DAB-controller switches off the AFMode of the FM tuner (The AFMode is active when the FM tuner is seeking for a certain PI-Code).

2 The DAB module decodes the alternative FM PI-Code (AFPI) corresponding to the currently tuned service. The DAB module sends the AFPI to the DAB controller via notification.

3 The DAB controller forwards the AFPI to the FM-tuner and activates the AFMode of the FM-tuner. Then the FM tuner seeks AFPI (DAB is still the source, processing in the background) and sends a message back to the DAB-controller to report the availability of the AFPI. AFMode is now activated.

3.3.8.2 Example MSCs

Not Available

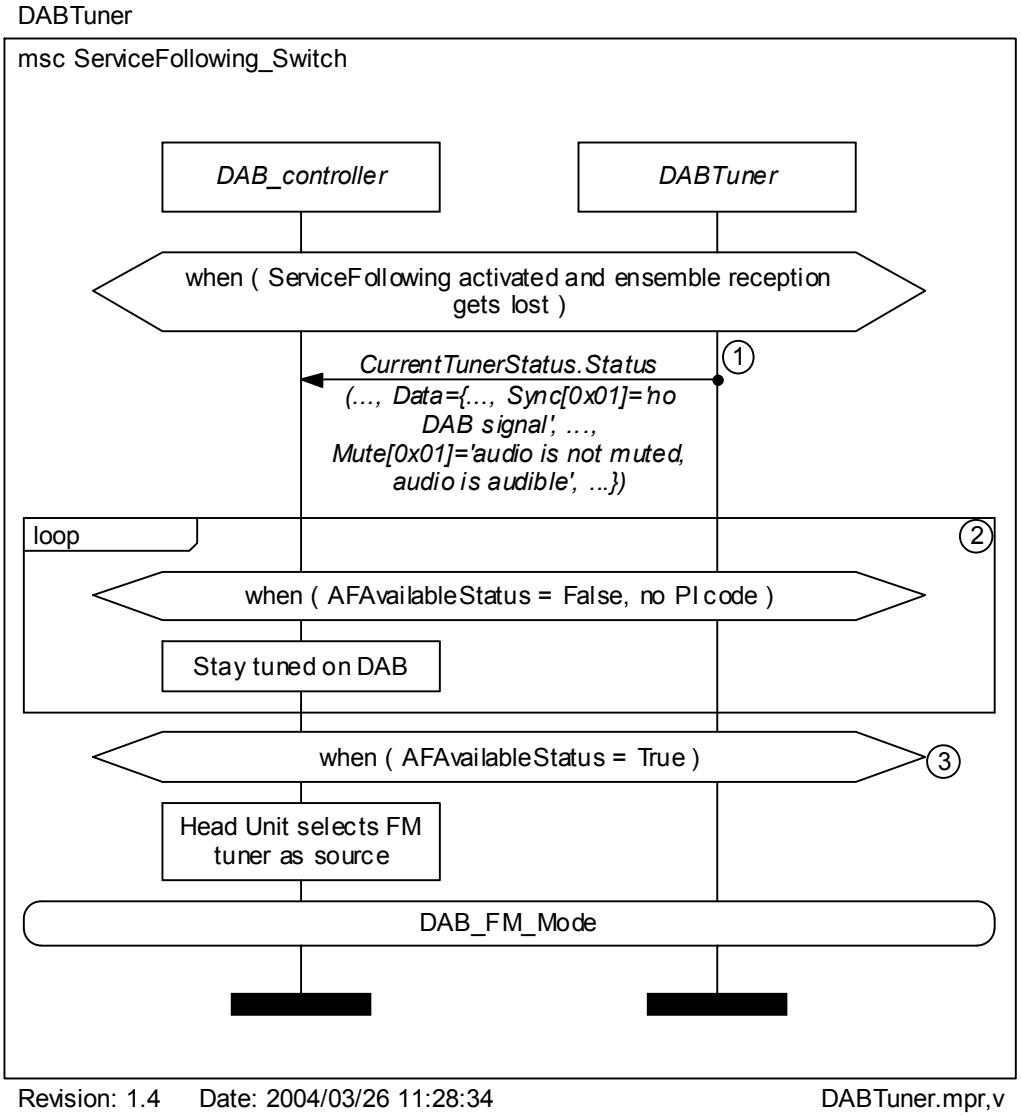
3.3.8.3 Error MSCs

Not Available

3.3.9 ServiceFollowing Switch

Use Case:	ServiceFollowing Switch		
Description:	If the ensemble signal gets lost (e.g., when leaving the coverage of a DAB transmitter), the DAB module mutes and informs the DAB_controller that the reception of the current service is not possible. If the FM tuner reports AFPI availability (AFAvailableStatus=True), the DAB_controller selects the FM tuner as new source.		
Prior Condition:	The tuner is in alternative FM mode because the DAB service was lost.		
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:			

3.3.9.1 General MSCs



MSC 16: ServiceFollowing Switch

1 If the ensemble signal gets lost (e.g., when leaving the coverage of a DAB transmitter), the DAB module mutes and informs the DAB-controller that the reception of the current service is not possible.

2 As long as the AFPI station is not available (AFAvailableStatus=False), the DAB module stays tuned on the current frequency.

3 If the FM tuner reports AFPI availability (AFAvailableStatus=True), the DAB-controller selects the FM tuner as source while the MMI still shows DAB display

3.3.9.2 Example MSCs

Not Available

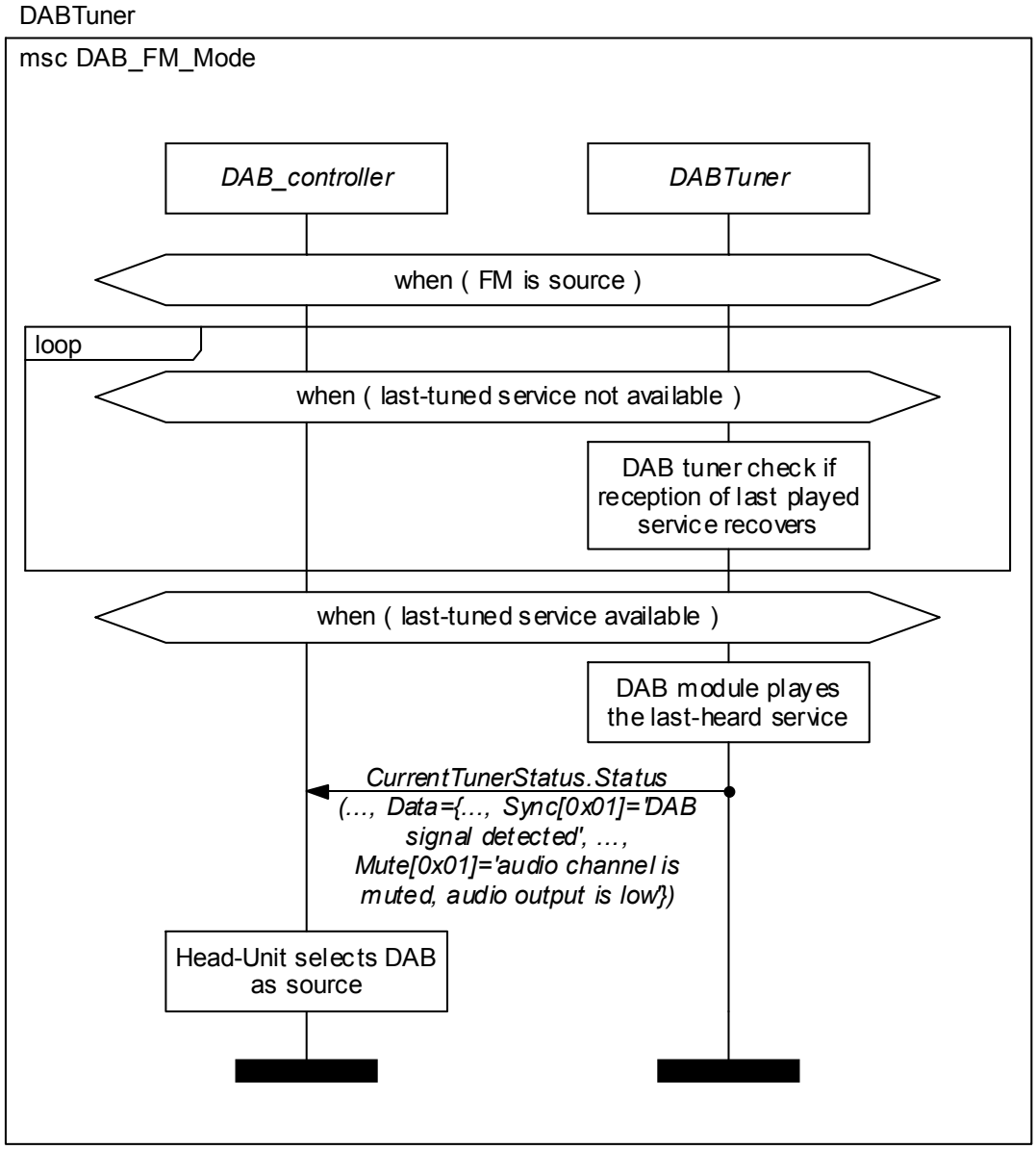
3.3.9.3 Error MSCs

Not Available

3.3.10 DAB FM Mode

Use Case:	DAB FM Mode		
Description:	As soon as the DAB service is available, the DAB Tuner is selected as source again.		
Prior Condition:	The tuner is in alternative FM mode because the DAB service was lost.		
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:			

3.3.10.1 General MSCs



MSC 17: DAB FM Mode

3.3.10.2 Example MSCs

Not Available

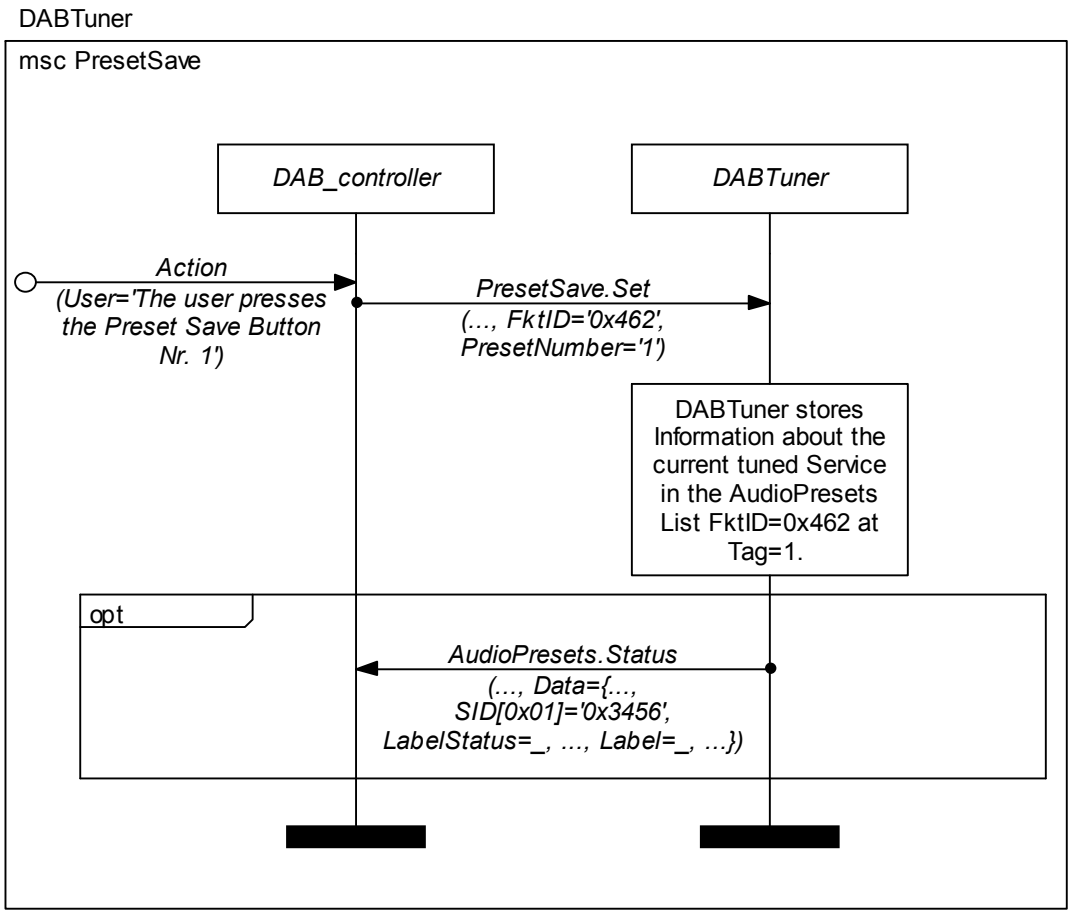
3.3.10.3 Error MSCs

Not Available

3.3.11 PresetSave

Use Case:	PresetSave		
Description:	The user chooses the Preset Button 1 to store the currently tuned service as a preset.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X		
Remarks:	Save the current service on the position PresetNumber in the list, which is given by the FktID. The PresetNumber is corresponding to the Tag in the preset list.		

3.3.11.1 General MSCs



MSC 18: PresetSave

3.3.11.2 Example MSCs

Not Available

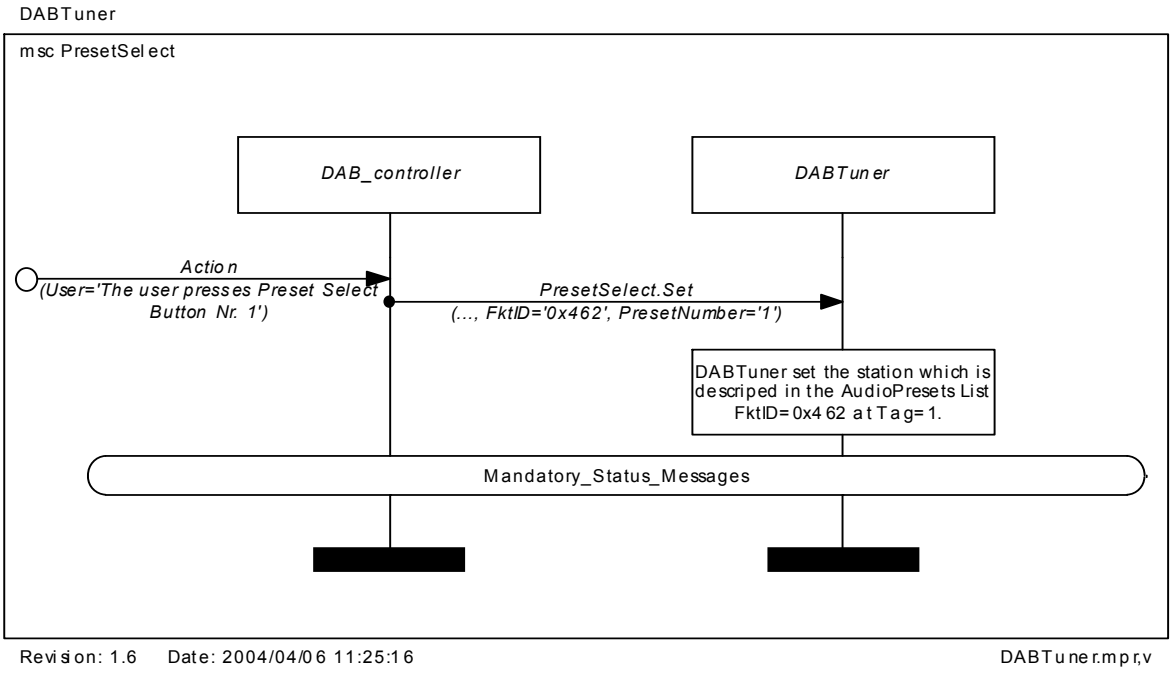
3.3.11.3 Error MSCs

Not Available

3.3.12 PresetSelect

Use Case:	PresetSelect		
Description:	The user presses the Preset Button 1 to select a service, which was previous stored as preset.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X		
Remarks:	Select the service, which is stored on the position PresetrNumber in the list, which is given by the FktID. The PresetNumber is corresponding to the Tag in the preset list.		

3.3.12.1 General MSCs



MSC 19: PresetSelect

3.3.12.2 Example MSCs

Not Available

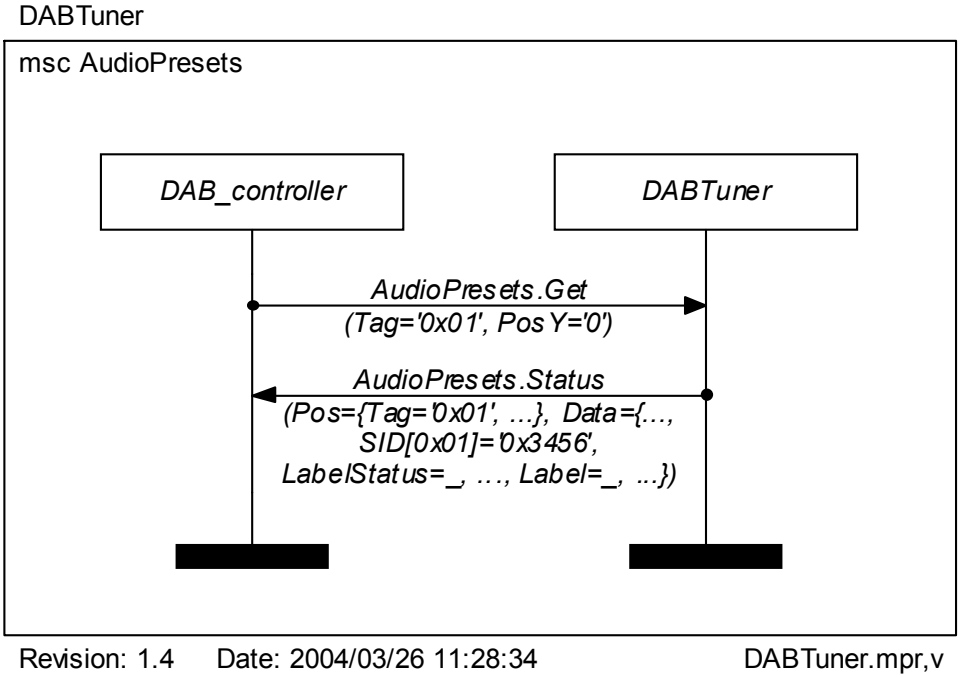
3.3.12.3 Error MSCs

Not Available

3.3.13 AudioPresets

Use Case:	AudioPreset		
Description:	Retrieve the information about a AudioPreset.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:	Different lists may be chosen by the reserved FktIDs of the AudioPresets lists.		

3.3.13.1 General MSCs



MSC 20: AudioPreset

3.3.13.2 Example MSCs

Not Available

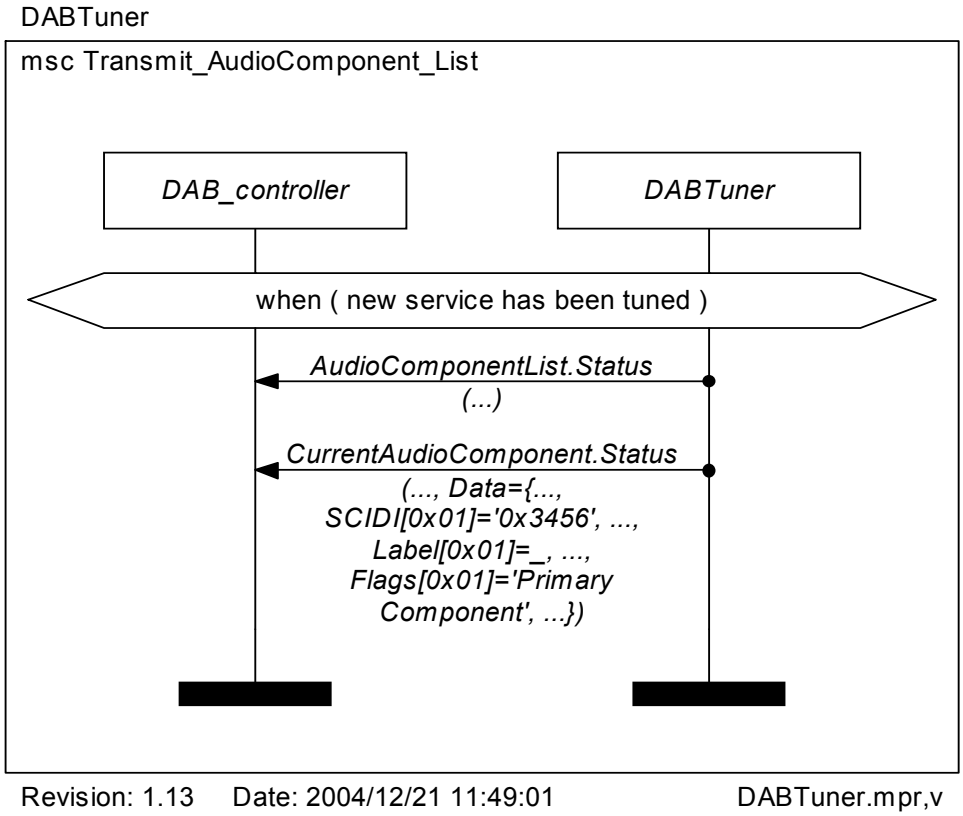
3.3.13.3 Error MSCs

Not Available

3.3.14 Transmit AudioComponent List

Use Case:	Transmit AudioComponent List		
Description:	The list informs the DAB_controller about the availability of audio service components.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:			

3.3.14.1 General MSCs



MSC 21: Transmit Component List

3.3.14.2 Example MSCs

Not Available

3.3.14.3 Error MSCs

Not Available

3.4 DataServices

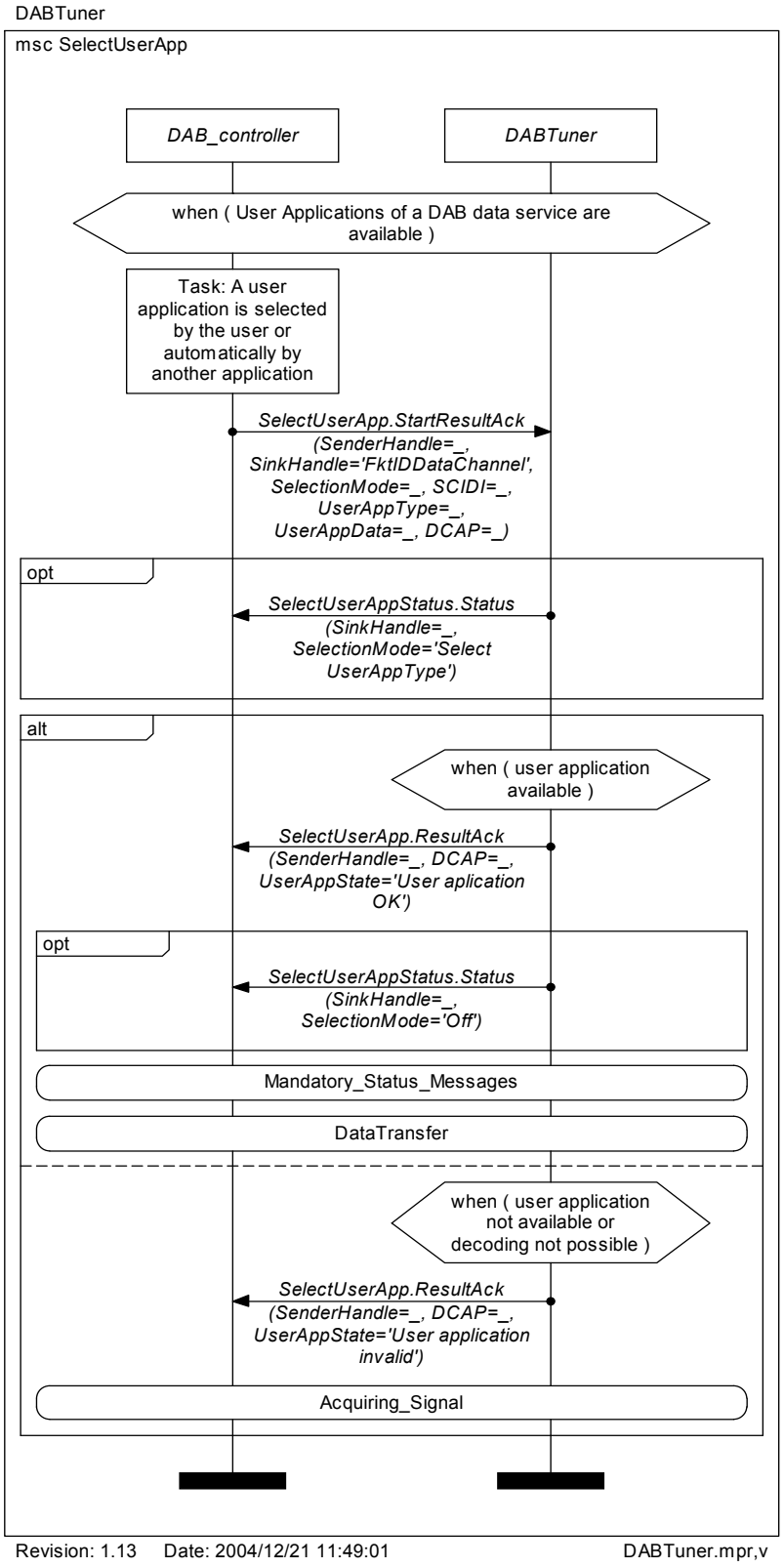
3.4.1 DataServices Notification Matrices

Use Case:	DataServices Notification Matrices		
Description:	For the functionality DataServices it's required that the listed devices are registered in the Notification matrices for the following functions.		
	DABTuner.0x00.CurrentEnsemble		DAB_controller
	DABTuner.0x00.CurrentDataService		DAB_controller
	DABTuner.0x00.CurrentDataComponent		DAB_controller
	DABTuner.0x00.CurrentDataUserApp		DAB_controller
	DABTuner.0x00.EnsembleList		DAB_controller
	DABTuner.0x00.DataServiceList		DAB_controller
	DABTuner.0x00.DataComponentList		DAB_controller
	DABTuner.0x00.DataUserAppList		DAB_controller
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:			

3.4.2 Select User Application

Use Case:	**Select User Application**		
Description:	Selection of a specific data service must be seen against the DAB data decoder capability. Only a high-level decoder is able to provide the service components and specific user applications. That means, the selection commands immerse from ensemble level down to user application level depending on the decoder capability. The data stream is in fact first transmitted on the lowermost level.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X	(X)	
Remarks:			

3.4.2.1 General MSCs



MSC 22: Select User Application

3.4.2.2 Example MSCs

Not Available

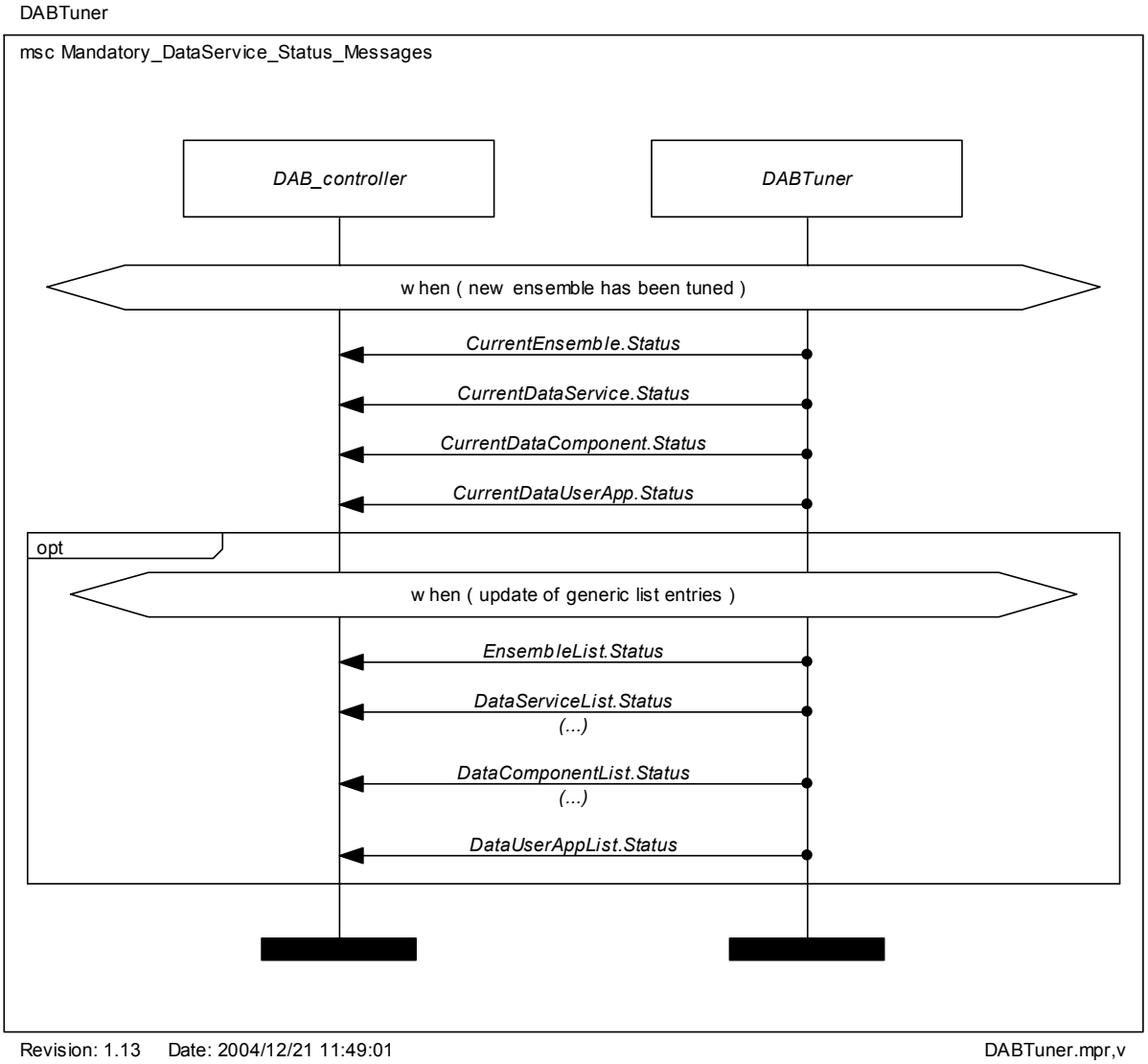
3.4.2.3 Error MSCs

Not Available

3.4.3 Mandatory DataService Status Messages

Use Case:	Mandatory DataService Status Messages		
Description:	If a new ensemble is tuned, these lists will be updated.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:			

3.4.3.1 General MSCs



MSC 23: Mandatory Dataservice Status Messages

3.4.3.2 Example MSCs

Not Available

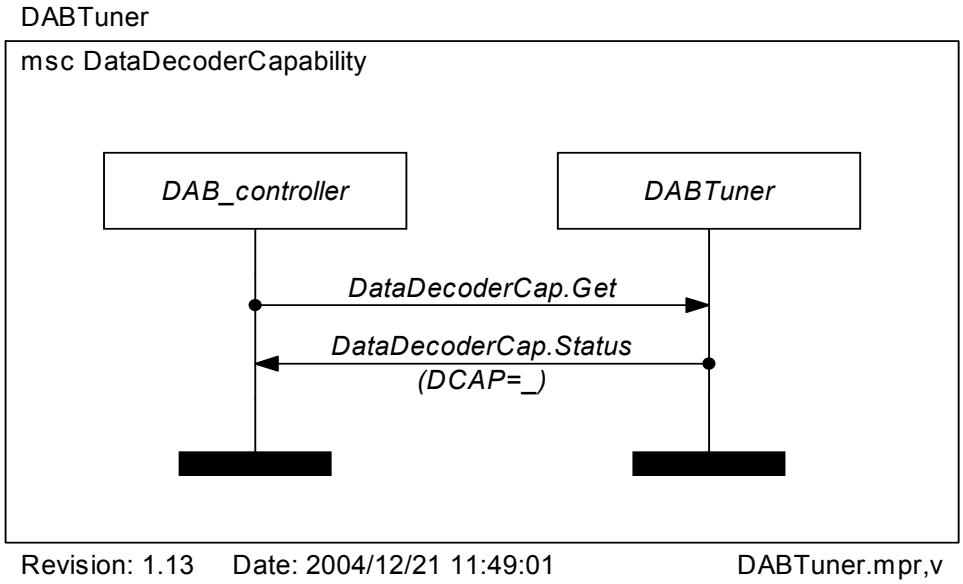
3.4.3.3 Error MSCs

Not Available

3.4.4 Data Decoder Capability

Use Case:	Data Decoder Capability		
Description:	To provide data services from the DAB stream, the DAB Tuner needs a DAB data decoder. The final decoding can be done by the DAB Tuner, but also by the sink application. Thus, different level of decoding capability and calculating capacity must be considered. This function gives information about the decoding capability of the DAB Tuner.		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
	X		
Remarks:			

3.4.4.1 General MSCs



MSC 24: Data decoder capability

3.4.4.2 Example MSCs

Not Available

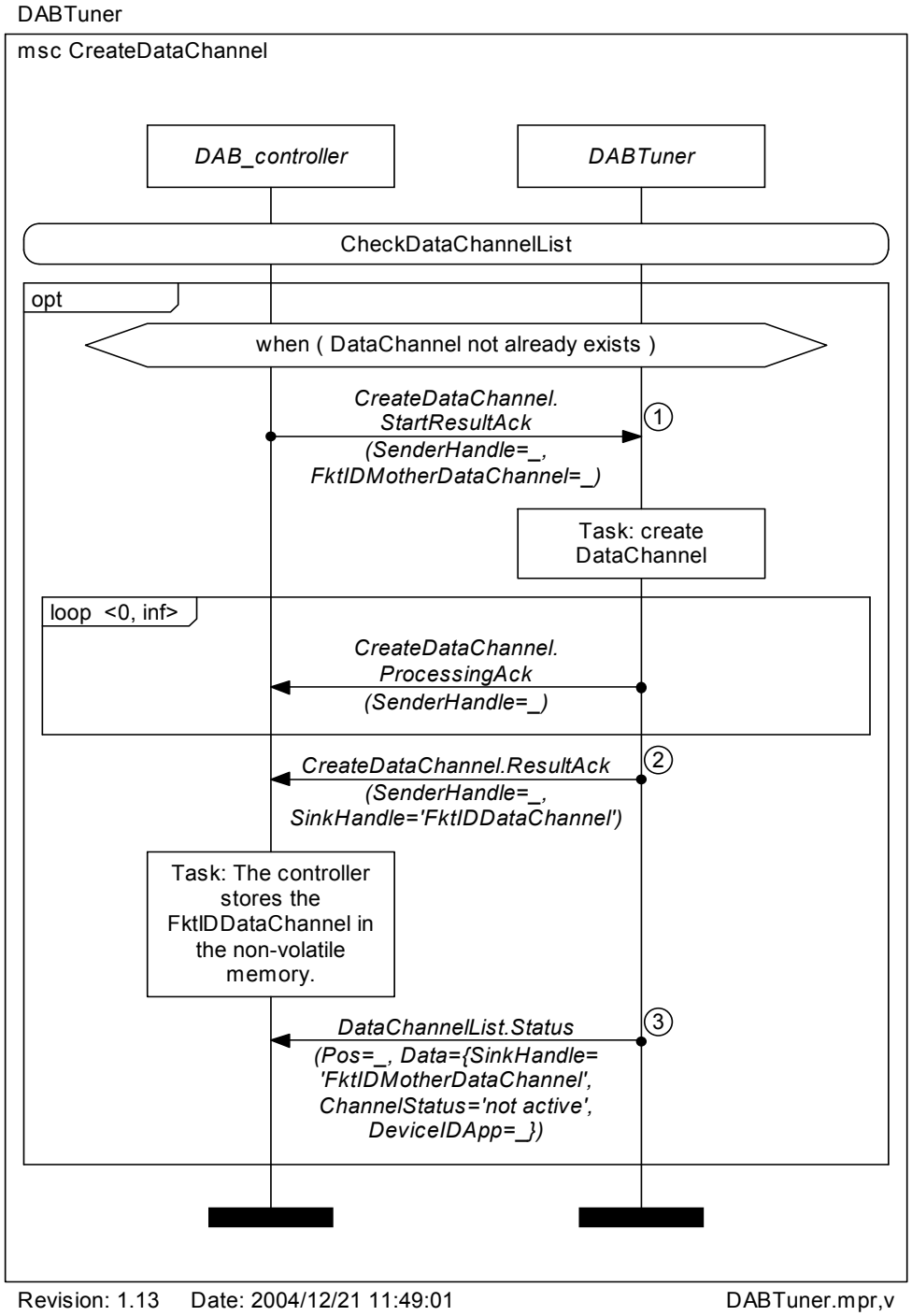
3.4.4.3 Error MSCs

Not Available

3.4.5 Create Data Channel

Use Case:	CreateDataChannel		
Description:	To transfer the data stream over MOST-High protocol or control channel the controller instantiates a data channel with the method CreateDataChannel (if not already existing). The DAB Tuner creates the function and the application is notified automatically on this new data channel. Now, data is transferred to the application via status messages. MotherFunctionID contains the FktID of the respective sample function (different kinds of data channels are possible).		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
		X	Controller
Remarks:			

3.4.5.1 General MSCs



MSC 25: CreateDataChannel

1 Creating a function dynamically implies its notification. This is necessary to enable the application to receive data by the status message of the new DataChannel function.

2 A new function DataChannel is dynamically created. The SinkHandle contains the FktID of the DataChannel to identify the data output in the DAB-Tuner.

3 The new channel is added to DataChannelList. DataChannelList itemizes every dynamically instantiated DataChannel function. By default, the new channel is set to "not active" due to priority handling. This status can be changed by SetDataChannelStatusActive.

3.4.5.2 Example MSCs

Not Available

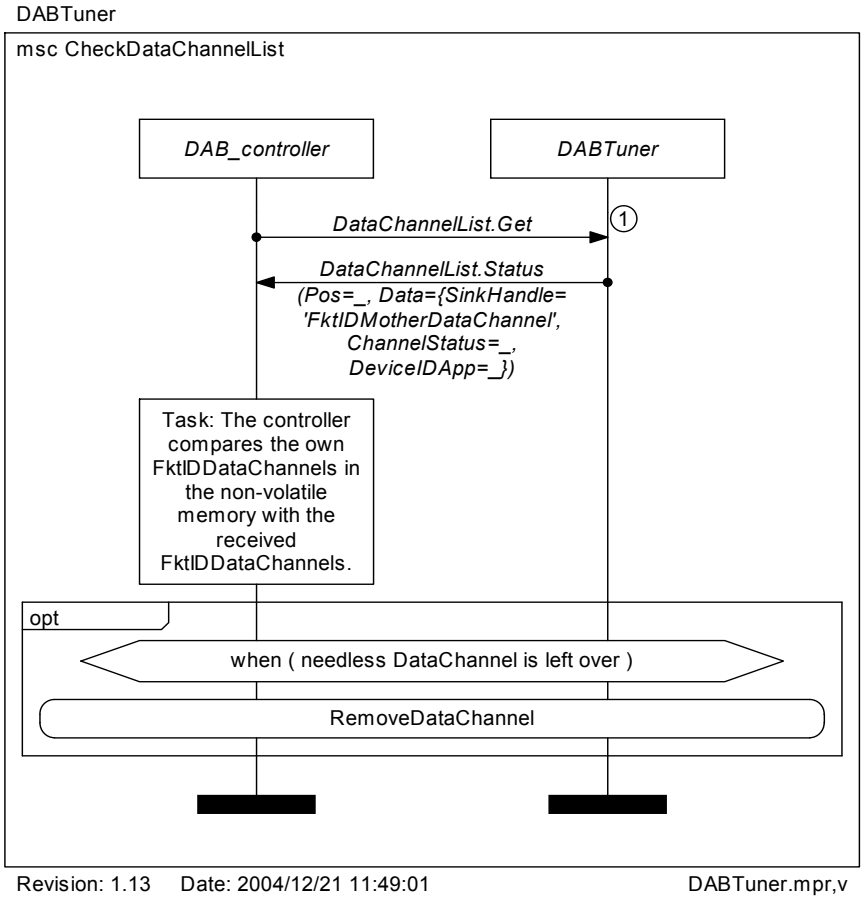
3.4.5.3 Error MSCs

Not Available

3.4.6 Check DataChannelList

Use Case:	CheckDataChannelList		
Description:	The controller compares its own information with the received DataChannelList data.		
Prior Condition:			
Initiator:	Passenger	Internal	Comment
		X	Controller
Remarks:			

3.4.6.1 General MSCs



MSC 26: Check DataChannelList

1 Before creating a new DataChannel function, the control application should ensure that there exists not already a previously instantiated function for this task. This is checked via DataChannelList.

3.4.6.2 Example MSCs

Not Available

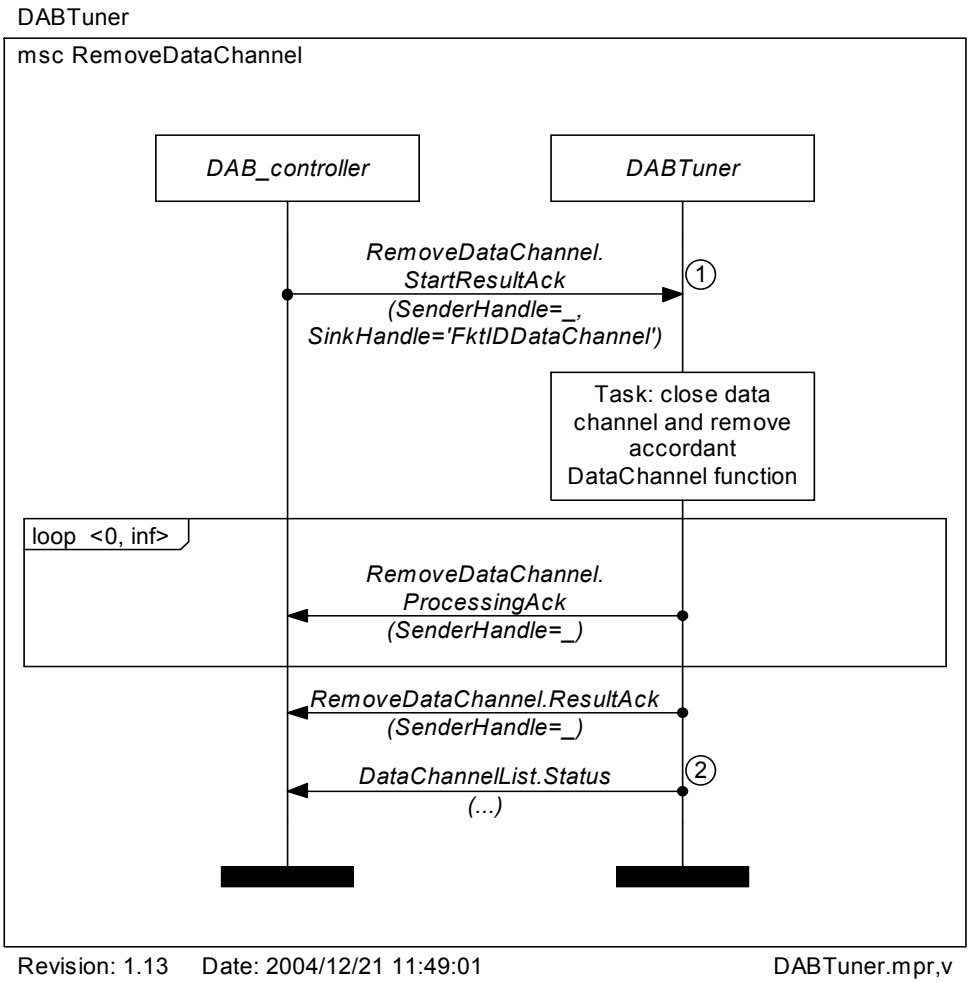
3.4.6.3 Error MSCs

Not Available

3.4.7 Remove Data Channel

Use Case:	RemoveDataChannel		
Description:	A controller removes a DataChannel with the method RemoveDataChannel.		
Prior Condition:	Data Channels are existing		
Initiator:	**Passenger**	**Internal**	**Comment**
		X	Controller
Remarks:			

3.4.7.1 General MSCs



MSC 27: Remove DataChannel

1 Removing a dynamic function implies its notification to be deleted.
2 The accordant entry of this SinkHandle is deleted in the DataChannelList. A new update of the list is sent to every notified application.

3.4.7.2 Example MSCs

Not Available

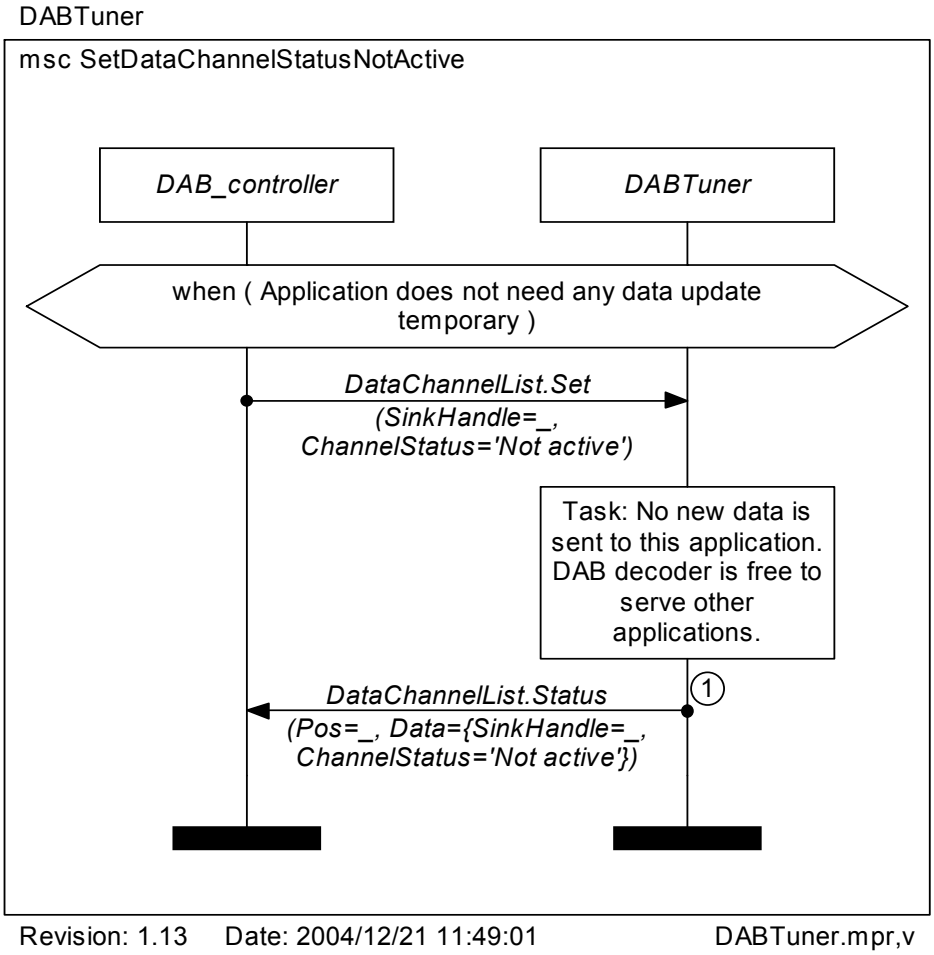
3.4.7.3 Error MSCs

Not Available

3.4.8 Set Data Channel not Active

Use Case:	Set Data Channel not Active		
Description:	An application with highest priority releases the DAB decoder to serve other applications.		
Prior Condition:	Data channel is active		
Initiator:	**Passenger**	**Internal**	**Comment**
		X	Controller
Remarks:	Look at showcase in *MSC 30: showcase priority handling*		

3.4.8.1 General MSCs



MSC 28: Change of Application

1 DataChannel is only paused, not removed

3.4.8.2 Example MSCs

Not Available

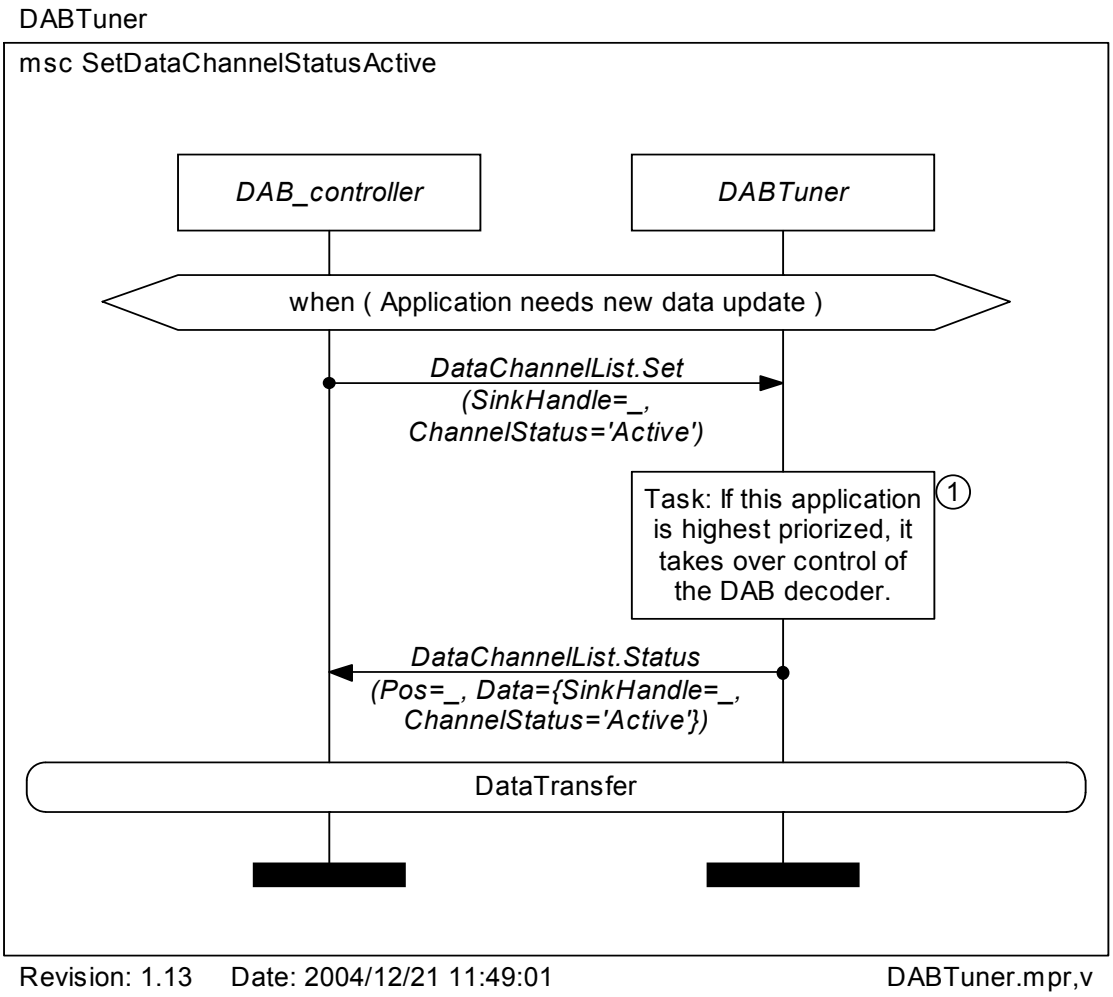
3.4.8.3 Error MSCs

Not Available

3.4.9 Set Data Channel Active

Use Case:	Set Data Channel active		
Description:	Highest priotized application takes over control of decoder again.		
Prior Condition:	Data channel to former application is active		
Initiator:	**Passenger**	**Internal**	**Comment**
		X	Controller
Remarks:	Look at showcase in *MSC 30: showcase priority handling*		

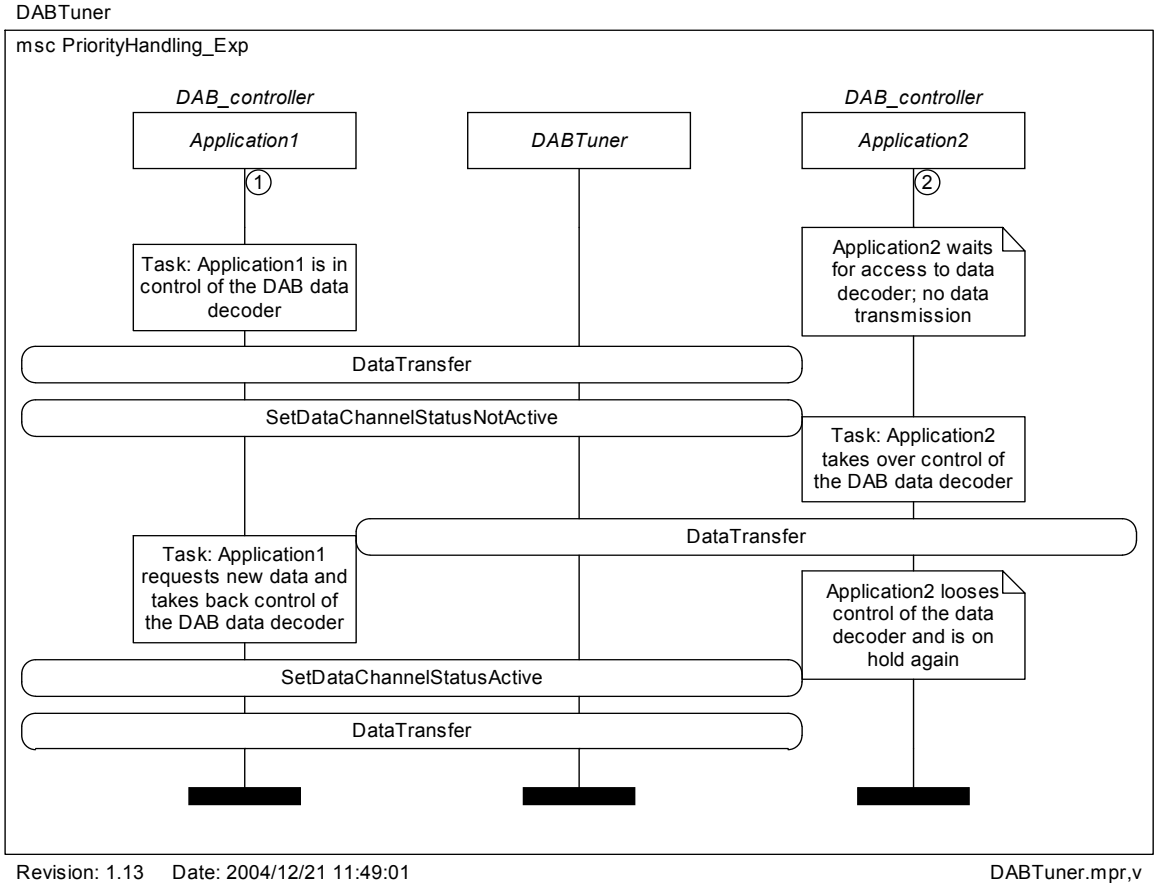
3.4.9.1 General MSCs



MSC 29: Set Data Channel active

1 Current active applications are paused

3.4.9.2 Example MSCs



MSC 30: showcase priority handling

1 Highest priorized application controls the DAB Tuner.
2 Application2 is lower priorized than application1

3.4.9.3 Error MSCs

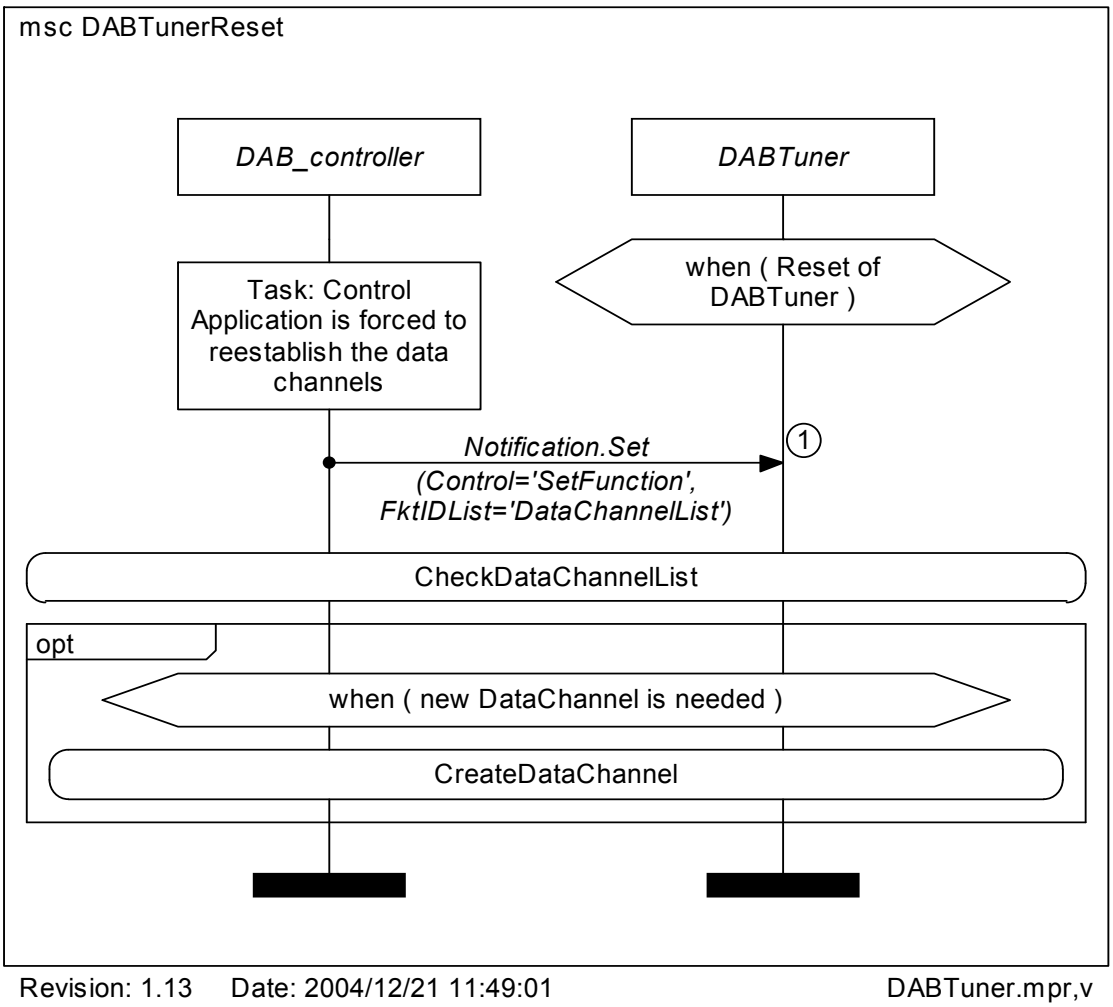
Not Available

3.4.10 DAB Tuner Reset

Use Case:	DAB Tuner Reset		
Description:	Behavior of the control application in case of a reset of the DAB Tuner.		
Prior Condition:	DAB Tuner reset		
Initiator:	**Passenger**	**Internal**	**Comment**
		X	Controller
Remarks:			

3.4.10.1 General MSCs

DABTuner



MSC 31: DAB Tuner Reset

1 Notification on dynamic functions is set automatically for the requesting device. In case of a device reset, the notification must be set manually again.

3.4.10.2 Example MSCs

Not Available

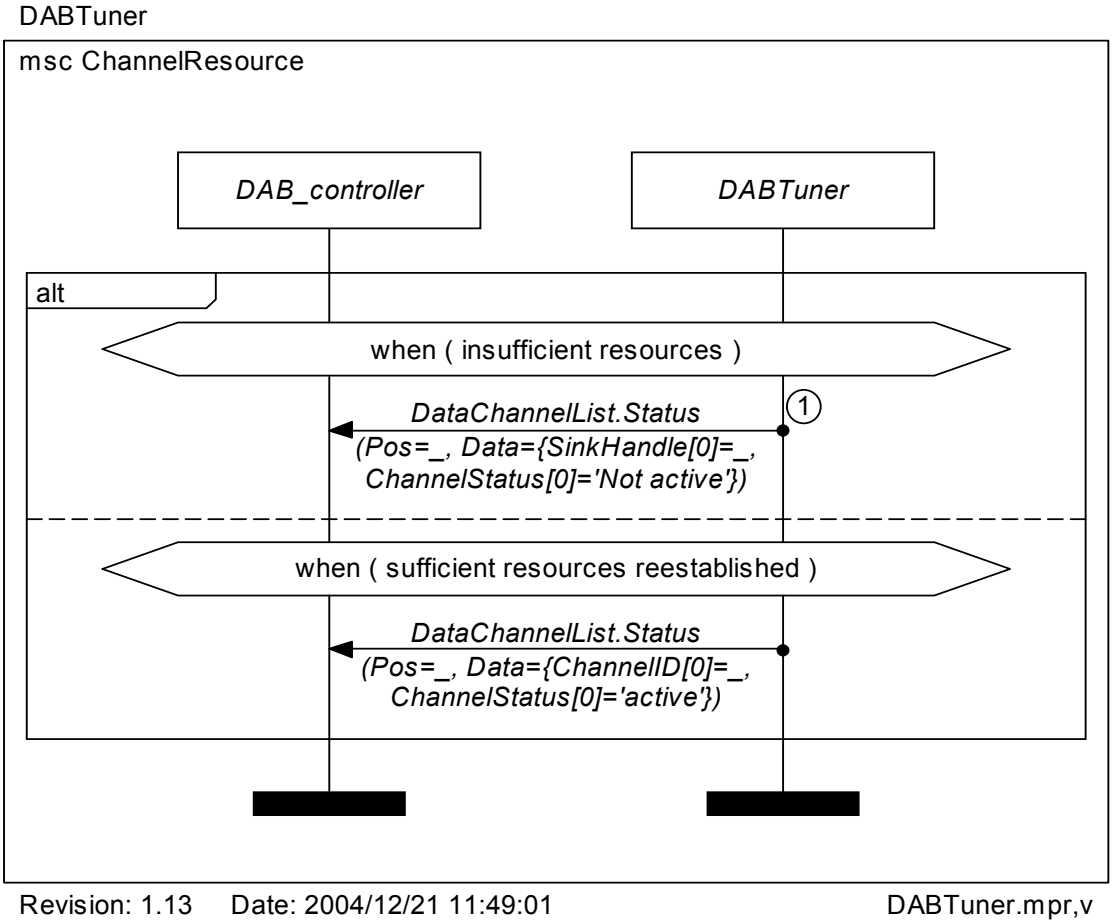
3.4.10.3 Error MSCs

Not Available

3.4.11 Channel Resource

Use Case:	ChannelResource		
Description:	Channel resources have changed. Possible reasons for this: 1. A Data channel is established or closed. 2. An active connection is interrupted. 3. An interrupted connection is reestablished again. 4. MOSTHigh overload (too many channels open).		
Prior Condition:			
Initiator:	**Passenger**	**Internal**	**Comment**
		X	DAB Data Services
Remarks:			

3.4.11.1 General MSCs



MSC 32: ChannelResource

1 If connection is not completely lost, the SinkHandle connections are only interrupted but not removed.

3.4.11.2 Example MSCs

Not Available

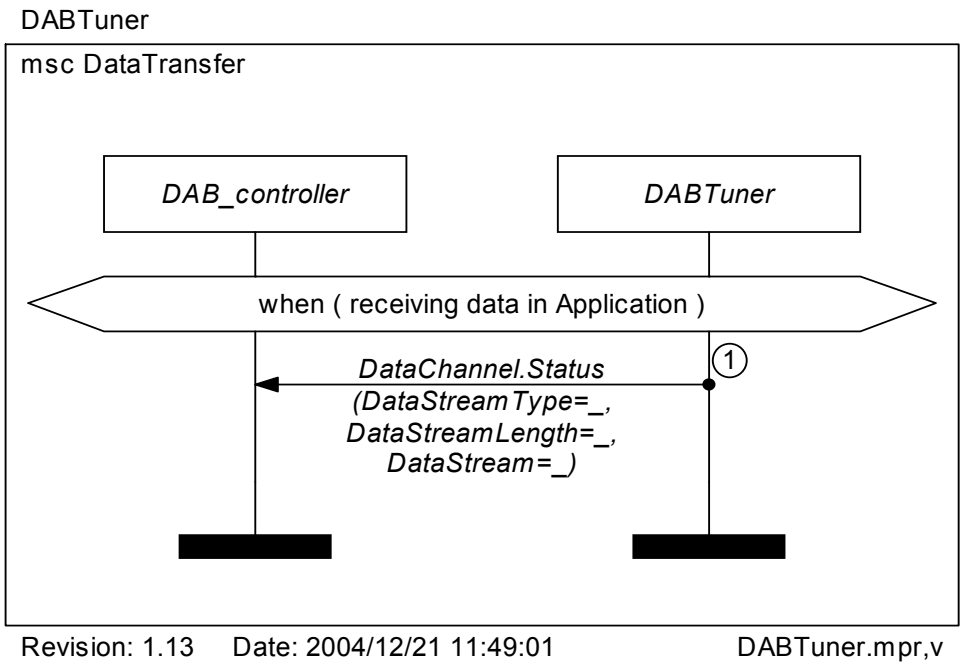
3.4.11.3 Error MSCs

Not Available

3.4.12 Data Transfer

Use Case:	DataTransfer		
Description:	DAB Data transfer via Data channel using the MOST-High protocol over the asynchronous channel or control channel		
Prior Condition:	Data channel is open		
Initiator:	**Passenger**	**Internal**	**Comment**
		X	
Remarks:			

3.4.12.1 General MSCs



MSC 33: DataTransfer

1 Data is transmitted over the MOST High protocol or control channel via this function based on the function MotherDataChannel. The control application receives data via DataChannel.Status.

3.4.12.2 Example MSCs

Not Available

3.4.12.3 Error MSCs

Not Available

Notes: