PMeV
PROFESSIONELLER
MOBILFUNK e.V.

... mit Sicherheit besser!

„Digitalfunkstecker“ (DF-Stecker) Technical Description

Publication of AK BOS-Leitstellen (DF-Stecker V0.2)

April 24, 2012

Bundesverband Professioneller Mobilfunk (PMeV)

Table of Contents

Table of Figures

1 Introduction

This document is an API description containing the DF-Stecker contracts. Those contracts are applicable between a DF-Client (Control Room Equipement) and a DF-Server (Concentrator). The document also provides information about functionality affected by the API and rules to follow to support the API. This document is not a description of a DF-Server (Concentrator) product and stays out of what could be considered vendor specific products and functionalities.

For more information about Concentrator products, please refer to appropriate and specific product documentation.

2 GLOSSARY

Term	Definition
API	Application Programming Interface.
DF-Client	Client application of a Concentrator (DF Server).
DF-Server	Concentrator (from an API perspective).
G.711	G.711 is an ITU-T standard for audio companding.
IP	Internet Protocol.
ISSI	Individual Short Subscriber ID.
LAN	Local Area Network.
LIP	Location Information Protocol.
PCM	Pulse Code Modulation.
PCMA	Pulse Code Modulation A-Law format.
PCMU	Pulse Code Modulation µ-Law format.
PTT	Push To Talk.
RTP	Real Time Transport Protocol.
Rx	Receive/Receiving
SDP	Session Description Protocol.
SDS	Short Data Services.
SOAP	Simple Object Access Protocol.
SSI	Short Subscriber ID.
TCP	Transmission Control Protocol.
TCS	TETRA Connectivity Server.
TCS-Client	Client of the TETRA system from a TCS

	perspective.
TETRA	Terrestrial Trunked Radio.
Tx	Transmit/Transmission/Transmitting
UDP	User Datagram Protocol.
VoIP	Voice Over IP.
WAN	Wide Area Network.
WSDL	Web Service Definition Language
XML	Extensible Markup Language.
XSD	XML Schema Definition

3 APPLICABLE DOCUMENTS

1. Real-Time Transport Protocol (RTP) Payload Format for the TETRA Audio Codec
   Stefan Wenk stefan.wenk@frequentis.com

4 OVERVIEW

4.1 CONTRACTS OVERVIEW

Description

This document contains Web Service descriptions for the following services.

Web Services

Name
LoginService
LoginEventsService
TETRAResourceService
TETRAResourceEventsService
TetraMassiveMonitoringGroupService
TetraMassiveMonitoringGroupEventsService
TetraGroupService
TetraGroupEventsService
TetraIndividualCallService
TetraIndividualCallEventsService
TetraSdsService
TetraSdsEventsService
TetraUnitTrackingService
TetraUnitTrackingEventsService

4.2 API Workflow

The first step to accomplish for a DF-Client to establish a connection using the API is to procede with the login contract.

After a successful login, resources request could take place and all the supported contracts matching the resources granted are in place and any request method can be invoked. At the same time, any event can be received in an asynchronus manner.

The only requirement to maintain the session active is to successfully receive the KeepAliveEvent. Once a session is completed, to achieve a gracefull termination of the session, the logout method shall be invoked.

4.2.1 Concentrator Business Logic

From a business logic and state management point of view, the concentrator (DF-Server) from a TETRA functionality point of view is in most cases stateless.

It is stateless for most of the request and responses concerning the TETRA group communication and monitoring, individual calls, SDS and unit tracking. It could mainly be seen as a proxy server and a language translator.

Where the DF-Server is managing states is for resource management (ISSIs, B channels and M channels). The DF-Server is responsible to manage sessions, DF- Clients and resources distribution and the way those things are done are Concentrator product specific.

4.2.2 DF-Client session establishment and management

The DF-Client has to be provisioned with IP Addresses in order to establish a connection and a session to a DF-Server. Minimally, the DF-Client shall be provisioned with a single IP address.

Failures of the session could be the result of multiple types of failures. Failures could be the result of networking equipment failure or network equipment misconfiguration. They can also be related to firewalls failures or cabling failures or disconnection. There is also the possibility of uncontrolled delays or congestions at the network level within the LAN or at the WAN (problem with the network provider).

In the case where a DF-Client is loosing its session with the DF-Server for example by having a failure at the HTTP or TCP layer or by not receiving regularly the Keep Alive Event, the DF Client has to resend periodically a login request to that IP Address in order to restore the session and could do it indefinitely, until it receives a valid answer. It is then the responsibility of the DF-Client to restore the session by requesting resources and requesting group monitoring.

In the scenario where two DF-Server are available in a redundancy model either as active-active or active-standby, it is preferable for the DF-Client to be configured with both IP Addresses and in the case of a failure, the DF-Client in order to re-establish a session shall periodically send a login request alternating between the two IP Addresses indefinitely until it gets a valid positive response.

Finally, in the scenario where the Concentrator is build with multiple active DF-Server in a cluster environment, the same principle as in the redundancy model can be applied where after a failure, a DF-Client is selecting the DF-Server to connect to using a round-robin algorithm. Any other algorithm could be used by the client if the client has a preferred DF-Server based on resources it owns or based on geographical location. To achieve this selection algorithm, the DF-Client has to be provisioned with all the DF-Server IP Addresses.

4.2.3 Audio Distribution

As far as audio is concerned, the audio is always provided in an agreed encryption state and in an agreed codec over RTP unicast or multicast for the dedicated slots.

The following diagram represents the different type of TCS audio clients available:

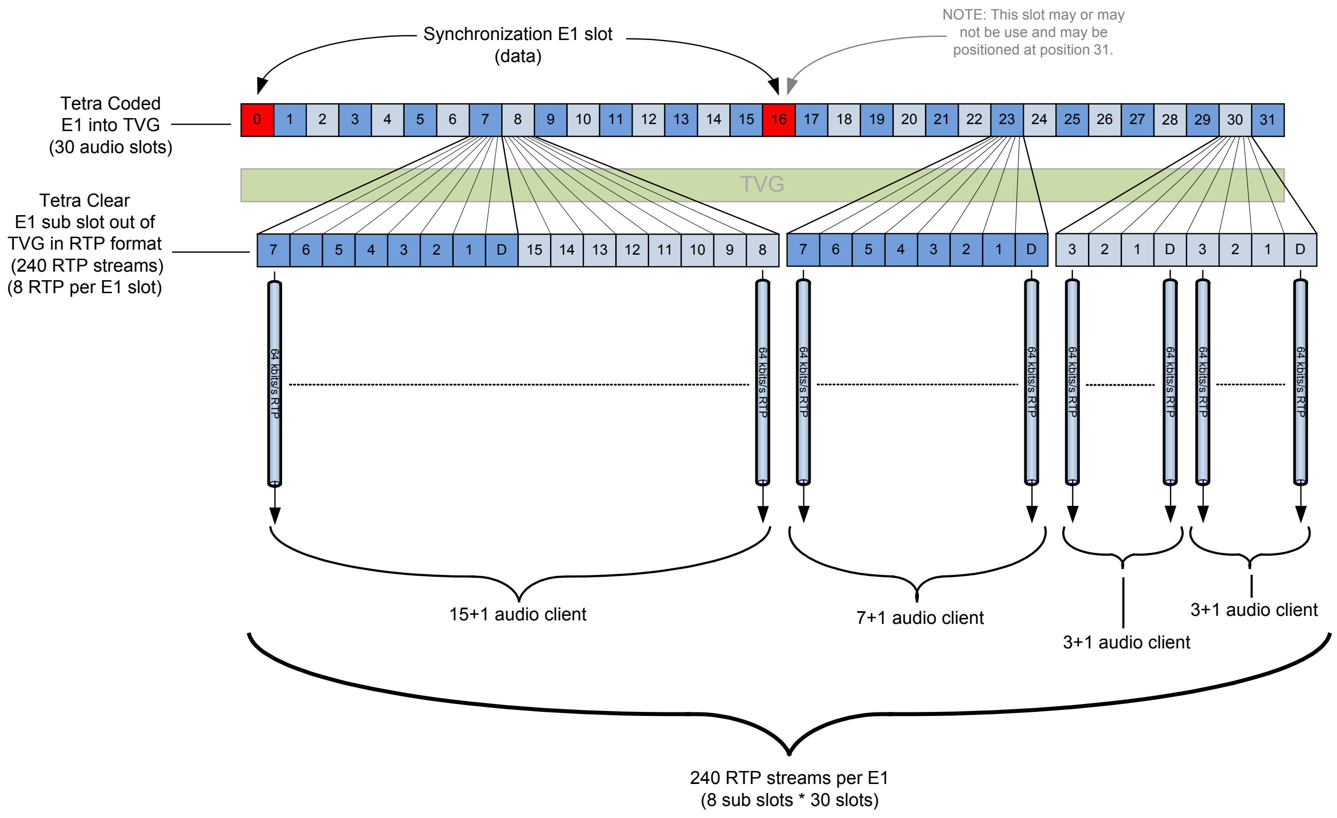

Synchronization E1 slot (data)
NOTE: This slot may or may not be use and may be positioned at position 31.
Tetra Coded E1 into TVG (30 audio slots)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
TVG
Tetra Clear E1 sub slot out of TVG in RTP format (240 RTP streams) (8 RTP per E1 slot)
7 6 5 4 3 2 1 D 15 14 13 12 11 10 9 8
7 6 5 4 3 2 1 D
3 2 1 D 3 2 1 D
64 kbit/s RTP
15+1 audio client
7+1 audio client
3+1 audio client
3+1 audio client
240 RTP streams per E1 (8 sub slots * 30 slots)

Figure 1 - Audio Distribution

4.2.4 B Channel

B channel is the term used to refer to a specific portion of a TCS Client. In this case we refer to the bi directional time slot (the +1 in the X + 1 description) and the first unidirectional time slot. It also includes the concept of identity meaning that the ISSI and OPTA are also attached to that B channel even if in some cases they are not required.

Note: The DF-Stecker API is not preventing a Concentrator product to offer more time slot part of the B channel resource even though it is not the original intention.

4.2.5 M Channel

M channel is a unidirectional time slot of a TCS Client used to only passively monitor a group audio. For example, in the case of a 1+15 TCS Client, we will have 1 B channel and 14 M channels.

4.2.6 Talker audio limitation

In the TETRA system, there is a limitation to take into consideration when breaking the model of one TCS Client for each user which is the fact that if you are transmitting on a group that you are monitoring, you will not receive your own audio on that group.

It is also not possible to monitor a group on a B and M channels that are part of the same TCS Client.

The result is that if you are talking on a group with a B Channel, you will not get that audio back but if you are also monitoring that group with a M Channel (of a different TCS Client), you will be able to get you audio back through that path.

The next set of diagrams is demonstrating API state management from the DF-Client point of view.

4.2.7 SIMPLE RESOURCE REQUEST WORKFLOW (NO SPECIFIC ISSI)

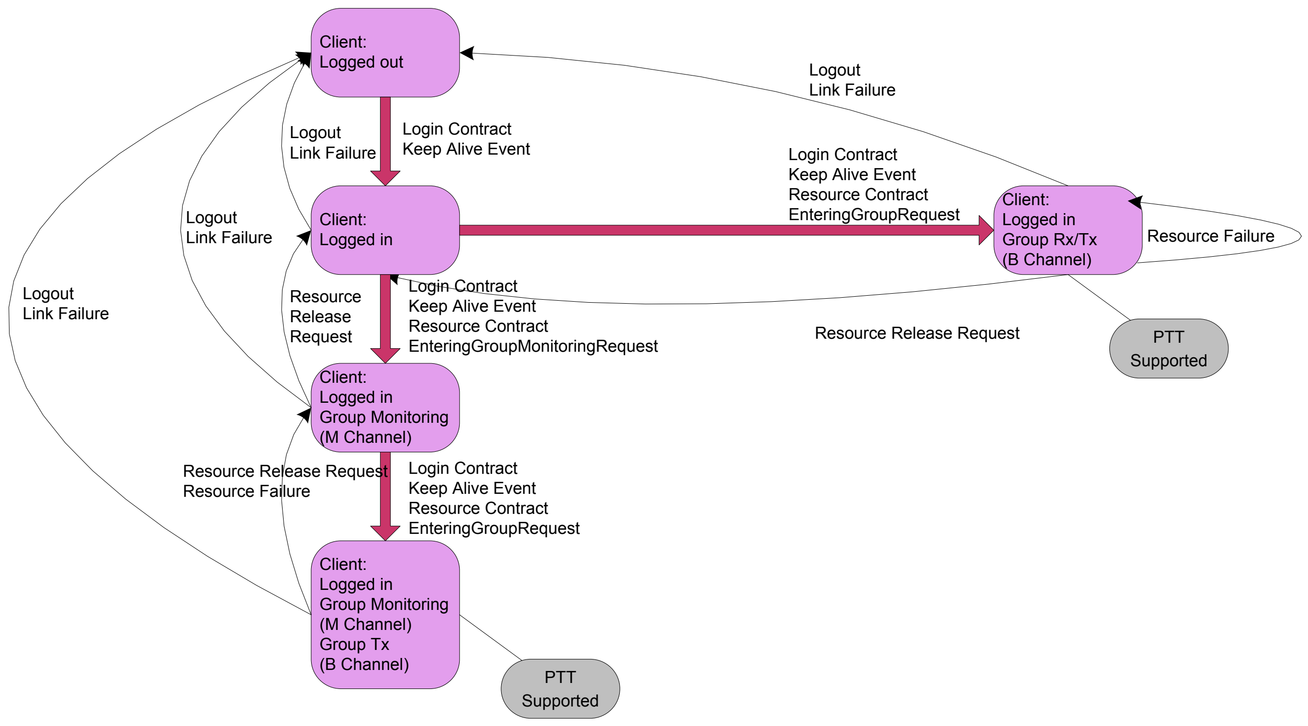


Figure 2 - Resource Request Workflow

4.2.8 SPECIFIC ISSI ASSIGNATION WORKFLOW

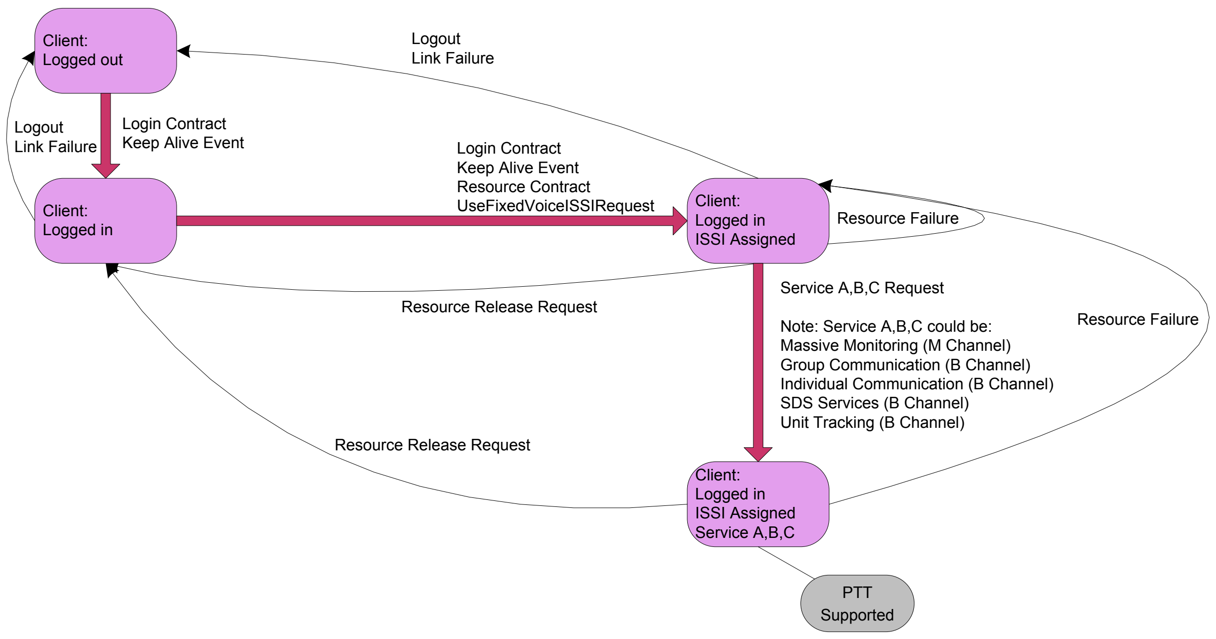


Figure 3 - Resource Request Specific ISSI

4.2.9 API VERSIONNING

The overall architecture of the API is based on WSDL1.1 and SOAP 1.2 W3C recommendations (see www.w3c.org). While the main driver for this architecture is to be able to target as many platforms (3 rd Party Applications) as possible without binding the API to a specific programming language, it was chosen because it can also provide for a better API control and versioning. For example, instead of a providing a C++ API in a binary format as a DLL with .h Header files that can be imported into a third party application, the decision was to use Web Services as they are ubiquitous across many OS's and programming language flavours.

Historically, C, C# or JAVA based APIs posed a challenge when it came to versioning the API methods and data signatures used between the consuming application and the application providing the API. As there was no practical standard in the industry, Web Services have evolved as the ubiquitous method of choice to exposing server based functionality to consuming applications. Although versioning was not part of the original Web Service specifications, a set of guidelines have evolved around Web Services, Those guidelines are captured in subsequent sections.

Once a Web Service contract is published, it shall not change in way that breaks an existing consumer of this contract. It shall remain two versions backwards compatible. Deprecating a Web Service shall undergo a formal inspection to ensure that the API life cycles are respected. When a callback is changed, it shall be considered a breaking change to allow the DF- Server to host multiple versions of the proxy for the same contract but for different versions of the contract.

Backwards-compatibility refers to the ability of the consuming application to use the Web Services from previous releases if a change has been made to an interface. If the consuming application can still work, then the Web Service is said to be backwards-compatible. If the change breaks existing consumer applications, then the change is said to be breaking compatibility, or a non-backwards-compatible change. A backwards-compatible change is typically expressed as an optional parameter in a message, or a new message in the contract in the definition of the portType.

There are no standards based approaches to versioning Web Services. What was found is that various companies employ different methodologies for versioning their Web Services. Various approaches were evaluated and combined to provide optimal versioning strategy for the API.

In general, there are mainly three approaches to Web Services versioning:

Strict: Every time a change is introduced, regardless whether it is backward compatible or not, a new version of the Web Service (including message and types definitions) is created. The Web Service is neither backward compatible, nor forward compatible.

Flexible: If a backward compatible change is introduced, then the current Web Service remains in service, while incompatible changes introduce a new version of the Web Service.

Loose: Any changes to the web service will result in a new version of the Web Service and the new version is both backward and forward compatible.

The *Strict* versioning approach requires new contracts to be rolled out more often and will require additional testing effort as new proxies will be generated with a brand new code base. The *Loose* strategy requires additional effort to ensure forward compatibility, which is not a requirement for the API. In addition, given the limited nature of the use of the API, it is not a desired overhead. With the *Flexible* approach, the Web Service versioning is more practical and less of a management overhead as the changes are well delineated between an incompatible change and a compatible one.

API shall use the flexible versioning strategy as follows:

major.minor.sub-minor having the following format:

nn:nn:nn: where n is a digit from 1 to 9.

Example: 1.0.3 – This version indicates that the Web Service has no new methods, but has additional optional parameters that were defined.

Example: 1.2.3 – This version indicates that the Web Service has additional new methods defined, but no additional optional parameters since release 1.0.3.

Example: 2.0.0– This version indicates that the Web Service has a new contract that superseded version 1.2.3.

major: This version represents the introduction of a new Web Service. Either a new Web Service was created, or non-backwards-compatible-change occurred.

minor: This version represents the introduction of a new method in the contract for a backwards-compatible change.

sub-minor: This version represents the introduction of an optional parameter in the message for a contract for a backwards-compatible change.

Note: This versioning scheme is only valid when updating the *<document> Version n </document>* XML tags. When updating the namespace version, only the {major} number shall be included in the definition of the name space.

The first official release of the API will only include a version number embedded in the <*wsdl:documentation*> XML tag. The rest of the XSD and WSDL file definitions remain intact. However, when a breaking change is introduced, we may need to preserve many of the definitions of the original Web Service while still maintaining the existing Web Service to allow consumers sufficient time to migrate to the new Web Service. Such a requirement can only be solved through the use of namespaces and their canonical representation.

Whenever a breaking change is introduced, a new contract shall be created. The new contract, if needed, can retain the old name of the original contract, but shall live under a new namespace to avoid collision with the existing contract. All messages that are used in the new contract, even though they may be the same as the ones in the original contract shall move into the new namespace.

4.3 API Lower layers protocols

This SOAP API is implemented on top of the TCP and HTTP protocols. Those protocols already have mechanism used to deliver messages and those mechanisms are not exposed to the SOAP layer.

The typical HTTP status messages (on top of the POST request) that are used are the following ones:

100 Continue

200 Ok

400 Bad Requests

4.4 API BINDING

The binding ReliableHttpBinding is the preferred binding to access the Concentrator service and those are the parameters selected:

1. Custom binding based on WS HTTP.
2. SOAP 1.2.
3. Reliale messaging (message reordering).
4. Flow Control On.
5. Security (Encryption) off.

4.5 DIGEST ACCESS AUTHENTICATION

To increase security of the authentication process, the server-client session shall use a principle similar to theDigest Access Authentication at the application level using the Concentrator credential set (It shall not use Windows domain credentials). The DF-Server and DF-Client applications shall also maintain their TCP sessions up to guarantee identity of the senders.

.

4.6 Networking aspects

4.6.1 Firewall

A Firewall should restrict reception of traffic coming from known and expected sources (Concentrator/DF-Server and Control rooms/DF-Client) and steps should be taken to prevent IP address spoofing while designing a DF-Server/DF-Client network.

Restriction on the port range and/or multicast address should be done to allow only expected audio streams sources and destinations to go through the firewalls.

4.6.2 NAT

No NAT will be supported between a DF-Server (Concentrator) and a DF-Client (Control room).

4.6.3 Traffic Tagging

Traffic tagging is not covered part of this DF Stecker interface (API definition) but is considered an important point while deploying a Concentrator/Control room combination. Beeing able to configure those parameters on both side should be considered a nice to have feature and minimally, each applications should support a predefine tagging.

4.6.4 Multicast IGMP

In the case where a Concentrator and/or Control room is implementing audio distribution using multicast group, IGMP V2.0 or 3.0 shall be supported on the network.

4.7 Mandatory and Optional features

4.7.1 Audio Compression

This section is listing which codecs a Concentrator server side implementation may contain to be compliant to the standard.

Codec	Status
G.711	Mandatory
ACELP	Mandatory
Any other codecs	Optional

4.7.2 Audio encryption

Encryption state	Status
Unencrypted	Mandatory
Encrypted	Optional

4.7.3 Audio IP distribution

Audio IP Distribution	Status
Unicast	Mandatory
Multicast	Mandatory

4.7.4 Load And Resource Sharing

This Concentrator switchover functionality is optional.

Based on the assumption that multiple servers could exist in the concentrator handling requests from control rooms, the following scenario shall be supported by the API. The exact details of the implementation on the server are out of the scope of this API document.

If a DF-Client request a resource on a server that does not have enough available resources to accommodate the DF-Client, the server may request other physical servers part of the concentrator about resource availability and return in the resource request event the IP address of the server that could provide the service requested.

It is then the responsibility of the DF-Client to establish a new session with that new server by re-authenticating and by re-requesting the resource using the same messages workflow.

It is up to the server implementation (out of the scope of this document) to determine how many sessions are allowed on a concentrator using the same credentials on the DF-Client side.

4.7.5 Concentrator Failover functionnality

In a regular/simple deployment, it is the responsibility of the DF-Client to establish or re-establish a valid session with the Concentrator and request resources. Resources are automatically released upon a session failure. It is the responsibility of the DF-Client to know (via initial provisioning/configuration) the available IP addresses it can connect to.

The followingConcentrator failover functionality is optional.

Based on the assumption that a concentrator server could exist in a redundant deployment model (Active-Standby or Active-Active), the following scenario shall be supported by the API. The exact details of the implementation on the server are out of the scope of this API document.

If a server fails while in service, the peer server (redundancy partner) shall be able to re-invite the clients that had an active session on the failed server and shall be able to restore the sessions in a state as close as possible to the failed session.

4.8 SDP Description

4.8.1 Example 1 – Operator Microphone

```
v=0\rs=dispatcher-speech\ro=- 0 0 IN IP4 172.20.3.88\rt=0 0\rm=audio 60130 RTP/AVP
8\rc=IN IP4 172.20.3.88\ra=sendonly\r
```

4.8.2 Example 2 – Incoming B Channel

```
v=0\rs=dispatcher-channel\ro=- 0 0 IN IP4 172.20.3.88\rt=0 0\rm=audio 60132
RTP/AVP 8\rc=IN IP4 238.0.0.1\ra=reconly\r
```

4.8.3 Example 3 – Incoming M Channel

```
v=0\rs=audio-channel\ro=- 0 0 IN IP4 172.20.3.88\rt=0 0\rm=audio 60134 RTP/AVP
8\rc=IN IP4 238.0.0.2\ra=reconly\r
```

4.9 Design constraints

1. Only one B channel can be associated to a DF-Client within the active session it has established with the server.
2. Credentials can only be used once to establish a valid session (login). In other words, user name cannot be used to login two sessions simultaneously.
3. Other restrictions would be product specific.

5 Common Types

5.1 Basic Types

In the context of this API, the following types are defined like:

Int: 32 bits value.
Short: 16 bits value.

5.2 ServiceTypes

ServiceTypes (Enum):

LoginEvents	;Login contract (including Keep Alive Event)
TETRAResourceEvents	;Resource contract
TETRAMassiveMonitoringGroupEvents	;Group Monitoring contract
TetraGroupEvents	;Group contract
TetraIndividualCallEvents	;Individual Call contract
TetraSdsEvents	;Short Data Services contract
TetraUnitTrackingEvents	;Unit Tracking contract

5.3 EntityType

EntityType (Enum):

TetraGroup	;TETRA Group
IndividualCall	;TETRA Individual Call
EntityInvalid	;Error cases

5.4 EncryptionState

EncryptionState (Enum):

EncryptionOff	;Encryption feature Off
EncryptionOn	;Encryption feature On
EncryptionDisable	;Encryption feature not available
EncryptionError	;Encryption error is detected

5.5 MonitorMode

MonitorMode (Enum)

MonitorNone	;Group not monitored
Select	;Group monitored actively (active use)
Unselect	;Group monitored (use)
EventMonitored	;Group in event monitored state (no audio)

5.6 MassiveMonitorMode

MassiveMonitorMode (Enum):

MonitorNone	;Group not monitored
MassiveMonitoring	;Group monitored using a M channel

5.7 CallMode

CallMode (Enum):

ModeNormal	;Normal call
ModeEmergency	;Emergency call
ModeDisable	;Call mode not available

5.8 DuplexMode

DuplexMode (Enum):

HalfDuplex	;Half Duplex Call
FullDuplex	;Full Duplex Call

5.9 PriorityType

PriorityType (Enum):

PriorityNormal	;Normal priority
PriorityPreemptive	;Preemptive priority
PriorityEmergency	;Emergency priority

5.10 CallbackType

CallbackType (Enum):

CallbackTypeNormal	;Normal callback
CallbackTypeUrgent	;Urgent callback
CallbackTypeEmergency	;Emergency callback
CallbackTypeDispatcher	;DF-Client callback

5.11 PreemptionState

PreemptionState (Enum):

PreemptionModeOff	;Preemption Off

PreemptionModeOn	;Preemption On
PreemptionModeDisabled	;Preemption Not available

5.12 TetraSdsAction

TetraSdsAction (Enum):

ActionNew	;New SDS
ActionReply	;Reply to SDS
ActionForward	;Forward SDS

5.13 ItemType

ItemType (Enum):

ItemTypeStatus	;Status SDS
ItemTypeText	;Text SDS
ItemTypeLip	;Lip SDS

5.14 LastAction

LastAction (Enum):

LastActionNew	;New SDS
LastActionReply	;Reply to SDS
LastActionForward	;Forward SDS
LastActionDelete	;Delete SDS

5.15 TetraReleaseReason

Type TetraReleaseReason:

Value	Description
None	No reason is present.
NoRights	No rights.
NotDefined	Reason not defined.
UserRequested	User requested disconnection.
CalledPartyBusy	Called party busy.
CalledPartyNotReachable	Called party not reachable.
EncryptionNotSupported	Encryption not supported.
Congestion	Congestion in infrastructure.
NotAllowed	Traffic case not allowed.
Incompatible	Incompatible traffic case.
NotAvailable	Requested service not available.
Preemptive	Preemptive use of resources.
InvalidCallIdentifier	Invalid call identifier.
Rejected	Call rejected by the called party.
NoCallControl	No idle Call Control entity.
TimerExpired	Expiry of timer.
SystemRequested	System requested disconnection.
NotReady	Acknowledged service not completed.
Answered	Answered by another dispatcher.
Unknown	Unknown reason.
EncryptionError	Encryption error.
NotLicensed	Not licensed.
SessionAlreadyExists	Session already exists.
InvalidUser	Invalid user.
TooManyClients	Too many clients.
InvalidTimeoutValue	Invalid timeout value.
ClearLoginForbidden	Clear login forbidden.
CapacityExceeded	Capacity exceeded.
SystemError	System error.
OperationTimeout	Operation timeout.
NoSession	No session.

ServerShutdown	Server shutdown.
MessageQueueFull	Message queue full.
NoSocketConnection	No socket connection.
ClientApplicationError	Client application error.
TargetDoesNotExist	Target does not exist.
TooManyWorkstations	Too many workstations.
TooManyGroups	Too many groups.
TargetDeleted	Target deleted.
G4WifInGroup	External interface is connected to the group.
G4WifIsOffLine	External interface is off-line.
NoDisconnectionRights	No rights to disconnect an emergency call.
CannotReleaseEmgCall	Cannot release an emergency call.
CannotReleaseAmbCall	Cannot release an ambience listening call.
RemoveEmergencyState	Remove emergency call state.
IndividualDeliveryFailed	Delivery of an individual addressed SDS message failed.
GroupDeliveryFailed	Delivery of a group addressed SDS message failed.
SystemDisturbance	System disturbance.
SdsTooLong	SDS message is too long.
IllegalService	Illegal service.
TargetUnknown	Target unknown.
FeatureNotInUse	Feature not in use.
OperationNotPerformed	Operation not performed.
G4WifDoesNotExist	External interface does not exist.
G4WifAlreadyInGroup	External interface is already connected to the group.
G4WifAlreadyUsedbyWS	External interface already used by a workstation user.
G4WifNotInGroup	External interface is not connected to the group.
NoMembersInBkgrndGroup	Members cannot be added to a background group.
OrgBlockDoesNotExist	Organization block does not exist.
RadioAlreadyInGroup	Radio subscriber is already a member of the group.
RadioNotInGroup	Radio subscriber is not a member of the group.
SubscriberDoesNotExist	Subscriber does not exist.
SubscriberHasNoGroupCBR	Subscriber has no callback request to the group.
TooManyRadioInGroup	Too many radio subscribers in the group.
WsUserDoesNotExist	Workstation user does not exist.
GroupAlreadyInOverlay	Group is already a member of the group overlay.
GroupNotInGroupOverlay	Group is not a member of the group overlay.
TooManyOverlaysForGroup	Too many group overlays for the group.
TooManyGroupsInOverlay	Too many groups in the group overlay.
PasswordEncMismatch	Password encryption mismatch.

ClientAppHasSession	Client application has a session.
OrgHasNoPreemptiveRights	Organization has no pre-emptive rights.
InvalidGroupCombineRights	Invalid group combine rights.
GroupAlreadyInCombination	Group is already part of the combination.
GroupNotInCombination	Group is not part of the combination.
GroupInAnotherCombination	Group is part of another combination.
TooManyInCombinedGroup	Too many groups in the combination.
BaseGroupDoesNotExist	Base group does not exist.
BaseGroupInAnotherCombination	Base group is part of another combination.
EmergencyCallInProgress	Emergency call is in progress.
EncryptionModeMismatch	Encryption mode mismatch.
OperationAlreadyPending	Operation already pending.
GroupIsPartOfCombination	Group is part of a combination.
CannotRemoveBaseGroup	Cannot remove the base group of a combination.
IllegalOperation	Illegal operation.
CannotCombineCoverGroup	Cover group cannot be combined.
SuccessInDxtFailedInCdd	Operation succeeded in exchanges but failed in the configuration and data distribution server.
NetworkSpecificOption	Network specific option.
ResourceExhaustion	No resources are available.
RadioDoesNotExist	Radio subscriber does not exist.
AirInterfaceEncryptionMismatch	Air interface encryption method mismatch.

5.16 TetraReleaseReasonData

Type TetraReleaseReasonData:

Component	Type	Occurs	Description
TetraReleaseReason	TetraReleaseReason	1..1	ReleaseReason.
text	string	1..1	Release reason text.

5.17 ArrayOfString

Type ArrayOfstring

Component	Type	Occurs	Description
string	string	0..*	

5.18 ARRAYOFCALLBACKDATA

Type ArrayOfCallbackData :

Component	Type	Occurs	Description
CallbackData	CallbackData	1..*	

5.19 CALLBACKDATA

Type CallbackData:

Component	Type	Occurs	Description
CallbackType	CallbackType	1..1	Call back type.
CallerId	TetraSubscriberData	1..1	Caller subscriber data.
IsCleared	boolean	1..1	Value indicating whether a call is cleared (true or false).
TimeStamp	dateTime	1..1	Call timestamp.

5.20 ARRAYOFCALLBACKSERVICEDATA

Type ArrayOfCallbackServiceData:

Component	Type	Occurs	Description
CallbackServiceData	CallbackServiceData	1..*	

5.21 CallbackServiceData

Type CallbackServiceData

Component	Type	Occurs	Description
Binding	BindingType (Enum): BasicHttpBinding ReliableHttpBinding	1..1	BindingType. Only ReliableHttpBinding is certified.
Type	ServiceTypes	1..1	ServiceType.
Url	string	1..1	URL. Those are for event to be received. The format is the following: http://[IPAddress:8130]/RCSAPI/[ServiceName]

5.22 ArrayOfTetraSubscriberData

Type ArrayOfTetraSubscriberData:

Component	Type	Occurs	Description
TetraSubscriberData	TetraSubscriberData	1..*	

5.23 TetraSubscriberData

Type TetraSubscriberData:

Component	Type	Occurs	Description
AddressType	TetraAddressType (Enum): TetraAddressTypeNone TetraAddressTypeSsi TetraAddressTypeTsi	1..1	Address type (SSI or TSI). This field is required when sending a request.
CountryCode	short	1..1	Country code (for TSI addresses). This field is required when sending a request if TSI is supported.
IsVisitorPassingAccepted	boolean	1..1	Value indicating whether a visitor organization block has accepted management of the subscriber. This field is not required when sending a request.
LocationMnemonic	string	1..1	Location area mnemonic. This field is not required when sending a request.

Component	Type	Occurs	Description
Mnemonic	string	1..1	Subscriber mnemonic. This field is not required when sending a request.
NetworkCode	short	1..1	Network code (for TSI addresses). This field is required when sending a request if TSI is supported.
OperativeTacticalAddress	string	1..1	Subscriber Operative Tactical Address (OPTA). This field is not required when sending a request.
OrgBlockMnemonic	string	1..1	Organization block mnemonic. This field is not required when sending a request.
Ssi	int	1..1	Short subscriber identifier (SSI). This field is required when sending a request.
SubscriberAddressList	ArrayOfSubscriberAddress	1..1	Array of all the subscriber addresses.

Component	Type	Occurs	Description
SubscriberType	TetraSubscriberType (Enum): SubscriberTypeUnknown SubscriberTypeRU SubscriberTypeWS SubscriberTypeG4Wif SubscriberTypeGroup	1..1	Subscriber type. This field is not required when sending a request.

5.24 TetraSpeechEventData

Type TetraSpeechEventData:

Component	Type	Occurs	Description
CallId	int	1..1	Call identifier.
CallerId	TetraSubscriberData	1..1	Caller subscriber data.
CallerSource	CallerSource (Enum): CallerSourceThisDp CallerSourceAnotherDp CallerSourceOther	1..1	Caller source. Dp is a reference to Dispatch position which could be seen as a DF-Client.
EntityType	EntityType	1..1	Entity type.
IsEncrypted	boolean	1..1	Value indicating whether audio is encrypted (true or false).
UniqueId	int	1..1	Unique identifier. GSSI of a group in the group context and ISSI in the individual call context.

5.25 TetraIncomingAudioData

Type TetraIncomingAudioData

Component	Type	Occurs	Description
CallId	int	1..1	Incoming call identifier.
CallerId	TetraSubscriberData	1..1	Caller subscriber data.
IsAudioOn	boolean	1..1	Value indicating whether a call has been established.
IsEmergency	boolean	1..1	Indicates whether the caller is using emergency or normal priority.
TimeStamp	dateTime	1..1	Datestamp of the call
IsEncrypted	boolean	1..1	Encryption state
IsAudioFailure	boolean	1..1	Audio failure

5.26 LinkStateData

Type LinkStateData

Component	Type	Occurs	Description
EntityType	EntityType	1..1	Entity Type.
LinkState	LinkState (Enum): LinkStateDown LinkStateUp	1..1	Link State.
UniqueId	int	1..1	UniqueId. GSSI of a group in the group context and ISSI in the individual call context.

5.27 TransmitEventData

Type TransmitEventData:

Component	Type	Occurs	Description
CallId	int	1..1	CallId.
EntityType	EntityType	1..1	Entity type.
TimeStamp	dateTime	1..1	Timestamp.
TransmitState	TransmitState (Enum): TransmitOff TransmitOnAtThisDp TransmitQueued TransmitOnPartial TransmitError TransmitTimedError TransmitDisabled TransmitOnNoCoverage TransmitFailed	1..1	TransmitState. Tx Off Tx by this DF-Client Tx Queued in the DXT Tx not on full group area Tx Error (DF-Server or DXT) Tx Momentary Error Tx not supported Transmit on with no area coverage Audio Failure
UniqueId	int	1..1	UniqueId. GSSI of a group in the group context and ISSI in the individual call context.
IsTransmitInterrupted	boolean	1..1	Value indicates whether the tranmsit was interrupted.

5.28 LeadingDigitData

Type LeadingDigitData:

Component	Type	Occurs	Description

Component	Type	Occurs	Description
Digit	string	1..1	Digits.
NumberingPlan	NumberingPlan (Enum): NumberingPlanTetra NumberingPlanFssn NumberingPlanExternal NumberingPlanMsisdn	1..1	NumberingPlan.
IsDigitForwarded	Boolean	1..1	Value indicating whether the leading digit is included when signaling the address to the TETRA network.

5.29 ArrayOfLeadingDigitData

Type ArrayOfLeadingDigitData

Component	Type	Occurs	Description
LeadingDigitData	LeadingDigitData	1..*	

5.30 SpecialNumbersData

Type SpecialNumbersData:

Component	Type	Occurs	Description
Number	string	1..1	Number.

Component	Type	Occurs	Description
NumberingPlan	NumberingPlan (Enum): NumberingPlanTetra NumberingPlanFssn NumberingPlanExternal NumberingPlanMsisdn	1..1	NumberingPlan.

5.31 ArrayOfSpecialNumbersData

Type ArrayOfSpecialNumbersData:

Component	Type	Occurs	Description
SpecialNumbersData	SpecialNumbersData	1..*	

5.32 ArrayOfSubscriberAddress

Type ArrayOfSubscriberAddress:

Component	Type	Occurs	Description
SubscriberAddress	SubscriberAddress	1..*	

5.33 SubscriberAddress

Type SubscriberAddress:

Component	Type	Occurs	Description
Number	String	1..1	The subscriber address in analysed format
NumberingPlan	NumberingPlan (Enum): NumberingPlanTetra NumberingPlanFssn NumberingPlanExternal NumberingPlanMsisdn	1..1	The numbering plan.

Technical Description of Digitalfunkstecker V0.2

Bundesverband Professioneller Mobilfunk (PMeV)

6 LOGIN & LOGINEVENTS SERVICE WEB SERVICE

Unlike the other events which are received on port 8130. The login events services are received on port 8131.

6.1 USE CASES

6.1.1 LOGIN

The login use-case illustrates which steps are required to establish a valid session between a DF-Client and a DF-Server. It is a set of synchronous request. This use case has no prerequisites has long as there is a valid IP connectivity Link to the DF Server from the DF-Client.

This use case is necessary in the following scenarios:

- Initial setup of a connection
- Restoration of a broken session after a failure
- Redirection to another DF-Server in case the initial DF –Server is not able to provide the requested resources.

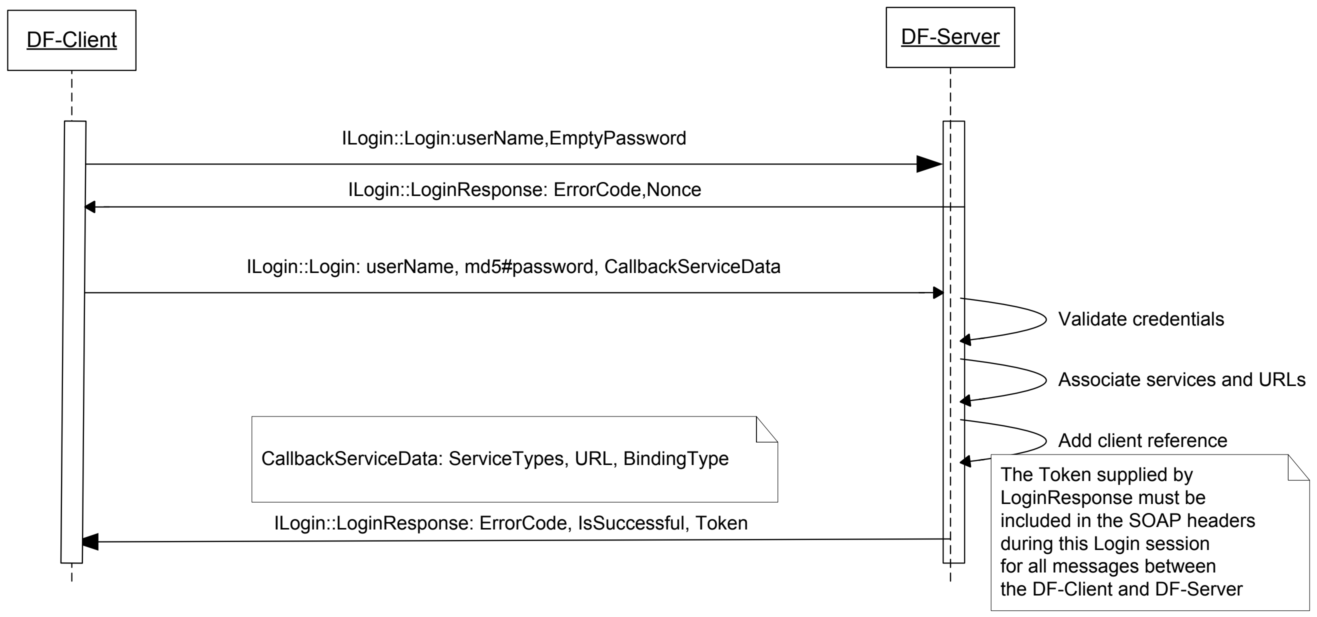

DF-Client
DF-Server
ILogin::Login:userName,EmptyPassword
ILogin::LoginResponse: ErrorCode,Nonce
ILogin::Login: userName, md5#password, CallbackServiceData
Validate credentials
Associate services and URLs
Add client reference
CallbackServiceData: ServiceTypes, URL, BindingType
The Token supplied by LoginResponse must be included in the SOAP headers during this Login session for all messages between the DF-Client and DF-Server
ILogin::LoginResponse: ErrorCode, IsSuccessful, Token

Figure 4 - Login

6.1.2 LOGOUT

Logout is the well behaving method to terminate a session. It enables the DF-Server to completely remove any references to the DF-Client. Once a logout is completed, the only way to re-establish a session is to restart the login process by following the login use cases presented above.

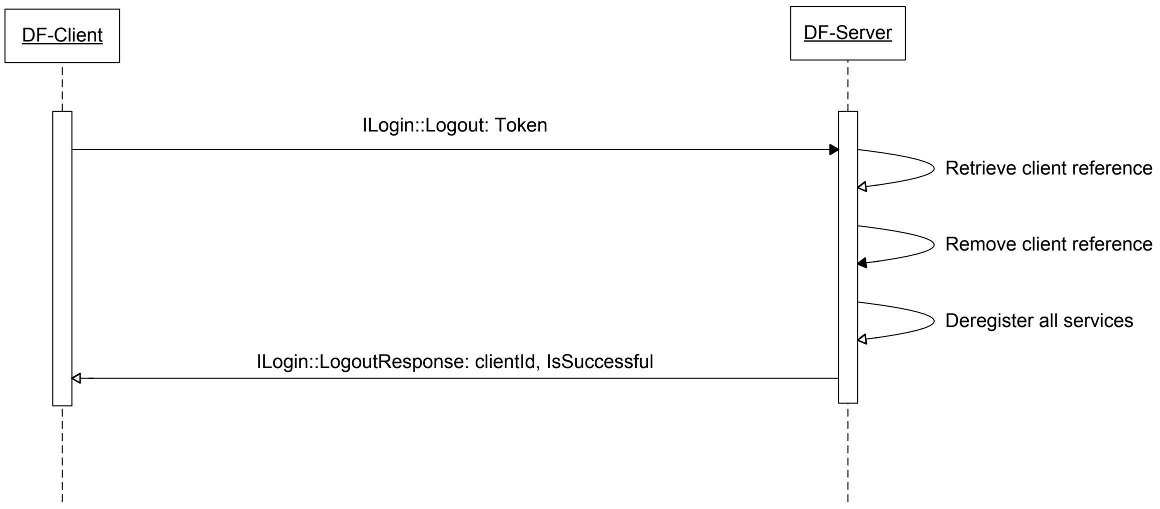


Figure 5 - Logout

6.2 LoginService Web Service

6.2.1 Method: Login

Login allows a DF-Client to log into the DF-Server. The username, password, and contract version are all validated during login. The LoginResponse indicates success or failure of the login validation. On login success, a token is returned to the DF-Client as part of the LoginResponse. This token uniquely identifies each login session; it must be saved by the DF-Client and included in the WCF header of all further requests. The .NET Framework provides a simple method for the DF-Client to include the token.

On login failure, an enum failure code is returned in the LoginResponse (and the token is invalid). The assumption is the DF-Client will localize the text corresponding to this failure code if text is displayed to the user.

A first login attempt must be made with an empty password to receive the Nonce used to hash the password that will enable achieving a succesfull login.

Input (Literal)

The input of this method is the document element Login having the structure defined by the following table.

Element	Type	Occurs	Description
userName	string	1..1	A valid user name; provisioned in the Concentrator

Element	Type	Occurs	Description
password	string	1..1	A valid password; provisioned in the ConcentratorThe password must be be sent as an MD5 hash string rather than in plain text format. Nonce need to be used in the following way: MD5(Username:nonce:MD5(password))
Language	string	1..1	Based on IETF language tags defined in RFC 3066. [Langage Name-Region Name]
CallbackServiceData	ArrayOfCallbackServiceData	1..1	CallbackServiceData services The list of callback URLs for all service events that the DF-Client wishes to receive.

Output (Literal)

LoginResponse: ErrorCode, IsSuccessfull, and Token

Element	Type	Occurs	Description
LoginResponse	LoginResponseType	1..1	

Type LoginResponseType:

Component	Type	Occurs	Description
IsSuccessfull	boolean	1..1	Positive authentication

Component	Type	Occurs	Description
ErrorCode	LoginErrorCode (Enum): NoError UnknownUser InvalidUser InvalidPassword UnknownError MissingLoginEventsServiceURL UserAlreadyLoggedIn MaximumClientsAlreadyLoggedIn NonceRequest	1..1	Error code, if there was a login error. Enum labels are self descriptive.
Token	string	1..1	Token supplied during login.
Text	String	1..1	Localized text associated with the login error code.
Nonce	String	1..1	Cryptographic Nonce use to hash the password.

6.2.2 METHOD: LOGOUT

Logout allows a DF-Client to log out of the DF-Service. Logout also deregisters the DF-Client from receiving all service events specified during login.

Input (Literal)

The input of this method is the document element Logout having the structure defined by the following table.

Element	**Type**	**Occurs**	**Description**
token	string	1..1	The token, supplied during login, used to uniquely identify the login session.

Output (Literal)

LogoutResponse: Clientid and IsSuccessfull

Element	**Type**	**Occurs**	**Description**
LogoutResult	LogoutResponse	1..1	

Type LogoutResponse:

Component	**Type**	**Occurs**	**Description**
IsSuccessfull	boolean	1..1	Value indicating whether logout was successfull.

6.3 LoginEventsService Web Service

6.3.1 Method: KeepAliveEvent

KeepAliveEvent is sent to clients once per second. If sent to a non-operational client a failure is detected and treated as a logout of the client. The client could use failure to receive this event to detect a non-operational service.

It is recommended that if a client receives a keep-alive from a server that it is not currently connected to, that it throws an exception on the client. This will allow the server to detect a failure.

Input (Literal)

The input of this method is the document element KeepAliveEvent having the structure defined by the following table.

Element	Type	Occurs	Description
ipAddress	string	1..1	ipAddress of the sender.

6.3.2 Method: ForceLogoutEvent

Required in cases where the DF-Server need to logout a DF-Client. Possible cases are in cases DF-Clients are violating the API contracts.

Input (Literal)

The input of this method is the document element ForceLogoutEvent that is always empty.

7 TETRA Resource & Resource Events Service Web Service

7.1 Use Cases

Three use-cases are demonstrated in this section. All the other use cases (other requests) are very similar to those ones and would be repetitions.

In the examples, the messages related to the HTTP layers are voluntarily omitted to keep the message flow simples. In the UseFixVoiceISSIRequest example, a Wireshark trace snapshot is included to give an example of all those HTTP messages.

7.1.1 Entering Group Monitoring

This use-case demonstrates how to gain access to a group for massive audio monitoring. The prerequisite is to have a valid session (login completed) between the DF-Server and DF-Client.

Once the M channels are available (refer to the message flow diagram), a DF-Client can proceed with the Tetra Massive Monitoring Group & Events Services. Those services are available until the DF-Client proceeds with the ResourceReleaseRequest of the TETRAMassiveMonitoringGroupEvents.

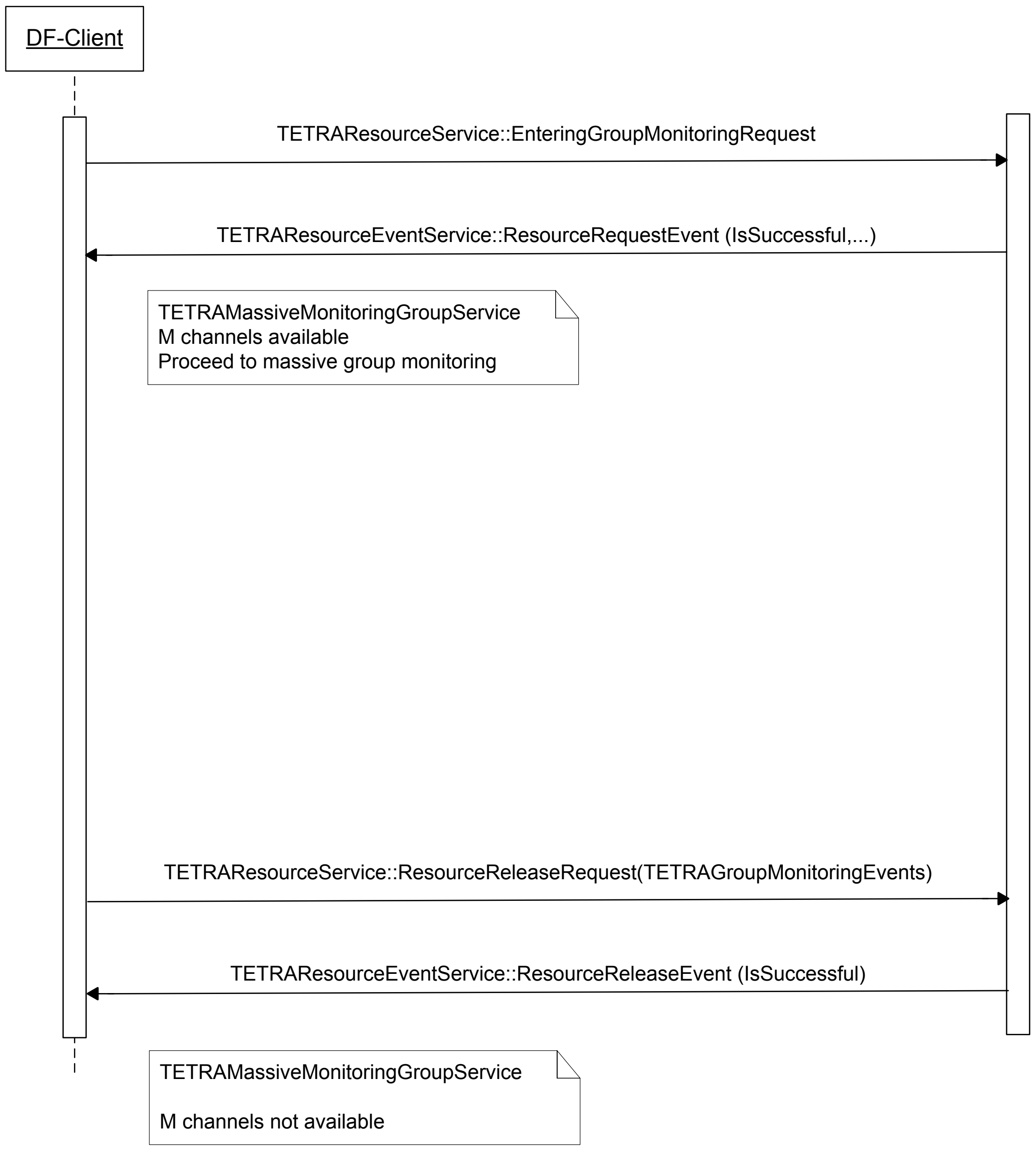

DF-Client
TETRAResourceService::EnteringGroupMonitoringRequest
TETRAResourceEventService::ResourceRequestEvent (IsSuccessful,...)
TETRAMassiveMonitoringGroupService
M channels available
Proceed to massive group monitoring
TETRAResourceService::ResourceReleaseRequest(TETRAGroupMonitoringEvents)
TETRAResourceEventService::ResourceReleaseEvent (IsSuccessful)
TETRAMassiveMonitoringGroupService
M channels not available

Figure 6 - Resource Request – Group Monitoring

7.1.2 ENTERING GROUP REQUEST

This use-case demonstrates how to enter into a group communication. In other words, how to get a B channel allocated to a DF-Client. The only prerequisite is to have a valid session (login completed) between the DF-Server and DF-Client. Also important to note is that only one B channel can be allocated within a session.

Once the B channel is assigned for group communication, all the requests and events from the Group Service Request & Events contracts are available (requests) and should be expected (events).

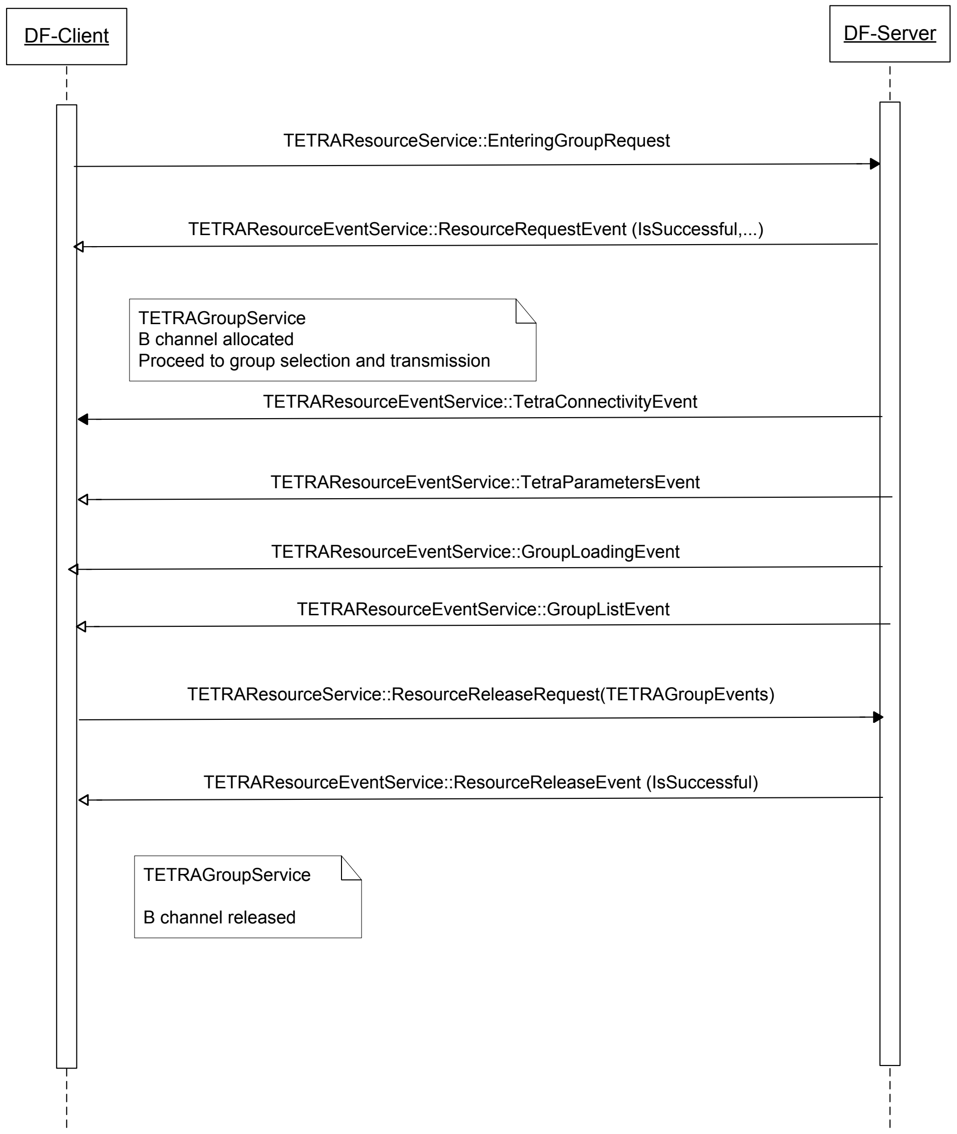

DF-Client
DF-Server
TETRAResourceService::EnteringGroupRequest
TETRAResourceEventService::ResourceRequestEvent (IsSuccessful,...)
TETRAGroupService
B channel allocated
Proceed to group selection and transmission
TETRAResourceEventService::TetraConnectivityEvent
TETRAResourceEventService::TetraParametersEvent
TETRAResourceEventService::GroupLoadingEvent
TETRAResourceEventService::GroupListEvent
TETRAResourceService::ResourceReleaseRequest(TETRAGroupEvents)
TETRAResourceEventService::ResourceReleaseEvent (IsSuccessful)
TETRAGroupService
B channel released

Figure 7 – Resource Request – Entering Group

7.1.3 USE FIX VOICE ISSI REQUEST

The following use-case is demonstrating how to get a fixed voice ISSI allocated to a DF-Client. The same prerequisite are applicable which are to have a valid session and not to have any B channel already allocated.

Once the request is succesfull, all the contracts contained in this document are fully supported and any event could be received.

The Wireshark trace is included to provide an example of all the HTTP messages that are exchanged during that resource request process. In this example, there is also the login process included and since it is a FixVoiceISSI request, all the contracts information is exchanged between the DF-Client and the DF-Server so that all the requests and events information could be exchanged between the two parties.

The UseFixDataISSIRequest Use-case is very similar to this one with the exception that the ISSI allocated is a data ISSI which does not have any voice capabilities.

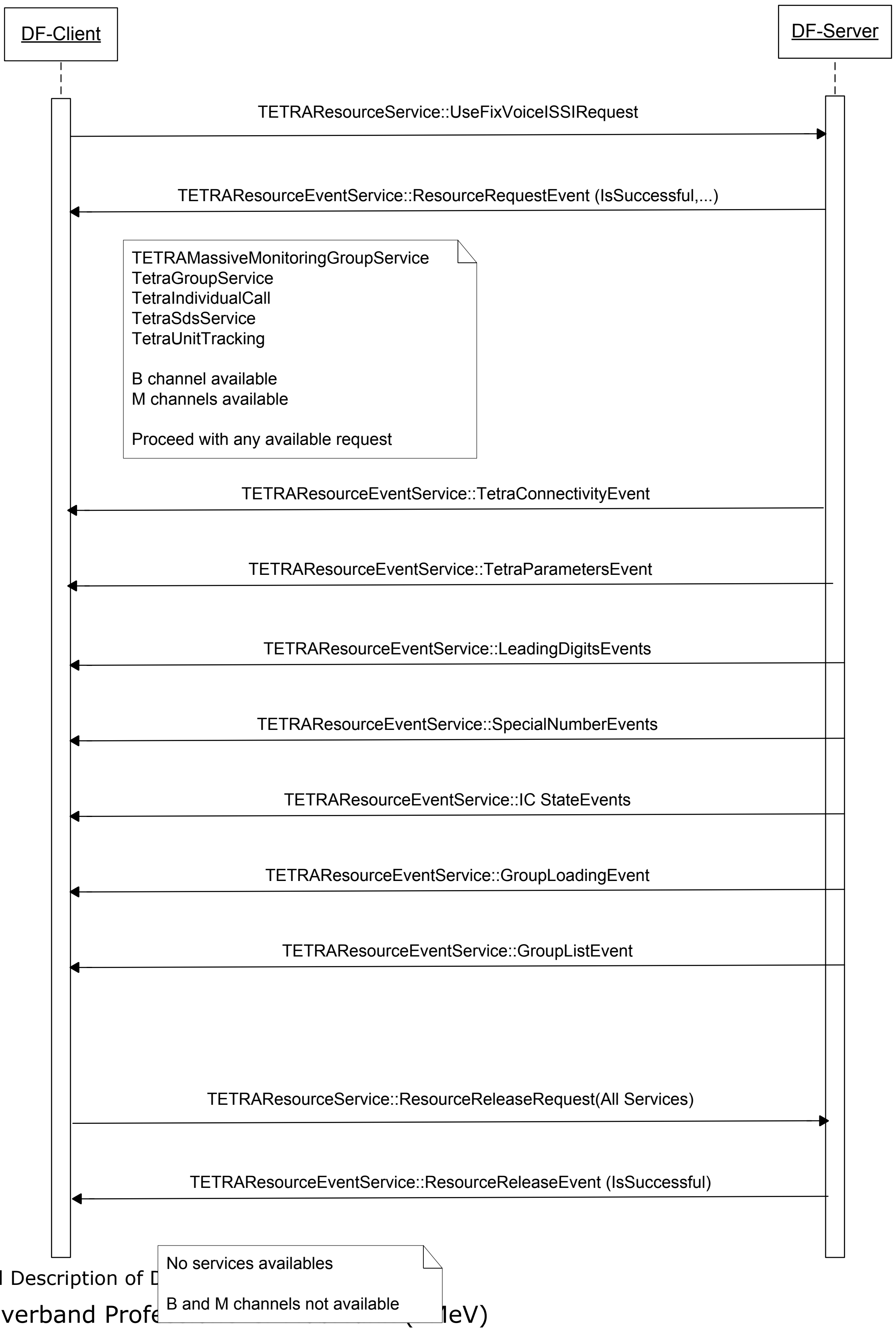
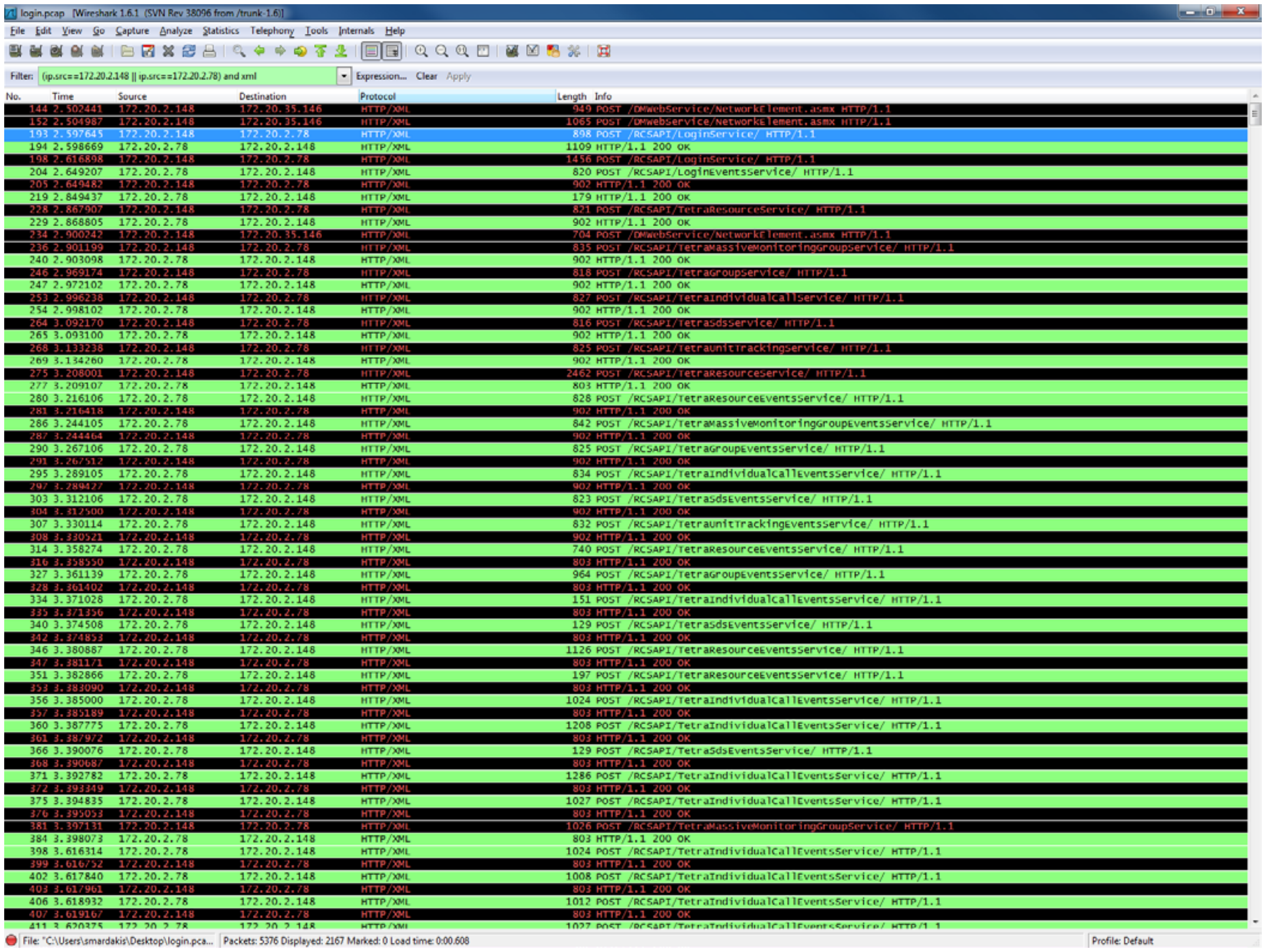

DF-Client
DF-Server
TETRAResourceService::UseFixVoiceISSIRequest
TETRAResourceEventService::ResourceRequestEvent (IsSuccessful,...)
TETRAMassiveMonitoringGroupService
TetraGroupService
TetraIndividualCall
TetraSdsService
TetraUnitTracking
B channel available
M channels available
Proceed with any available request
TETRAResourceEventService::TetraConnectivityEvent
TETRAResourceEventService::TetraParametersEvent
TETRAResourceEventService::LeadingDigitsEvents
TETRAResourceEventService::SpecialNumberEvents
TETRAResourceEventService::IC StateEvents
TETRAResourceEventService::GroupLoadingEvent
TETRAResourceEventService::GroupListEvent
TETRAResourceService::ResourceReleaseRequest(All Services)
TETRAResourceEventService::ResourceReleaseEvent (IsSuccessful)
No services availables
B and M channels not available

Figure 8 - Resource Request - UseFixVoiceISSIRequest


login.pcap [Wireshark 1.6.1 (SVN Rev 38096 from /trunk-1.6)]

Filter: (ip.src==172.20.2.148 || ip.src==172.20.2.78) and xml

```
No.  Time      Source         Destination     Protocol   Length Info
 144 2.502441  172.20.2.148   172.20.35.146   HTTP/XML    949 POST /DMWebService/NetworkElement.asmx HTTP/1.1
 152 2.504987  172.20.2.148   172.20.35.146   HTTP/XML   1065 POST /DMWebService/NetworkElement.asmx HTTP/1.1
 193 2.597645  172.20.2.148   172.20.2.78     HTTP/XML    898 POST /RCSAPI/LoginService/ HTTP/1.1
 194 2.598669  172.20.2.78    172.20.2.148    HTTP/XML   1109 HTTP/1.1 200 OK
 198 2.616898  172.20.2.148   172.20.2.78     HTTP/XML   1456 POST /RCSAPI/LoginService/ HTTP/1.1
 204 2.649207  172.20.2.78    172.20.2.148    HTTP/XML    820 POST /RCSAPI/LoginEventsService/ HTTP/1.1
 205 2.649482  172.20.2.148   172.20.2.78     HTTP/XML    902 HTTP/1.1 200 OK
 219 2.849437  172.20.2.78    172.20.2.148    HTTP/XML    179 HTTP/1.1 200 OK
 228 2.867907  172.20.2.148   172.20.2.78     HTTP/XML    821 POST /RCSAPI/TetraResourceService/ HTTP/1.1
 229 2.868805  172.20.2.78    172.20.2.148    HTTP/XML    902 HTTP/1.1 200 OK
 234 2.900242  172.20.2.148   172.20.35.146   HTTP/XML    704 POST /DMWebService/NetworkElement.asmx HTTP/1.1
 236 2.901199  172.20.2.148   172.20.2.78     HTTP/XML    835 POST /RCSAPI/TetraMassiveMonitoringGroupService/ HTTP/1.1
 240 2.903098  172.20.2.78    172.20.2.148    HTTP/XML    902 HTTP/1.1 200 OK
 246 2.969174  172.20.2.148   172.20.2.78     HTTP/XML    818 POST /RCSAPI/TetraGroupService/ HTTP/1.1
 247 2.972102  172.20.2.78    172.20.2.148    HTTP/XML    902 HTTP/1.1 200 OK
 253 2.996238  172.20.2.148   172.20.2.78     HTTP/XML    827 POST /RCSAPI/TetraIndividualCallService/ HTTP/1.1
 254 2.998102  172.20.2.78    172.20.2.148    HTTP/XML    902 HTTP/1.1 200 OK
 264 3.092170  172.20.2.148   172.20.2.78     HTTP/XML    816 POST /RCSAPI/TetraSdsService/ HTTP/1.1
 265 3.093100  172.20.2.78    172.20.2.148    HTTP/XML    902 HTTP/1.1 200 OK
 268 3.133238  172.20.2.148   172.20.2.78     HTTP/XML    825 POST /RCSAPI/TetraUnitTrackingService/ HTTP/1.1
 269 3.134260  172.20.2.78    172.20.2.148    HTTP/XML    902 HTTP/1.1 200 OK
 275 3.208001  172.20.2.148   172.20.2.78     HTTP/XML   2462 POST /RCSAPI/TetraResourceService/ HTTP/1.1
 277 3.209107  172.20.2.78    172.20.2.148    HTTP/XML    803 HTTP/1.1 200 OK
 280 3.216106  172.20.2.78    172.20.2.148    HTTP/XML    828 POST /RCSAPI/TetraResourceEventsService/ HTTP/1.1
 281 3.216418  172.20.2.148   172.20.2.78     HTTP/XML    902 HTTP/1.1 200 OK
 286 3.244105  172.20.2.78    172.20.2.148    HTTP/XML    842 POST /RCSAPI/TetraMassiveMonitoringGroupEventsService/ HTTP/1.1
 287 3.244464  172.20.2.148   172.20.2.78     HTTP/XML    902 HTTP/1.1 200 OK
 290 3.267106  172.20.2.78    172.20.2.148    HTTP/XML    825 POST /RCSAPI/TetraGroupEventsService/ HTTP/1.1
 291 3.267512  172.20.2.148   172.20.2.78     HTTP/XML    902 HTTP/1.1 200 OK
 295 3.289105  172.20.2.78    172.20.2.148    HTTP/XML    834 POST /RCSAPI/TetraIndividualCallEventsService/ HTTP/1.1
 297 3.289427  172.20.2.148   172.20.2.78     HTTP/XML    902 HTTP/1.1 200 OK
 303 3.312106  172.20.2.78    172.20.2.148    HTTP/XML    823 POST /RCSAPI/TetraSdsEventsService/ HTTP/1.1
 304 3.312500  172.20.2.148   172.20.2.78     HTTP/XML    902 HTTP/1.1 200 OK
 307 3.330114  172.20.2.78    172.20.2.148    HTTP/XML    832 POST /RCSAPI/TetraUnitTrackingEventsService/ HTTP/1.1
 308 3.330521  172.20.2.148   172.20.2.78     HTTP/XML    902 HTTP/1.1 200 OK
 314 3.358274  172.20.2.78    172.20.2.148    HTTP/XML    740 POST /RCSAPI/TetraResourceEventsService/ HTTP/1.1
 316 3.358550  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 327 3.361139  172.20.2.78    172.20.2.148    HTTP/XML    964 POST /RCSAPI/TetraGroupEventsService/ HTTP/1.1
 328 3.361402  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 334 3.371028  172.20.2.78    172.20.2.148    HTTP/XML    151 POST /RCSAPI/TetraIndividualCallEventsService/ HTTP/1.1
 335 3.371356  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 340 3.374508  172.20.2.78    172.20.2.148    HTTP/XML    129 POST /RCSAPI/TetraSdsEventsService/ HTTP/1.1
 342 3.374853  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 346 3.380887  172.20.2.78    172.20.2.148    HTTP/XML   1126 POST /RCSAPI/TetraResourceEventsService/ HTTP/1.1
 347 3.381171  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 351 3.382866  172.20.2.78    172.20.2.148    HTTP/XML    197 POST /RCSAPI/TetraResourceEventsService/ HTTP/1.1
 353 3.383090  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 356 3.385000  172.20.2.78    172.20.2.148    HTTP/XML   1024 POST /RCSAPI/TetraIndividualCallEventsService/ HTTP/1.1
 357 3.385189  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 360 3.387775  172.20.2.78    172.20.2.148    HTTP/XML   1208 POST /RCSAPI/TetraIndividualCallEventsService/ HTTP/1.1
 361 3.387972  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 366 3.390076  172.20.2.78    172.20.2.148    HTTP/XML    129 POST /RCSAPI/TetraSdsEventsService/ HTTP/1.1
 368 3.390687  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 371 3.392782  172.20.2.78    172.20.2.148    HTTP/XML   1286 POST /RCSAPI/TetraIndividualCallEventsService/ HTTP/1.1
 372 3.393349  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 375 3.394835  172.20.2.78    172.20.2.148    HTTP/XML   1027 POST /RCSAPI/TetraIndividualCallEventsService/ HTTP/1.1
 376 3.395053  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 381 3.397131  172.20.2.148   172.20.2.78     HTTP/XML   1026 POST /RCSAPI/TetraMassiveMonitoringGroupService/ HTTP/1.1
 384 3.398073  172.20.2.78    172.20.2.148    HTTP/XML    803 HTTP/1.1 200 OK
 398 3.616314  172.20.2.78    172.20.2.148    HTTP/XML   1024 POST /RCSAPI/TetraIndividualCallEventsService/ HTTP/1.1
 399 3.616752  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 402 3.617840  172.20.2.78    172.20.2.148    HTTP/XML   1008 POST /RCSAPI/TetraIndividualCallEventsService/ HTTP/1.1
 403 3.617961  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 406 3.618932  172.20.2.78    172.20.2.148    HTTP/XML   1012 POST /RCSAPI/TetraIndividualCallEventsService/ HTTP/1.1
 407 3.619167  172.20.2.148   172.20.2.78     HTTP/XML    803 HTTP/1.1 200 OK
 411 3.620375  172.20.2.78    172.20.2.148    HTTP/XML   1027 POST /RCSAPI/TetraIndividualCallEventsService/ HTTP/1.1
```

File: "C:\Users\smardakis\Desktop\login.pca... Packets: 5376 Displayed: 2167 Marked: 0 Load time: 0:00.608 Profile: Default

Figure 9 - ResourceRequrest - NetworkTrace

7.2 TETRAResource Service Web Service

7.2.1 Method: EnteringGroupMonitoringRequest

This message enables the reception of massive monitoring event from the DF-Server.

Input (Literal)

The input of this method is the document element EnteringGroupMonitoringRequest having the structure defined by the following table.

Element	Type	Occurs	Description
RequestID	int	1..1	Id for the DF-Client to match the request. ID generated by the DF-Client that will enable the DF-Client later on to match the response of that request in the case of multiple simultaneous requests.
services	CallbackServiceData	1..1	CallbackServiceData services. The list of callback URLs for all service events that the DF-Client wishes to receive. Expected : TETRA ResourceEvents & TETRAMassiveMonitoringGroupEvents. Request will fail if both are not provided – any others provided will be ignored.

7.2.2 Method: EnteringGroupRequest

Assigning a TCS Client B channel to a DF-Client and loading the requested group . Also enables the reception of group events after a MonitorModeRequest.

Input (Literal)

The input of this method is the document element EnteringGroupRequest having the structure defined by the following table.

Element	Type	Occurs	Description
RequestID	int	1..1	Id for the DF-Client to match the request. ID generated by the DF-Client that will enable the DF-Client later on to match the response of that request in the case of multiple simultaneous requests.
OperativeTactical Address	string	1..1	Subscriber Operative Tactical Address (OPTA).
GSSIUniqueID	int	1..1	Unique ID (more specifically GSSI)
services	CallbackService Data	1..1	CallbackServiceData services The list of callback URLs for all service events that the DF-Client wishes to receive. Expected: TETRAResourceEvents & TETRAGroupEvents. Request will fail if both are not provided – any others provided will be ignored.

7.2.3 METHOD: ENTERINGINDIVIDUALCALLREQUEST

Assigning a TCS Client B channel to a DF-Client. Also enables the reception of individual call events.

Input (Literal)

The input of this method is the document element EnteringIndividualCallRequest having the structure defined by the following table.

Element	Type	Occurs	Description
RequestID	int	1..1	Id for the DF-Client to match the request. ID generated by the DF-Client that will enable the DF-Client later on to match the response of that request in the case of multiple simultaneous requests.
OperativeTacticalAddress	string	1..1	Subscriber Operative Tactical Address (OPTA).
services	CallbackServiceData	1..1	CallbackServiceData services The list of callback URLs for all service events that the DF-Client wishes to receive. Expected: TETRAResourceEvents + TetraIndividualCallEvents. Request will fail if both are not provided – any others provided will be ignored.

7.2.4 METHOD: ENTERINGSDSREQUEST

Assigning a TCS Client (Data only) to a DF-Client. Also enables the reception of SDS events.

Input (Literal)

The input of this method is the document element EnteringSDSRequest having the structure defined by the following table.

Element	Type	Occurs	Description
RequestID	int	1..1	Id for the DF-Client to match the request. ID generated by the DF-Client that will enable the DF-Client later on to match the response of that request in the case of multiple simultaneous requests.
OperativeTacticalAddress	string	1..1	Subscriber Operative Tactical Address (OPTA).
IsDataOnly	boolean	1..1	If TRUE provide a Data only ISSI if available. If not available, a regular voice ISSI will be provided based on current availability rules.
services	CallbackServiceData	1..1	CallbackServiceData services The list of callback URLs for all service events that the DF-Client wishes to receive. Expected: TETRAResourceEvents + TetraSdsEvents. Request will fail if both are not provided – any others provided will be ignored.

7.2.5 Method: EnteringUnitTrackingRequest

Assigning a TCS Client (Data only) to a DF-Client. Also enables the reception of unit tracking events.

Input (Literal)

The input of this method is the document element EnteringUnitTrackingRequest having the structure defined by the following table.

Element	Type	Occurs	Description
RequestID	int	1..1	Id for the DF-Client to match the request. ID generated by the DF-Client that will enable the DF-Client later on to match the response of that request in the case of multiple simultaneous requests.
IsDataOnly	boolean	1..1	If TRUE provide a Data only ISSI if available. If not available, a regular voice ISSI will be provided based on current availability rules.
services	CallbackServiceData	1..1	CallbackServiceData services The list of callback URLs for all service events that the DF-Client wishes to receive. Expected : TETRAResourceEvents + TetraUnitTrackingEvents. Request will fail if both are not provided – any others provided will be ignored.

7.2.6 Method: UseFixedVoiceISSIRequest

Assigning a TCS Voice Client to a DF-Client.

Input (Literal)

The input of this method is the document element UseFixedVoiceISSIRequest having the structure defined by the following table.

Element	Type	Occurs	Description
RequestID	int	1..1	Id for the DF-Client to match the request. ID generated by the DF-Client that will enable the DF-Client later on to match the response of that request in the case of multiple simultaneous requests.
OperativeTactical Address	string	1..1	Subscriber Operative Tactical Address (OPTA).
services	ArrayOfCallback ServiceData	1..1	CallbackServiceData services The list of callback URLs for all service events that the DF-Client wishes to receive. Expected: TETRAResourceEvents. Request will fail if not provided. Others provided will be used to determine which services are activated. LoginEvents is ignored.
Preferred ISSI	int	1..1	Optional: ISSI the DF-Client would like to use.

7.2.7 METHOD: USEFIXEDDATAISSIREQUEST

Assigning a TCS Data Client to a DF-Client.

Input (Literal)

The input of this method is the document element UseFixedDataISSIRequest having the structure defined by the following table.

Element	Type	Occurs	Description
RequestID	int	1..1	Id for the DF-Client to match the request. ID generated by the DF-Client that will enable the DF-Client later on to match the response of that request in the case of multiple simultaneous requests.
OperativeTactical Address	string	1..1	Subscriber Operative Tactical Address (OPTA).
services	ArrayOfCallback ServiceData	1..1	CallbackServiceData services The list of callback URLs for all service events that the DF-Client wishes to receive. Expected : TETRAResourceEvents, and optionally TetraSdsEvents, TetraUnitTrackingEvents. Request will fail if TETRAResourceEvents not provided, others provided will be used to determine which services are activated. All other unexpected services are ignored.
Preferred ISSI	int	1..1	Optional: ISSI the DF-Client would like to use.

7.2.8 METHOD: RESOURCERELEASEREQUEST

Releasing the resource used and own by a DF-Client.

Input (Literal)

The input of this method is the document element ResourceReleaseRequest having the structure defined by the following table.

Type ArrayOfServiceTypes:

Component	Type	Occurs	Description
ArrayOfServiceTypes	ServiceTypes	1..*	Array of ServiceTypes.

7.3 TETRAResourceEventsService Web Service

7.3.1 Method: TetraConnectivityEvent

TetraConnectivityEvent indicates the connectivity state of the TETRA system and Concentrator (DF-Server) System including TCS, DXT, CDD, Voice end Encryption capabilities.

TCS or Login loss of connectivity is considered a failure of all Groups, Individual Call, Unit List, Unit Tracking, and SDS managed by the TCS client. A LinkStateEvent(down) is received for each Group (that the DF-Client has previously sent a SynchronizeResourceRequest) and Individual Call. A UnitListEvent(null) is also received. The DF-Client is expected to flush the TETRA Unit Tracking data and mark TETRA SDS as out-of-service if desired. Voice or Encrpytion loss of connectivity is considered a failure of all Groups and Individual Call affected. A LinkStateEvent(down) is received for each Group (that the DF-Client has previously sent a SynchronizeResourceRequest) and Individual Call. On Encryption loss of connectivity the DF-Client is expected to mark encryption for SDS as out-of-service if desired. DXT and CDD loss of connectivity is simply reported and no action is taken.

On recovery of both TCS and Login, the list of Groups is retrieved from TCS and a new GroupListEvent is received. The current state (including LinkStateEvent) for each Group (that the DF-Client has sent a SynchronizeResourceRequest) and Individual Call is received. If a Group was deleted during the outage, the Group is marked as deleted in the new GroupListEvent. On recovery of both Voice and Encryption, a LinkStateEvent(up) is received for each Group (that the DF-Client has sent a SynchronizeResourceRequest) and Individual Call.

Input (Literal)

The input of this method is the document element TetraConnectivityEvent having the structure defined by the following table.

Element	Type	Occurs	Description
isTcsUp	boolean	1..1	Returns true if TCS is up.
isAudioUp	boolean	1..1	Returns true if audio path is up.
EncryptionState	EncryptionState	1..1	Returns EncryptionOn if Encryption is up, EncryptionOff if Encryption is down, EncryptionDisable if no Encryption is configured in the network.
isLoggedIn	boolean	1..1	Returns true if logged in.
isDxtUp	boolean	1..1	Returns true if DXT is up.

Element	Type	Occurs	Description
isCddUp	boolean	1..1	Returns true if CDD is up.
TcsInfo	string	1..1	TCS Server IP Address
AudioInfo	String	1..1	IP Address of the audio device
EncryptionInfo	string	1..1	IP Address of the end to end encryption device

7.3.2 Method: TetraParametersEvent

TetraParametersEvent indicates the DF-Client address and mnemonic details. For applicable deployments, the DF-Client operative tactical address is also included. This message is sent after resources are allocated to the DF-Client.

Input (Literal)

The input of this method is the document element TetraParametersEvent having the structure defined by the following table.

Element	Type	Occurs	Description
subscriber	TetraSubscriberData	1..1	TETRA subscriber data for the DF-Client.
TETRA Version	String	1..1	TETRA Network version information
Concentrator Version	String	1..1	Concentrator Version

7.3.3 METHOD: CONSOLELOGEVENT

This method is used to provide to the DF-Client additional information or error scenarios coming from the DF-Server (Concentrator) or directly from the TETRA system.

Input

The inputs of this method are the arguments defined by the following table.

Argument	Type	Occurs	Description
ConsoleLog	ConsoleLogCodeType	1..1	The Log code.
paramList	ArrayOfint	1..1	The parameter list.
text	string	1..1	The localized message text to be displayed.

Type Enum Console Log Code:

Value	Description
UnknownCode	The code value is unknown or not supported.
InvalidRequest	An invalid request was received from a DF-Client. Parameter list contains: 1. the request identifier (ConsoleLogRequestType) 2. The entity type (EntityType) 3. The entity unique identifier 4. A reason why the request was considered invalid (ConsoleLogInvalidReason).
InvalidParameter	A request with an invalid parameter was received from a DF-Client. Parameter list contains: 1. The request identifier (ConsoleLogRequestType) 2. The entity type (EntityType) 3. The entity unique identifier 4. The invalid parameter.
SetupCallFailed	Failed to setup Individual Call. Parameter list contains the callId.
ConnectCallFailed	Failed to connect Individual Call. Parameter list contains the callId.

Value	Description
HoldCallFailed	Failed to put the Individual Call on hold. Parameter list contains the callId.
UnholdCallFailed	Failed to take the Individual Call from hold. Parameter list contains the callId.
TransferCallFailed	Failed to transfer the Individual Call. Parameter list contains the callId.
ReleaseCallFailed	Failed to release the call. Parameter list contains the callId.
UnitListFailed	Failed to retrieve the unit list. Parameter list is empty.
UnitTrackingFailed	Unit tracking failure. Parameter list contains the TETRA short subscriber identifier of the unit.
TrackingReleaseCallOk	Release of the tracked unit call was successful. Parameter list contains the TETRA short subscriber identifier of the tracked unit.
TrackingReleaseCallFailed	Failed to release the call of the tracked unit. Parameter list contains the TETRA short subscriber identifier of the tracked unit.
GroupMembersFailed	Failed to add, remove, or retrieve members of the TETRA group. Parameter list contains the TETRA short subscriber identifiers of the base group and optionally the TETRA short subscriber identifier of the member.
LinkState	Link state change for an entity. Parameter list contains: 1. The entity type (EntityType) 2. The entity unique identifier 3. The link state (LinkState)
EncryptionKeyChangeProgress	Progress of an encryption key change request for a TETRA group. Parameter list contains: 1. The entity type (EntityType) 2. The TETRA short subscriber identifier of the group 3. Key change progress (0=started, 1=complete) 4. Optionally, a failure reason (TetraReleaseReason) when the key change is complete.
GroupMonitorModeFailed	Monitor mode request failure for a TETRA group. Parameter list contains the TETRA short subscriber identifier of the group and failure reason (TetraReleaseReason).

Value	Description
TetraLoginFailed	Failed to login to the TETRA network. Parameter list contains the failure reason (TetraReleaseReason).

Type ArrayOfint:

Component	Type	Occurs	Description
ArrayInt	int	0..*	

Type ConsoleLogInvalidReason:

FeatureDisabled
RequestNotAllowed
RequestNotSupported
DuplicateRequest
UnableToProcess
ResourceExhaustion
UnknownCode

Type ConsoleLogRequestType:

AddUnitTrackingRequest
CallModeRequest
ClearCallbackRequest
ConnectCallRequest
DeleteItemRequest
DuplexRequest
EmergencyClearRequest
EncryptionRequest
EncryptionKeyChangeRequest
EnteringGroupMonitoringRequest
EnteringGroupRequest
EnteringIndividualCallRequest
EnteringSDSRequest
EnteringUnitTrackingRequest
GroupAddMemberRequest
GroupDeleteMemberRequest
GroupListRequest

GroupMembersRequest
HoldCallRequest
MonitorModeRequest
PreemptionRequest
ReadItemRequest
ReleaseCallRequest
ReleaseUnitTrackingCallRequest
RemoveUnitTrackingRequest
ResourceReleaseRequest
SendCallbackRequest
SendStatusRequest
SendTextRequest
SendUnitAlertRequest
SetupCallRequest
SynchronizeResourceRequest
TransferCallRequest
TransmitRequest
UnholdCallRequest
UnitListRequest
UnknownRequest
UseFixedVoiceISSIRequest
UseFixedDataISSIRequest

7.3.4 Method: ConsoleLogTextEvent

ConsoleLogTextEvent allows the Radio Service to signal a text string to the DF-Client. The text string is not localized.

Input

The inputs of this method are the arguments defined by the following table.

Argument	Type	Occurs	Description
ConsoleLogText	string	1..1	The text string to display.

7.3.5 METHOD: RESOURCEREQUESTEVENT

This event is providing confirmation of resource assignement.

Input

The inputs of this method are the arguments defined by the following table.

Argument	Type	Occurs	Description
RequestID	int	1..1	Id for the DF-Client to match the request. ID generated by the DF-Client that will enable the DF-Client later on to match the response of that request in the case of multiple simultaneous requests.
IsSuccessful	Boolean	1..1	TRUE if resource granted.
MicrophoneAudioSource	String	1..1	SDP Session profile string.
IndividualCallAudioSource	String	1..1	SDP Session profile string.
GroupAudioSource	ArrayOfstring	1..1	SDP Session profile string. In the case where this is to respond to a fix ISSI resource request.
UniqueID	int	1..1	ISSI used.

Argument	Type	Occurs	Description
ServerAddress	String	1..1	Server that will provide services. It can be the same server as the one that is doing the authentication or in case where a redirection is required, a different IP address can be sent to the DF-Client.
FailureCause	FailureCauseType(Enum) NoISSIAvailable UnsupportedOnCurrentISSI UnidentifedError NoError UnknownUser UnsupportedOnUser	1..1	If IsSuccessful is false, then this indicates the failure type
IsClearOnlyResource	Boolean	1..1	In the case of resource shortage and multiple failures, if the only resources available are resources with encryption capabilities non operational, this parameter is set to ON.

7.3.6 METHOD: RESOURCERELEASEEVENT

Event providing confirmation of resource release request.

Input

The inputs of this method are the arguments defined by the following table.

Argument	Type	Occurs	Description
IsSuccessful	Boolean	1..1	TRUE if resource deallocated.

8 TetraMassiveMonitoringGroup & Events Service Web Service

8.1 Use Cases

Same use cases as TETRA Group & Event use cases.

8.2 TetraMassiveMonitoringGroup Service Web Service

8.2.1 Method: GroupListRequest

Same method as TetraGroupService:GroupListRequest.

8.2.2 Method: MonitorModeRequest

Same method as TetraGroupService:MonitorModeRequest with the exception that the enum used is the MassiveMonitorMode.

8.2.3 Method: SynchronizeResourceRequest

Same method as TetraGroupService:SynchronizeResourceRequest.

8.3 TetraMassiveMonitoringGroupEventsService Web Service

8.3.1 Method: CallbackEvent

Same method as TetraGroupEventsService:CallbackEvent.

8.3.2 Method: CallModeEvent

Same method as TetraGroupEventsService:CallModeEvent.

8.3.3 Method: CallSpeechEvent

Same method as TetraGroupEventsService:CallSpeechEvent.

8.3.4 Method: CallStateEvent

Same method as TetraGroupEventsService:CallStateEvent.

8.3.5 Method: CrossModeEvent

Same method as TetraGroupEventsService:CrossModeEvent.

8.3.6 Method: DuplexEvent

Same method as TetraGroupEventsService:DuplexEvent.

8.3.7 Method: EmergencyEvent

Same method as TetraGroupEventsService:EmergencyEvent.

8.3.8 Method: EncryptionEvent

Same method as TetraGroupEventsService:EncryptionEvent.

8.3.9 Method: GroupCombineEvent

Same method as TetraGroupEventsService:GroupCombineEvent.

8.3.10 Method: GroupListEvent

Same method as TetraGroupEventsService:GroupListEvent.

8.3.11 Method: GroupFeaturesEvent

Same method as TetraGroupEventsService:GroupFeaturesEvent.

8.3.12 Method: GroupMembersEvent

Same method as TetraGroupEventsService:GroupMembersEvent.

8.3.13 Method: IncomingAudioEvent

Same method as TetraGroupEventsService:IncomingAudioEvent.

8.3.14 Method: LinkStateEvent

Same method as TetraGroupEventsService:LinkStateEvent.

8.3.15 Method: MonitorModeEvent

Same method as TetraGroupEventsService:MonitorModeEvent with the exception that the enum used is the MassiveMonitorMode.

8.3.16 Method: PreemptionEvent

Same method as TetraGroupEventsService:PreemptionEvent.

8.3.17 Method: ResourceLabelEvent

Same method as TetraGroupEventsService:ResourceLabelEvent.

8.3.18 METHOD: TRANSMITEVENT

Same method as TetraGroupEventsService:TransmitEvent.

8.3.19 METHOD: UNITEMERGENCYEVENT

Same method as TetraGroupEventsService:UnitEmergencyEvent.

9 TETRAGROUPSERVICE & EVENT WEB SERVICE

9.1 USE CASES

TETRA Group services are available when a B channel or a Voice ISSI is assigned to a DF-Client. They enable the participation of the DF-Client to active communications within that group.

The first use-case is showing how to get access to a specific group and other use cases are demonstrating how to proceed to a transmission, how to handle the reception of an incoming call or how to change parameters on a group. Those use-cases are providing an overview of all the functionalities supported on group. Other functionalities (requests and events) use-cases could be derived from those basic ones.

9.1.1 MONITORMODEREQUEST

The Monitor Mode Request use-case is presenting the selection of a group and the events that are to be expected when the monitoring mode of a group is modified.

By default when a resource is requested, the groups are not monitored. If the resource request is to specifically get access to a group then at that time the group is loaded from TETRA but monitoring it actively is a second step (which is presented here) and some timing constraints from the DXT make it simpler to have the monitor mode request as a separate required second step. Proceeding to a MonitorModeRequest is automatically causing the system to perform a SynchronizeResourceRequest in the background.

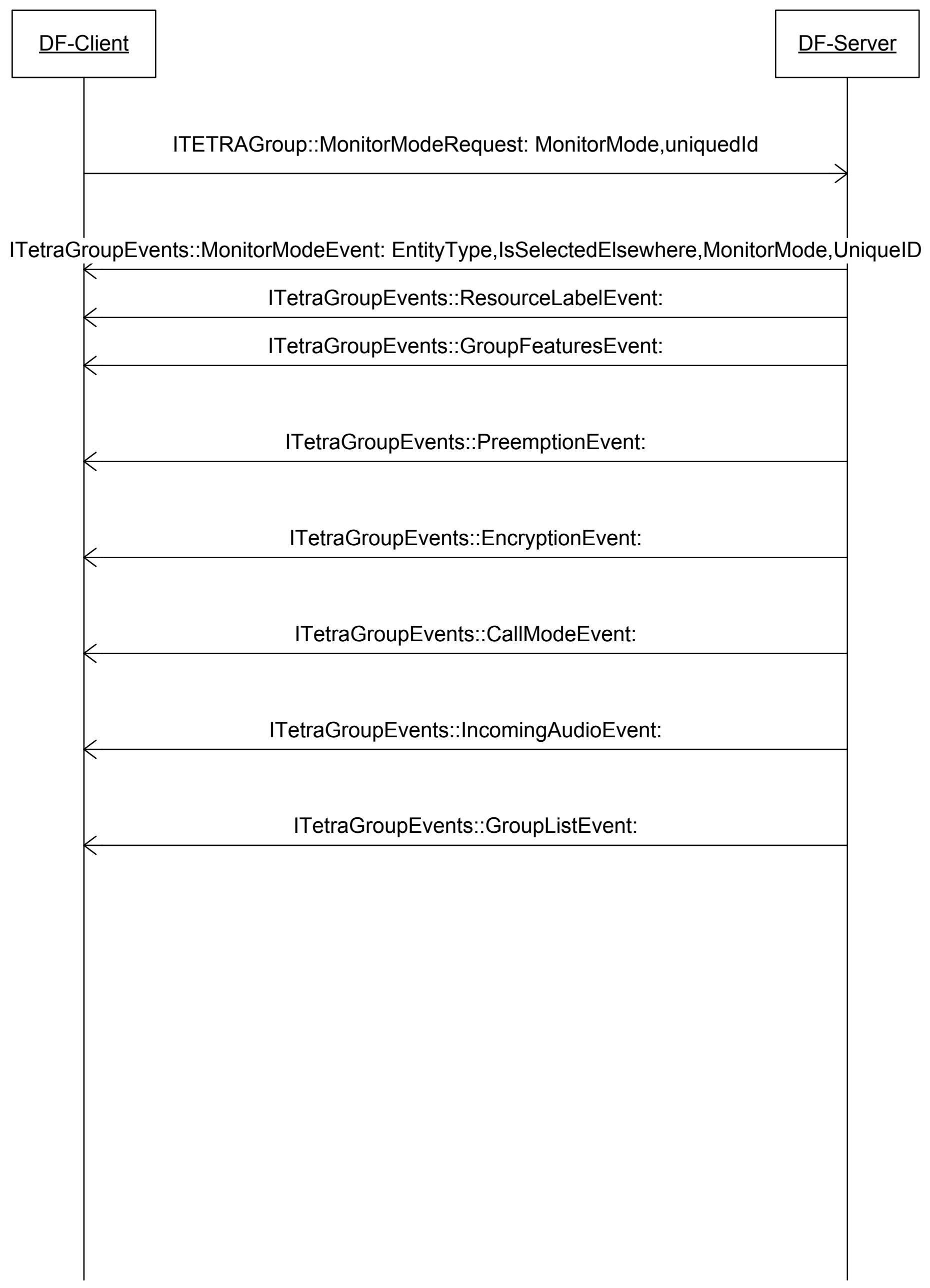

DF-Client
DF-Server
ITETRAGroup::MonitorModeRequest: MonitorMode,uniquedId
ITetraGroupEvents::MonitorModeEvent: EntityType,IsSelectedElsewhere,MonitorMode,UniqueID
ITetraGroupEvents::ResourceLabelEvent:
ITetraGroupEvents::GroupFeaturesEvent:
ITetraGroupEvents::PreemptionEvent:
ITetraGroupEvents::EncryptionEvent:
ITetraGroupEvents::CallModeEvent:
ITetraGroupEvents::IncomingAudioEvent:
ITetraGroupEvents::GroupListEvent:

Figure 10 – Group Service - Monitor Mode Request

9.1.2 Transmit Request

The transmit request use-case is illustrating a transmission from a DF-Client (through a DF-Server) on the TETRA system. There are several prerequisite to this use case. First of all, a valid session is required and the DF-Client has to be logged in. it is also required that a B channel/ISSI is allocated to that DF Client and that the DF Client has selected the group he want to transmit on.

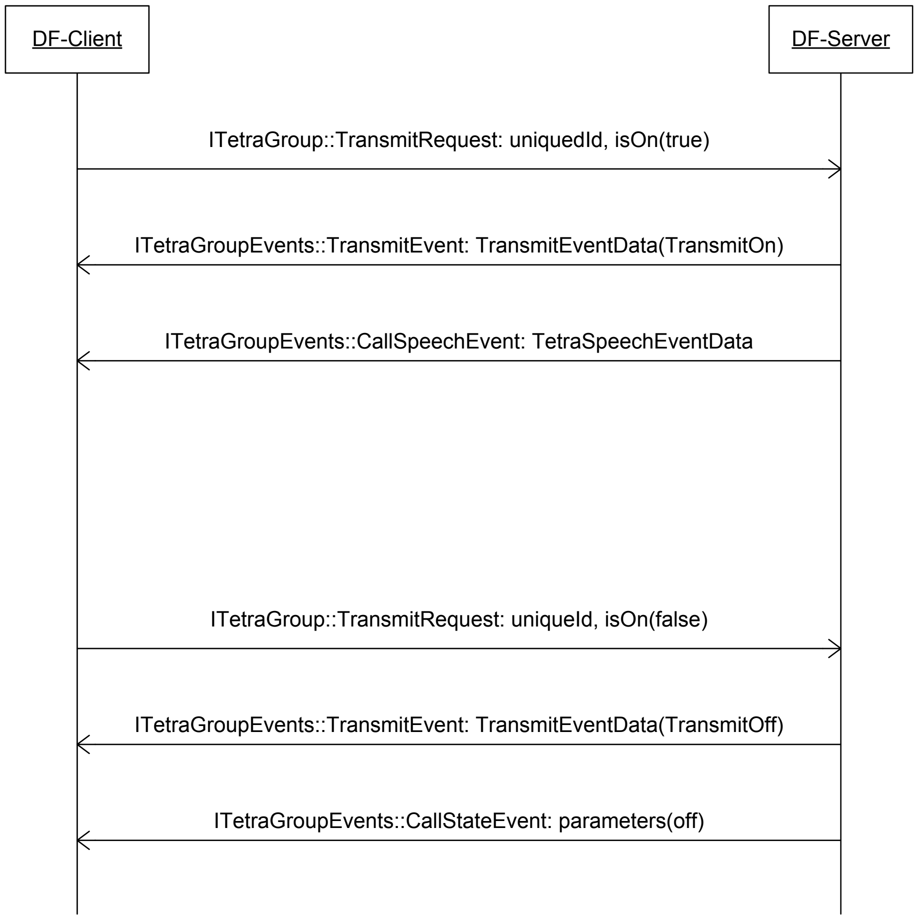
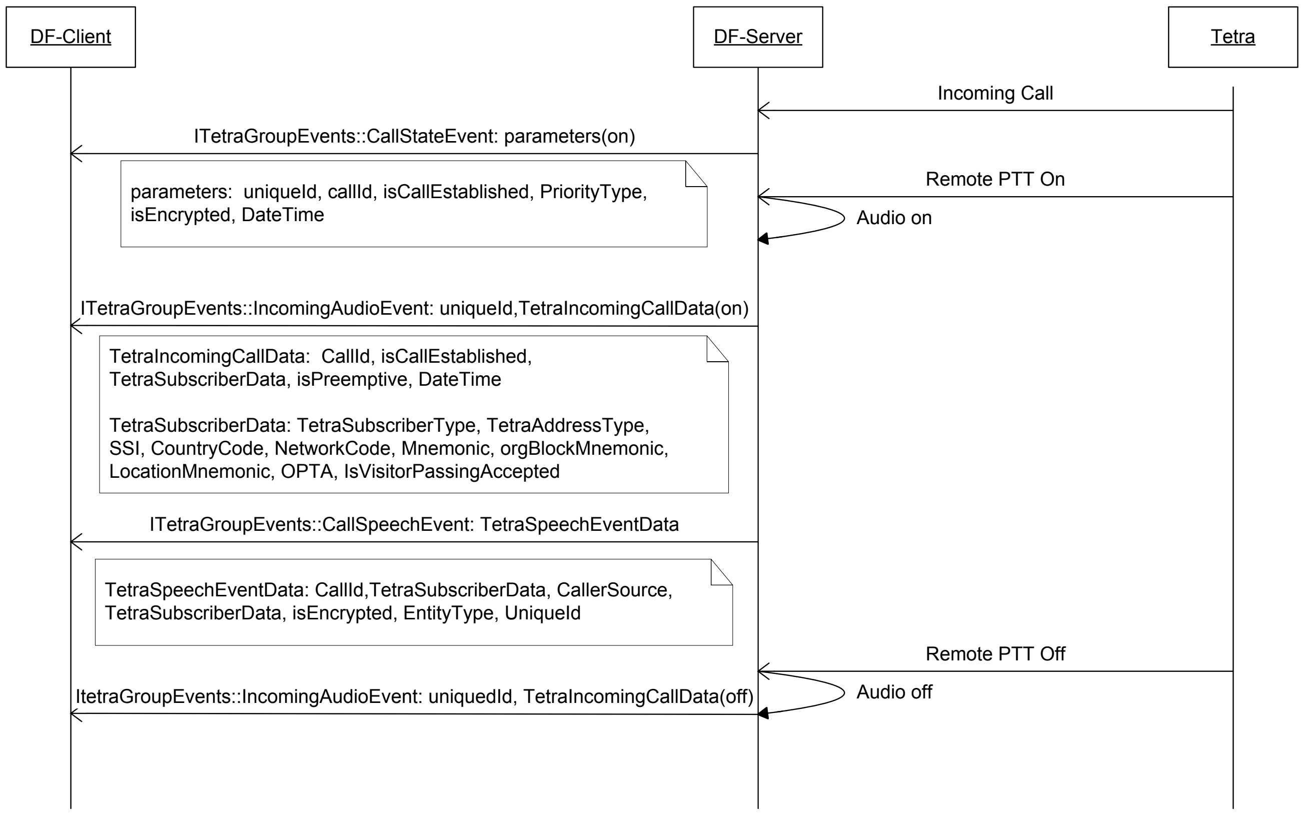

DF-Client
DF-Server
ITetraGroup::TransmitRequest: uniquedId, isOn(true)
ITetraGroupEvents::TransmitEvent: TransmitEventData(TransmitOn)
ITetraGroupEvents::CallSpeechEvent: TetraSpeechEventData
ITetraGroup::TransmitRequest: uniqueId, isOn(false)
ITetraGroupEvents::TransmitEvent: TransmitEventData(TransmitOff)
ITetraGroupEvents::CallStateEvent: parameters(off)

Figure 11 - Group Transmit Request

9.1.3 INCOMING CALL

The incoming call use-case is sort of a mirror of the transmit request use-case since it is illustrating a call triggered by another participant of a group. For the DF-Client to be involved in that incoming call, it has to comply to the same set of prerequisite has the transmit request.



DF-Client
DF-Server
Tetra
Incoming Call
ITetraGroupEvents::CallStateEvent: parameters(on)
parameters: uniqueId, callId, isCallEstablished, PriorityType, isEncrypted, DateTime
Remote PTT On
Audio on
ITetraGroupEvents::IncomingAudioEvent: uniqueId,TetraIncomingCallData(on)
TetraIncomingCallData: CallId, isCallEstablished, TetraSubscriberData, isPreemptive, DateTime
TetraSubscriberData: TetraSubscriberType, TetraAddressType, SSI, CountryCode, NetworkCode, Mnemonic, orgBlockMnemonic, LocationMnemonic, OPTA, IsVisitorPassingAccepted
ITetraGroupEvents::CallSpeechEvent: TetraSpeechEventData
TetraSpeechEventData: CallId,TetraSubscriberData, CallerSource, TetraSubscriberData, isEncrypted, EntityType, UniqueId
Remote PTT Off
Audio off
ItetraGroupEvents::IncomingAudioEvent: uniquedId, TetraIncomingCallData(off)

Figure 12 - Group Incoming Call

9.1.4 ENCRYPTION REQUEST

The Encryption request is a very simple use-case and most of the TETRA group requests are similar to this one in the sense that it is a direct match of a request and event. They are all dependant on the same set of prerequisite as the tranmsit request use-case.

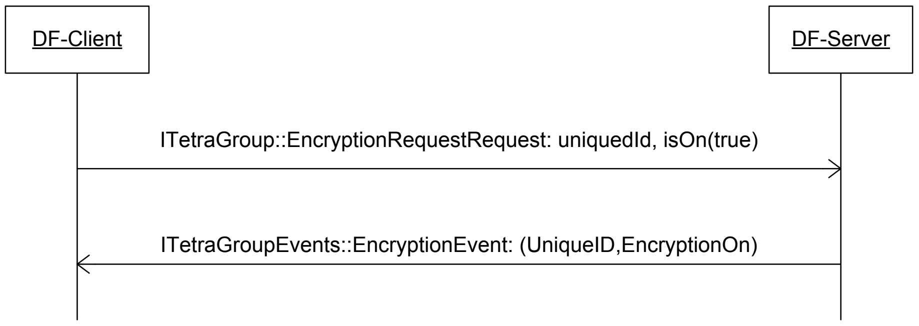

DF-Client
DF-Server
ITetraGroup::EncryptionRequestRequest: uniquedId, isOn(true)
ITetraGroupEvents::EncryptionEvent: (UniqueID,EncryptionOn)

Figure 13 - Group Encryption Request

9.2 TetraGroupService Web Service

9.2.1 Method: GroupListRequest

This method will force the DF-Server to return the list of groups the DF-Client as the right to access. The groupListEvent will contain the groups list.

Input (Literal)

The input of this method is the document element GroupListRequest and is always empty.

9.2.2 Method: CallModeRequest

CallModeRequest changes the DF-Client emergency state within the group.CallModeEvent indicates the DF-Client emergency state. The call mode (normal or emergency) is applied whenever the DF-Client speaks in the group. An ongoing normal priority group call is upgraded to emergency priority if the DF-Client speaks with call mode of emergency. Clearing the group call resets the call mode to normal. The call can be cleared by the DF-Client or the network. Changing the call mode to normal (from emergency) will clear the group call only if the DF-Client setup the emergency call. Failure to clear the emergency call is reported in a console log.

Input (Literal)

The input of this method is the document element CallModeRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	GSSI of the group.
callMode	CallMode	1..1	

9.2.3 METHOD: CLEARCALLBACKREQUEST

ClearCallbackRequest clears all active callback requests received from the subscriber.

Input (Literal)

The input of this method is the document element ClearCallbackRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
callerId	TetraSubscriberData	1..1	Details about the subscriber requesting the callback.

9.2.4 METHOD: EMERGENCYCLEARREQUEST

EmergencyClearRequest clears an emergency on the group.The emergency group call is cleared.

Input (Literal)

The input of this method is the document element EmergencyClearRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.

9.2.5 METHOD: ENCRYPTIONREQUEST

EncryptionRequest requests to change the encryption mode on a group. EncryptionEvent returns the encryption mode on a group.

Input (Literal)

The input of this method is the document element EncryptionRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
isOn	boolean	1..1	Set to true to enable encryption. Set to false to disable encryption.

9.2.6 METHOD: GROUPADDMEMBERREQUEST

GroupAddMemberRequest adds a member to a group. GroupMembersEvent is received whenever the group membership changes. The complete member list for the group is included. ConsoleLogEvent is received if the member cannot be added to the group.

Input (Literal)

The input of this method is the document element GroupAddMemberRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The group identifier (GSSI) to which the member is added.
member	TetraSubscriberData	1..1	The subscriber to add to the group. Radio units, other groups, and external systems (via generic 4 wire interfaces) can all be added as members to a group. A radio unit can be added as permanent or visiting members of a group. Groups are combined by adding constituent groups as members to a base group. Up to eight groups can be combined together in this way. Communication targeted to any of these groups is then delivered to all groups of the combined group. A group can be part of only one combined group at a time.
isPermanent	boolean	1..1	Set to true if the subscriber should be permanently added to the group.
isForced	boolean	1..1	Set to true to force combine groups using different air encryption.

9.2.7 METHOD: GROUPDELETEMEMBERREQUEST

GroupDeleteMemberRequest deletes a member from a group. GroupMembersEvent is received whenever the group membership changes. The complete member list for the group is included. ConsoleLogEvent is received if the member cannot be deleted from the group.

Input (Literal)

The input of this method is the document element GroupDeleteMemberRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
member	TetraSubscriberData	1..1	

9.2.8 METHOD: GROUPMEMBERREQUEST

GroupMemberRequest requests the members of a group. The GroupMembersEvent returns the members of the group. The complete member list for the group is included. ConsoleLogEvent is received if the member list cannot be retrieved.

Input (Literal)

The input of this method is the document element GroupMemberRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.

9.2.9 METHOD: MONITORMODEREQUEST

MonitorModeRequest requests to change the monitor mode of a group. MonitorModeEvent returns the monitor mode state.

Input (Literal)

The input of this method is the document element MonitorModeRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
monitorMode	MonitorMode	1..1	The monitor mode. Select is equivalent to Active use and Unselect is equivalent tu use.

9.2.10 METHOD: PREEMPTIONREQUEST

PreemptionRequest enables or disables DF-Client pre-emption within the group. The DF-Client transmits with a higher priority when pre-emption is enabled. PreemptionEvent is received whenever the DF-Client pre-emption state changes.

Input (Literal)

The input of this method is the document element PreemptionRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
isOn	boolean	1..1	Set to true to turn on.

9.2.11 METHOD: RELEASECALLREQUEST

ReleaseCallRequest is used to release the group call. The CallStateEvent is received when the group call is released. A ConsoleLogEvent is received if the group call cannot be released.

Input (Literal)

The input of this method is the document element ReleaseCallRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI fo the Group.
isForced	boolean	1..1	Set to true to force the call release. If not set to TRUE, the call continue without the participant who requested to release the call.

9.2.12 METHOD: TRANSMITREQUEST

TransmitRequest is used when the DF-Client wishes to transmit on a group. TransmitEvent indicates a DF-Client talking on a group.

Input (Literal)

The input of this method is the document element TransmitRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
isOn	boolean	1..1	Set to true to start transmit. Set to false to stop transmit.

9.2.13 Method: SynchronizeResourceRequest

SynchronizeResourceRequest requests to update the DF-Client with all the current group states.

Input (Literal)

The input of this method is the document SynchronizeResourceRequest element having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The unique id of the group (GSSI).

9.2.14 Method: GroupLoadRequest

Description

Client indicates which resources to load from the TETRA system. An empty list will request the system to load all the groups available.

Input (Literal)

The input of this method is the document element ArrayOfGroupLoadData having the structure defined by the following table.

Type ArrayOfGroupLoadData:

Component	Type	Occurs	Description
GroupLoadData	GroupLoadData	1..*	

Type GroupLoadData:

Component	Type	Occurs	Description
UniqueId	int	1..1	Gets or sets UniqueId.

9.2.15 METHOD: ENCRYPTIONKEYCHANGEREQUEST

Description

EncryptionKeyChangeRequest requests to change the encryption key on a group. ConsoleLogEvent returns the progress of the key negotiation.

Input (Literal)

The input of this method is the document element EncryptionKeyChangeRequest having the structure defined by the following table.

Component	Type	Occurs	Description
uniqueId	int	1..1	The unique id of the group (GSSI).

9.3 TetraGroupEventsService Web Service

9.3.1 Method: CallbackEvent

CallbackEvent is received whenever a callback request is cleared or received from a subscriber in the TETRA network. All active callback requests are sent to the DF-Client initially and thereafter as changes occur.

Input (Literal)

The input of this method is the document element CallbackEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
dataList	ArrayOfCallbackData	1..1	Returns the callback data.

9.3.2 Method: CallModeEvent

CallModeRequest changes the DF-Client emergency state within the group. CallModeEvent indicates the DF-Client emergency state.

The call mode (normal or emergency) is applied whenever the DF-Client speaks in the group. An ongoing normal priority group call is upgraded to emergency priority if the DF-Client speaks with call mode of emergency. Clearing the group call resets the call mode to normal. The call can be cleared by the DF-Client or the network. Changing the call mode to normal (from emergency) will clear the group call only if the DF-Client setup the emergency call. Failure to clear the emergency call is reported in a console log.

Input (Literal)

The input of this method is the document element CallModeEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
mode	CallMode	1..1	Call mode: ModeNormal, ModeEmergency, ModeDisable ModeDisable indicates DF-Client requests to change the DF-Client emergency state are not supported such as when the group is event monitored or not monitored or provisioning "allow-emergency-calls" is false for the DF-Client.

9.3.3 Method: CallSpeechEvent

CallSpeechEvent provides details of the talking party in a group.

Input (Literal)

The input of this method is the document element CallSpeechEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	TetraSpeechEventData	1..1	TETRA speech data.

Technical Description of Digitalfunkstecker V0.2

Bundesverband Professioneller Mobilfunk (PMeV)

9.3.4 Method: CallStateEvent

CallStateEvent indicates the attributes of the group call.

Input (Literal)

The input of this method is the document element CallStateEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
callId	int	1..1	The call identifier.
isCallEstablished	boolean	1..1	True if call established.
callPriority	PriorityType	1..1	The Call priority type.
isEncrypted	boolean	1..1	True if call is encrypted.
releaseReason	TetraReleaseReasonData	1..1	The reason for the call to be released.
timestamp	dateTime	1..1	The Date Time.

9.3.5 Method: CrossModeEvent

CrossModeEvent indicates a mismatch between the encryption setting and the audio on the group. A mismatch occurs if the group call is encrypted but is transmitting or receiving clear audio, as well as if the group call is not encrypted but is transmitting or receiving encrypted audio.

Input (Literal)

The input of this method is the document element CrossModeEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
isCrossMode	boolean	1..1	Returns true if there is a mismatch.

9.3.6 METHOD: DUPLEXEVENT

DuplexEvent indicates the duplex mode on a group. TETRA groups support only half duplex. DF-Client requests to change the duplex mode are not supported.

Input (Literal)

The input of this method is the document element DuplexEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
state	DuplexMode	1..1	Duplex mode. Only half duplex in this specific case.

9.3.7 METHOD: EMERGENCYEVENT

EmergencyEvent indicates the group emergency state.

Input (Literal)

The input of this method is the document element EmergencyEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
state	EmergencyChanState (Enum): EmergencyChanStateNone EmergencyChanStateAcked EmergencyChanStateCleared EmergencyChanStateNew	1..1	The emergency state.

9.3.8 METHOD: ENCRYPTIONEVENT

EncryptionEvent indicates the encryption mode on a group.

Input (Literal)

The input of this method is the document element EncryptionEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
state	EncryptionState	1..1	EncryptionState. Disabled indicates that DF-Client requests to change the encryption mode are not supported.

9.3.9 METHOD: GROUPCOMBINEEVENT

GroupCombineEvent indicates whether or not the group is part of a combined group as either the base group or a constituent group. The GroupAddMemberRequest is used to combine groups by adding a constituent group as a member of a base group. Up to eight groups can be combined together in this way. Communication targeted to any of these groups is then delivered to all groups of the combined group. A group can be part of only one combined group at a time.

Input (Literal)

The input of this method is the document element GroupCombineEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
mode	GroupCombineMode (Enum): GroupCombineModeOff GroupCombineModeBaseGroup GroupCombineModeMemberGroup	1..1	Group combine mode.
BaseGSSI	int	1..1	GSSI of the base group.

Technical Description of Digitalfunkstecker V0.2

Bundesverband Professioneller Mobilfunk (PMeV)

9.3.10 METHOD: GROUPFEATURESEVENT

GroupFeaturesEvent returns group specific configuration data.

Input (Literal)

The input of this method is the document element GroupFeaturesEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	GroupFeaturesData	1..1	Group configuration data.

Type GroupFeaturesData:

Component	Type	Occurs	Description
EntityType	EntityType	1..1	EntityType
GroupMode	GroupMode (Enum): GroupNormal GroupSemiConf GroupBroadcast GroupBackground GroupVirtual	1..1	GroupMode. All members equals Only DF-Client is hearing a unit One way call Broadcast group always scanned Not supported
IsOutOfUse	boolean	1..1	Value indicating whether out of use (true or false).
OrgBlockId	string	1..1	OrgBlockId.
OrgBlockMnemonic	string	1..1	OrgBlockMnemonic.
UniqueId	int	1..1	The GSSI of the Group.
IsEncryptionSupported	boolean	1..1	Is the encryption feature supported.

9.3.11 METHOD: GROUPLISTEVENT

GroupListEvent returns the complete list of TETRA groups that the DF-Client is authorized to access. The GroupListEvent is received on group resource allocation and whenever groups are added, modified, or deleted in the TETRA system.

On resource allocation, the GroupListEvent contains the complete list of available groups. Whenever groups are added or modified in the TETRA system, the GroupListEvent contains the specific subset of modified groups. Whenever groups are deleted in the TETRA system, the GroupListEvent contains all deleted groups in the list and marked as deleted.

The DF-Client should be prepared to receive more than one event if the list of groups is large.

Input (Literal)

The input of this method is the document element GroupListEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	ArrayOfGroupListData	1..1	Group List Data.

Type ArrayOfGroupListData:

Component	Type	Occurs	Description
GroupListData	GroupListData	1..*	

Type GroupListData:

Component	Type	Occurs	Description
EntityType	EntityType	1..1	EntityType.
IsDeleted	boolean	1..1	Value indicating whether deleted (true or false).
UniqueId	int	1..1	The GSSI of the Group.
IsModitoredInSystem	boolean	1..1	Is the GSSI already monitored
IsOutOfUse	Boolean	1..1	Is the GSSI out of use
Label	String	1..1	Label of the GSSI

Technical Description of Digitalfunkstecker V0.2

Bundesverband Professioneller Mobilfunk (PMeV)

9.3.12 METHOD: GROUPLOADINGEVENT

GroupLoadingEvent indicates the group loading status. The action of loading group is the process done in the Concentrator (DF-Server) to receive from the TETRA system the information about the group it has access to for a specific TCS Client. The amount of information is significant and the process is intensive. It subsequently enables the Concentrator (DF-Server) to provide TETRA group services efficiently.

Input (Literal)

The input of this method is the document element GroupLoadingEvent having the structure defined by the following table.

Element	Type	Occurs	Description
state	LoadingState (Enum): LoadedNone LoadingAll LoadingSubset LoadedAll LoadedSubset	1..1	Group loading state.

9.3.13 METHOD: GROUPMEMBERSEVENT

The GroupMembersEvent returns the members of the group. The complete member list for the group is included. DF-Client shall support receiving multiple events in the case the member list is too large for a single event.

Input (Literal)

The input of this method is the document element GroupMembersEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	GroupMembersData	1..1	Group member data.

Type GroupMembersData

Component	Type	Occurs	Description
EntityType	EntityType	1..1	EntityType
IsStartOfList	boolean	1..1	True indicate that it is the beginning of the list and false indicate that it is an additional message and that the content should be appended to the previous elements received.
MemberList	ArrayOfGroupMemberData	1..1	Group members.
UniqueId	int	1..1	The GSSI of the Group..

Type ArrayOfGroupMemberData:

Component	Type	Occurs	Description
GroupMemberData	GroupMemberData	1..*	

Type GroupMemberData:

Component	Type	Occurs	Description
IsRuDownloadPendingForAdd	boolean	1..1	Value indicating whether a radio unit download is pending to add to the group (true or false).
IsRuDownloadPendingForDelete	boolean	1..1	Value indicating whether a radio unit download is pending to remove from the group (true or false).
IsRuDownloadPending	boolean	1..1	Value indicating whether a radio unit download is pending (true or false).
IsRuPermanent	boolean	1..1	Value indicating whether a radio unit is permanent in the group (true or false).
IsRuRegInGroupArea	boolean	1..1	Value indicating whether a radio unit is registered in the group area (true or false).
IsRuSelectedGroup	boolean	1..1	Value indicating whether a radio unit is in the selected group (true or false).
MemberId	TetraSubscriberData	1..1	Member subscriber information.

Component	Type	Occurs	Description
MemberState	GroupMemberState (Enum): MemberStateNone MemberStateSelect MemberStateUnselect MemberStateEventMonitored MemberStateTracked MemberStateIncoming MemberStateOutgoing MemberStateBoth MemberStateBase MemberStateConstituent	1..1	GroupMemberState. Default State Select= Active Use Unselect= Use Event Monitored Unit tracked G4wif Rx G4wif Tx G4wif Tx and Rx Group Combined base Group Combined member

9.3.14 METHOD: INCOMINGAUDIOEVENT

IncomingAudioEvent indicates a non-DF-Client talking on a group.

Input (Literal)

The input of this method is the document element IncomingAudioEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
data	TetraIncomingAudioData	1..1	Incoming call data.

9.3.15 METHOD: LINKSTATEEVENT

LinkStateEvent indicates the link state of the group.

Input (Literal)

The input of this method is the document element LinkStateEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	LinkStateData	1..1	Link state data.

9.3.16 Method: MonitorModeEvent

MonitorModeRequest requests to change the monitor mode of a group. MonitorModeEvent indicates the monitor mode of the group.

Input (Literal)

The input of this method is the document element MonitorModeEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	MonitorModeEventData	1..1	MonitorModeEventData.

Type : MonitorModeEventData

Component	Type	Occurs	Description
EntityType	EntityType	1..1	Entity type.
IsSelectedSomewhere	boolean	1..1	Value indicating whether the entity was selected by a DF-Client (true or false).
MonitorMode	MonitorMode	1..1	Monitor mode. Select is equivalent to active use and Unselect is equivalent to use.
UniqueId	int	1..1	The GSSI of the Group.
AudioSource	String	1..1	SDP Session profile string. Only applicable in the Select and Unselect Monitor mode.

9.3.17 Method: PreemptionEvent

PreemptionRequest enables or disables DF-Client pre-emption within the group. The DF-Client transmits with a higher priority when pre-emption is enabled. PreemptionEvent indicates the DF-Client pre-emption state.

Input (Literal)

The input of this method is the document element PreemptionEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
State	PreemptionState	1..1	The preemption state. Disabled state indicates DF-Client requests to enable pre-emption are not supported.

9.3.18 Method: ResourceLabelEvent

ResourceLabelEvent identifies the TETRA group unique ID and label.

Input (Literal)

The input of this method is the document element ResourceLabelEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
label	string	1..1	Group label.

9.3.19 METHOD: TRANSMITEVENT

TransmitRequest is used when the DF-Client wishes to transmit on a group. TransmitEvent indicates a DF-Client talking on a group.

Input (Literal)

The input of this method is the document element TransmitEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	TransmitEventData	1..1	The transmit event data.

9.3.20 Method: UnitEmergencyEvent

UnitEmergencyEvent reports the emergency status of individual radio units in the group. All units currently in emergency are provided to the DF-Client when the group is monitored and thereafter as changes occur.

Input (Literal)

The input of this method is the document element UnitEmergencyEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	The GSSI of the Group.
data	UnitEmergencyData	1..1	Unit emergency data.

Type UnitEmergencyData:

Component	Type	Occurs	Description
IsEmergencyState	boolean	1..1	Value indicating whether a unit is in an emergency state (true or false).
TimeStamp	dateTime	1..1	Unit time stamp.
UnitId	TetraSubscriberData	1..1	Subscriber data for the unit.

10 TetraIndividualCall & Event Service Web Service

10.1 Use Cases

The TETRA Individual Call services are following the same philosophy as the TETRA Group Calls and the mechanism linking the requests and events is the same. One use-case is presented showing the complete flow of information about an individual call and the other use-cases can be derived from the TETRA Group Call ones.

10.1.1 Individual Call Received and answered

The use-case is showing the process of receiving an individual call and answering it. This use case like all the other individual call use cases has to comply with multiple prerequisite. First of all, a valid session is required and the DF-Client has to be logged in. it is also required that a B channel/ISSI.

The first part of the use-case is about receiving an individual call and receiving the audio associated with it. The second half of the use-case is showing the DF-Client responding to the originator of the call by sending voice back.

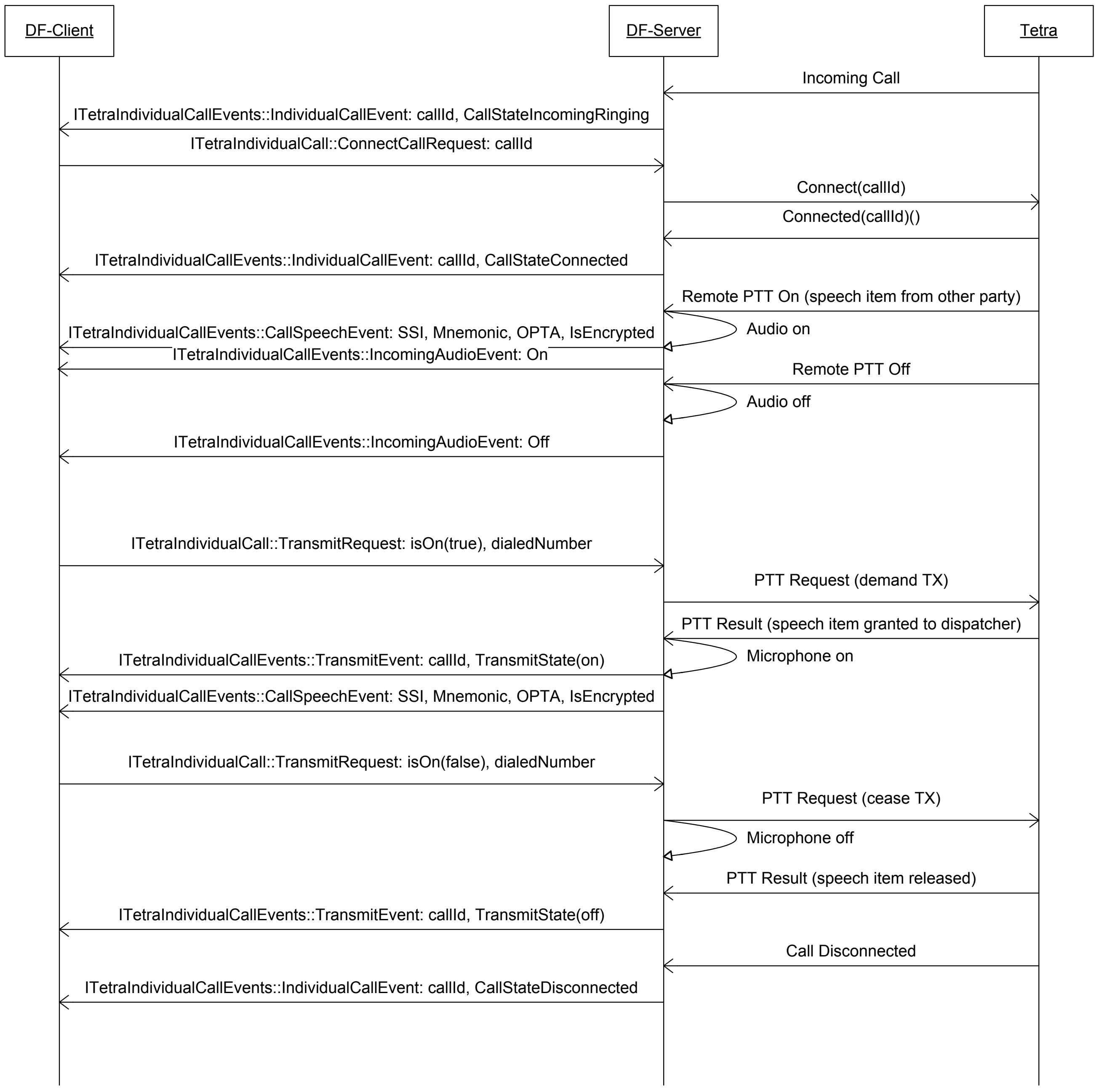

DF-Client
DF-Server
Tetra
Incoming Call
ITetraIndividualCallEvents::IndividualCallEvent: callId, CallStateIncomingRinging
ITetraIndividualCall::ConnectCallRequest: callId
Connect(callId)
Connected(callId)()
ITetraIndividualCallEvents::IndividualCallEvent: callId, CallStateConnected
Remote PTT On (speech item from other party)
Audio on
ITetraIndividualCallEvents::CallSpeechEvent: SSI, Mnemonic, OPTA, IsEncrypted
ITetraIndividualCallEvents::IncomingAudioEvent: On
Remote PTT Off
Audio off
ITetraIndividualCallEvents::IncomingAudioEvent: Off
ITetraIndividualCall::TransmitRequest: isOn(true), dialedNumber
PTT Request (demand TX)
PTT Result (speech item granted to dispatcher)
Microphone on
ITetraIndividualCallEvents::TransmitEvent: callId, TransmitState(on)
ITetraIndividualCallEvents::CallSpeechEvent: SSI, Mnemonic, OPTA, IsEncrypted
ITetraIndividualCall::TransmitRequest: isOn(false), dialedNumber
PTT Request (cease TX)
Microphone off
PTT Result (speech item released)
ITetraIndividualCallEvents::TransmitEvent: callId, TransmitState(off)
Call Disconnected
ITetraIndividualCallEvents::IndividualCallEvent: callId, CallStateDisconnected

Figure 14 - Individual Call – Received and Answer

Technical Description of Digitalfunkstecker V0.2

Bundesverband Professioneller Mobilfunk (PMeV)

10.2 TetraIndividualCall Service Web Service

10.2.1 Method: CallModeRequest

CallModeRequest changes the DF-Client emergency state for the Individual Call. CallModeEvent returns the DF-Client emergency state.

The call mode (normal or emergency) is applied whenever the DF-Client places an outgoing call or speaks in a call. The outgoing call has emergency priority when the call mode is emergency. DF-Client speech items have emergency priority when the call mode is emergency. Clearing the call resets the call mode to normal. The call can be cleared by the DF-Client or the network. The Call mode can be change while an active call is in progress but will only be applicable on the establishment of the next call.

Input (Literal)

The input of this method is the document element CallModeRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
callMode	CallMode	1..1	Individual Call call mode.

10.2.2 Method: ClearCallbackRequest

ClearCallbackRequest clears all active callback requests received from the subscriber.

Input (Literal)

The input of this method is the document element ClearCallbackRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
callerId	TetraSubscriberData	1..1	Details about the subscriber requesting the callback.

10.2.3 Method: ConnectCallRequest

ConnectCallRequest answers an incoming call.

IndividualCallEvent is received on success, otherwise ConsoleLogEvent is received. There must not be an active call in progress in order for this request to succeed.

Input (Literal)

The input of this method is the document element ConnectCallRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
callId	int	1..1	The callId parameter as signaled in the IndividualCallEvent.

10.2.4 Method: DuplexRequest

DuplexRequest requests to change the duplex mode on an Individual Call. DuplexEvent returns the duplex mode on an Individual Call.

The duplex mode is applied when making outgoing calls and when answering incoming calls. Incoming full duplex calls are converted to half duplex if answered in half duplex mode.

Input (Literal)

The input of this method is the document element DuplexRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
duplexMode	DuplexMode	1..1	Duplex mode: HalfDuplex, FullDuplex

10.2.5 Method: EncryptionRequest

EncryptionRequest requests to change the encryption mode on an Individual Call. EncryptionEvent returns the encryption mode on an Individual Call.

Input (Literal)

The input of this method is the document element EncryptionRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
isOn	boolean	1..1	Set to true to enable encryption. Set to false to disable encryption.

10.2.6 Method: HoldCallRequest

HoldCallRequest places the currently active call on hold.

IndividualCallEvent is received on success, otherwise ConsoleLogEvent is received. There must be an active call in progress in order for this request to succeed.

Input (Literal)

The input of this method is the document element HoldCallRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.

10.2.7 Method: PreemptionRequest

PreemptionRequest enables or disables DF-Client pre-emption within the call. The DF-Client transmits with a higher priority when pre-emption is enabled. PreemptionEvent indicates the DF-Client pre-emption state.

Input (Literal)

The input of this method is the document element PreemptionRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
isOn	boolean	1..1	Set to true to turn preemption on, false to turn it off.

10.2.8 Method: ReleaseCallRequest

ReleaseCallRequest releases an incoming call or the currently active call. IndividualCallEvent is received on success, otherwise ConsoleLogEvent is received.

Input (Literal)

The input of this method is the document element ReleaseCallRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
callId	int	1..1	The callId parameter as signaled in the IndividualCallEvent.
isForced	boolean	1..1	Set to true to force a release.

10.2.9 METHOD: SETUPCALLREQUEST

SetupCallRequest allows the DF-Client to place an outgoing call to a subscriber in the TETRA network. All numbering plans can be used (TETRA, FSSN, MSISDN, External). The numbering plan is determined by either the leading digit of the dialedNumber or by the dialedNumber being an exact match to a provisioned special number.

IndividualCallEvent is received on success, otherwise ConsoleLogEvent is received. There must not be an active call in progress in order for this request to succeed.

Input (Literal)

The input of this method is the document element SetupCallRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
dialedNumber	string	1..1	The number of the subscriber to call. The dialedNumber parameter includes the leading digit. TETRA Numbering plan is using a "-" as a leading digit and also between the 3 components of the id. For example: "-XXX-YYY-ZZZ"
isAmbianceListening	boolean	1..1	The isAmbianceListening parameter. An ambiance listening call is connected without the called subscriber knowledge. Audio is received from the called subscriber but the DF-Client is unable to transmit on the ambiance listening call.

10.2.10 Method: TransferCallRequest

TransferCallRequest transfers the currently active call to a third party in the TETRA network. All numbering plans can be used (TETRA, FSSN, MSISDN, External). The numbering plan is determined by either the leading digit of the dialedNumber or by the dialedNumber being an exact match to a provisioned special number.

IndividualCallEvent is received on success, otherwise ConsoleLogEvent is received. There must be an active call in progress in order for this request to succeed.

Input (Literal)

The input of this method is the document element TransferCallRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
dialedNumber	string	1..1	The number of the third party subscriber to call. The dialedNumber parameter includes the leading digit. TETRA Numbering plan is using a "-" as a leading digit and also between the 3 components of the id. For example: "-XXX-YYY-ZZZ"

10.2.11 Method: TransmitRequest

TransmitRequest is used when the DF-Client wishes to transmit on the Individual Call. TransmitEvent indicates a DF-Client talking on the Individual Call.

Input (Literal)

The input of this method is the document element TransmitRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
isOn	boolean	1..1	Set to true to start transmit. Set to false to stop transmit.
dialedNumber	string	1..1	The dialedNumber is used to signal a direct call setup. The TransmitEvent(on) is returned once the direct call is connected. The call remains connected if the TransmitRequest(off) is received. The direct call can be cleared by the ReleaseCallRequest. Direct calls to another DF-Client are automatically converted to a hook style call. TETRA Numbering plan is using a "-" as a leading digit and also between the 3 components of the id. For example: "-XXX-YYY-ZZZ"

10.2.12 Method: UnholdCallRequest

UnholdCallRequest connects a call on hold. IndividualCallEvent is received on success, otherwise ConsoleLogEvent is received. There must not be an active call in progress in order for this request to succeed.

Input (Literal)

The input of this method is the document element UnholdCallRequest having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
callId	int	1..1	The callId parameter as signaled in the IndividualCallEvent.

10.3 TetraIndividualCallEventsService Web Service

10.3.1 Method: CallbackEvent

CallbackEvent is received whenever a callback request is cleared or received from a subscriber in the TETRA network. All active callback requests are sent to the DF-Client during configuration and thereafter as changes occur.

Input (Literal)

The input of this method is the document element CallbackEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
dataList	ArrayOfCallbackData	1..1	Returns the callback data.

10.3.2 Method: IncomingAudioEvent

IncomingAudioEvent indicates a non-DF-Client talking on an individual call.

Input (Literal)

The input of this method is the document element IncomingAudioEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
data	TetraIncomingAudioData	1..1	Incoming call data.

10.3.3 METHOD: CALLMODEEVENT

CallModeRequest changes the DF-Client emergency state for the Individual Call. CallModeEvent returns the DF-Client emergency state.

The call mode (normal or emergency) is applied whenever the DF-Client places an outgoing call or speaks in a call. The outgoing call has emergency priority when the call mode is emergency. DF-Client speech items have emergency priority when the call mode is emergency. Clearing the call resets the call mode to normal. The call can be cleared by the DF-Client or the network.

Input (Literal)

The input of this method is the document element CallModeEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
mode	CallMode	1..1	Call mode: ModeNormal, ModeEmergency, ModeDisable ModeDisable indicates DF-Client requests to change the DF-Client emergency state are not supported such as when provisioning "allow-emergency-calls" is false for the user.

10.3.4 METHOD: CALLSPEECHEVENT

CallSpeechEvent provides details of the talking party in an Individual Call.

Input (Literal)

The input of this method is the document element CallSpeechEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	TetraSpeechEventData	1..1	TETRA speech data.

10.3.5 METHOD: CROSSMODEEVENT

CrossModeEvent indicates a mismatch between the encryption setting and the audio on the Individual Call. A mismatch occurs if the call is encrypted but is transmitting or receiving clear audio, as well as if the call is not encrypted but is transmitting or receiving encrypted audio.

Input (Literal)

The input of this method is the document element CrossModeEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
isCrossMode	boolean	1..1	Returns true if there is a mismatch.

10.3.6 METHOD: DUPLEXEVENT

DuplexRequest requests to change the duplex mode on an Individual Call. DuplexEvent returns the duplex mode on an Individual Call.

The duplex mode is applied when making outgoing calls and when answering incoming calls. Incoming full duplex calls are converted to half duplex if answered in half duplex mode. The DF-Client must still use TransmitRequest to obtain a speech item in a full duplex call.

Input (Literal)

The input of this method is the document element DuplexEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
state	DuplexMode	1..1	Duplex mode: HalfDuplex, FullDuplex

10.3.7 METHOD: ENCRYPTIONEVENT

EncryptionRequest requests to change the encryption mode on an Individual Call. EncryptionEvent returns the encryption mode on an Individual Call.

Input (Literal)

The input of this method is the document element EncryptionEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
state	EncryptionState	1..1	EncryptionState. Disabled indicates that DF-Client requests to change the encryption mode are not supported.

10.3.8 Method: IndividualCallEvent

IndividualCallEvent reports the call state and other party details. The subscriber mnemonic data may be empty due to TCS delays in retrieving data.

Input (Literal)

The input of this method is the document element IndividualCallEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	IndividualCallEventData	1..1	The individualCall data.

Type IndividualCallEventData:

Component	Type	Occurs	Description
CallId	int	1..1	The Call ID.
DuplexState	DuplexMode	1..1	Duplex state
EntityType	EntityType	1..1	EntityType.

Component	Type	Occurs	Description
IndividualCallState	IndividualCallState (Enum): CallStateNotDefined CallStateProceeding CallStateAlerting CallStateQueuing CallStateConnected CallStateOnHold CallStateDisconnected CallStateTransferred CallStateIncomingRinging CallStateImminentDisconnect CallStateWaiting CallStateContinue	1..1	State of IndividualCall.
IsAmbianceListening	boolean	1..1	Value indicating whether ambiance listening True or false).
IsCallEncryptive	boolean	1..1	Value indicating whether call is encryptive (true or false).
IsDispatcherCall	boolean	1..1	Value indicating whether a dispatcher call (true or false).
OtherPartyId	TetraSubscriberData	1..1	Subscriber data for the other party on the call.
OtherPartyNumber	String	1..1	OtherPartyNumber.
PriorityType	PriorityType	1..1	Priority type.
ReleaseReason	TetraReleaseReasonData	1..1	Release reason.
TimeStamp	dateTime	1..1	DateTime.
UniqueId	int	1..1	Our own ISSI.

10.3.9 METHOD: LINKSTATEEVENT

LinkStateEvent indicates the link state of the Individual Call.

Input (Literal)

The input of this method is the document element LinkStateEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	LinkStateData	1..1	Link state data.

10.3.10 METHOD: PREEMPTIONEVENT

PreemptionRequest enables or disables DF-Client pre-emption within the call. The DF-Client transmits with a higher priority when pre-emption is enabled. PreemptionEvent indicates the DF-Client pre-emption state.

Input (Literal)

The input of this method is the document element PreemptionEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
state	PreemptionState	1..1	The preemption state. The disabled state indicates DF-Client requests to change the state are not supported.

10.3.11 Method: ResourceLabelEvent

ResourceLabelEvent identifies the TETRA Individual Call unique ID and label.

Input (Literal)

The input of this method is the document element ResourceLabelEvent having the structure defined by the following table.

Element	Type	Occurs	Description
uniqueId	int	1..1	Our own ISSI.
label	string	1..1	Individual Call label.

10.3.12 Method: TransmitEvent

TransmitRequest is used when the DF-Client wishes to transmit on an Individual Call. TransmitEvent indicates a DF-Client talking on an Individual Call.

Input (Literal)

The input of this method is the document element TransmitEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	TransmitEventData	1..1	The transmit event data.

10.3.13 Method: LeadingDigitsEvent

LeadingDigitsEvent indicates the mapping of leading digits to TETRA numbering plan as provisioned. This information can be used by the DF-Client to determine the numbering plan from a string of input digits.

Input (Literal)

The input of this method is the document element LeadingDigitsEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	ArrayOfLeadingDigitData	1..1	The list of leading digit data: Digit and NumberingPlan

10.3.14 Method: SpecialNumbersEvent

SpecialNumbersEvent indicates the special numbers as provisioned. This information can be used by the DF-Client to determine the numbering plan from a string of input digits that exactly match a provisioned special number.

Input (Literal)

The input of this method is the document element SpecialNumbersEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	ArrayOfSpecialNumbersData	1..1	List of special numbers data: Number and NumberingPlan

11 TETRASDSSERVICE & EVENT WEB SERVICE

11.1 USE CASES

The TetraSds Service and Events contracts are simple contracts to use. The complexity resides in the understanding of the capabilities that the TETRA network can offer in term of short data services. The use cases included are demonstrating the initialization and also the sending and receiving of text messages.

In all the cases, the prerequisite is to have a valid session and a resource properly allocated to proceed with those services.

11.1.1 TETRA SDS INITIALIZATION

This use-case is tking place when a DF-Client set up resources to operate with the Short Data Services capabilities. It is mainly providing information to enable the DF-Client to properly structure the information.

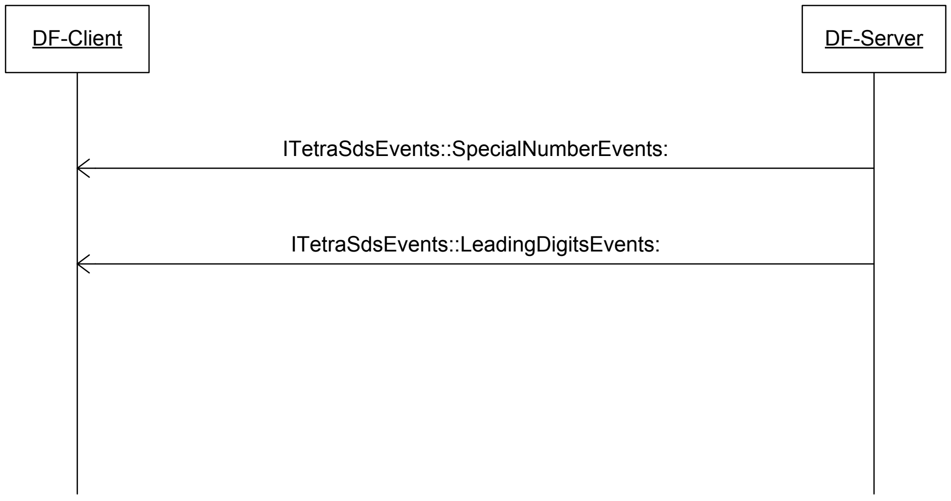

DF-Client
DF-Server
ITetraSdsEvents::SpecialNumberEvents:
ITetraSdsEvents::LeadingDigitsEvents:

Figure 15 - SDS Initialization

11.1.2 Sending SDS

The use-case to send an SDS text message is a simple request and event message like presenting in the following figure.

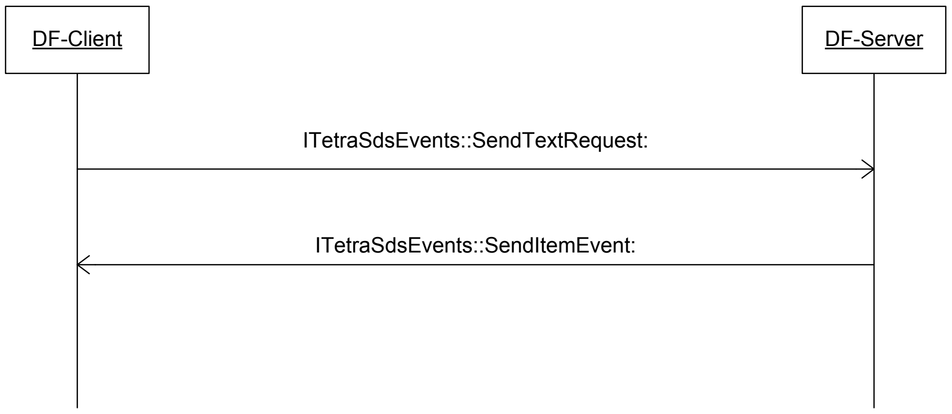

DF-Client
DF-Server
ITetraSdsEvents::SendTextRequest:
ITetraSdsEvents::SendItemEvent:

Figure 16 - Sending SDS

11.1.3 Receiving SDS

Receiving an SDS message is a simple event received from the DF-Server.

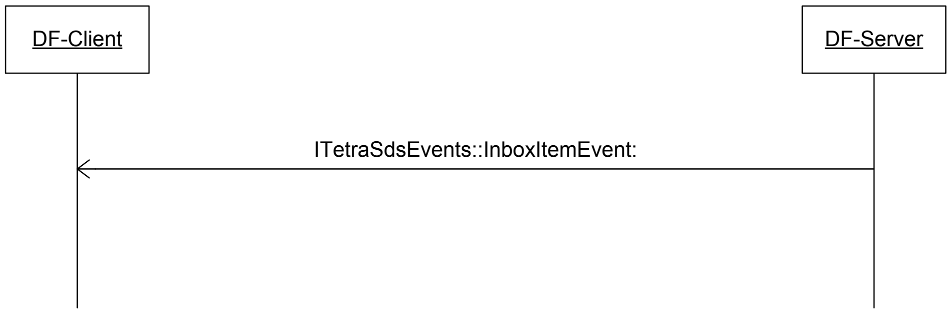


Figure 17 - Receiving SDS

11.2 TetraSdsService Web Service

11.2.1 Method: DeleteItemRequest

DF-Client request to delete an item. For a Sent item, SentItemEvent is returned on success, otherwise a console error log is generated. For an Inbox item, InboxItemEvent is returned on success, otherwise a console error log is generated.

Input (Literal)

The input of this method is the document element DeleteItemRequest having the structure defined by the following table.

Element	Type	Occurs	Description
messageId	int	1..1	The message identifier.

11.2.2 Method: ReadItemRequest

DF-Client request to mark an item as read or unread. For a Sent item, SentItemEvent is returned on success, otherwise a console error log is generated. For an Inbox item, InboxItemEvent is returned on success, otherwise a console error log is generated.

Input (Literal)

The input of this method is the document element ReadItemRequest having the structure defined by the following table.

Element	Type	Occurs	Description
messageId	int	1..1	The message identifier.
isRead	boolean	1..1	Set to true if the message has been read; set to false if unread.

11.2.3 METHOD: SENDCALLBACKREQUEST

SendCallbackRequest sends a callback request status message to a list of TETRA subscribers. SendItemEvent is always received regardless of the success or failure to send. An InboxItemEvent with an undeliverable message is received on failure to send.

Input (Literal)

The input of this method is the document element SendCallbackRequest having the structure defined by the following table.

Element	Type	Occurs	Description
dstList	ArrayOfTetraSubscriberData	1..1	List of destination Subscriber Identifiers.
callbackType	CallbackType	1..1	The callback type: CallbackTypeNormal, CallbackTypeUrgent, CallbackTypeEmergency CallbackTypeDispatcher is not valid for SendCallbackRequest.
isDeliveryReport	boolean	1..1	Set to true for a delivery report indicating whether or not delivery was successful. An InboxItemEvent is received with the delivery report message.
clientMessageId	int	1..1	The server returns this message identifier on all messages exchanged with the DF-Client.

11.2.4 Method: SendStatusRequest

SendStatusRequest sends a status message to a list of TETRA subscribers. SendItemEvent is always received regardless of the success or failure to send. An InboxItemEvent with an undeliverable message is received on failure to send.

SendStatusRequest is used to create a new status message or reply to an existing message with a status message. When replying to an existing message set action=ActionReply and include the messageId of the existing message.

Input (Literal)

The input of this method is the document element SendStatusRequest having the structure defined by the following table.

Element	Type	Occurs	Description
dstNumber	ArrayOfstring	1..1	List of destination numbers. All numbering plans can be used (TETRA, FSSN, MSISDN, External). The numbering plan is determined by either the leading digit of the dstNumber or by the dstNumber being an exact match to a provisioned special number. TETRA Numbering plan is using a "-" as a leading digit and also between the 3 components of the id. For example: "-XXX-YYY-ZZZ"
status	int	1..1	The status value as provisioned.
isDeliveryReport	boolean	1..1	Set to true for a delivery report indicating whether or not delivery was successful. An InboxItemEvent is received with the delivery report message.
action	TetraSdsAction	1..1	The Tetra SDS action. New or reply actions are permitted. The forward action is not permitted.

Element	Type	Occurs	Description
messageId	int	1..1	The message identifier must reference an existing status or text message when the reply action is specified. The message identifier is not used when the new action is specified.
clientMessageId	int	1..1	The server returns this message identifier on all messages exchanged with the DF-Client.

11.2.5 METHOD: SENDTEXTREQUEST

SendTextRequest sends a text message to a list of TETRA subscribers. SendItemEvent is always received regardless of the success or failure to send. An InboxItemEvent with an undeliverable message is received on failure to send.

SendTextRequest is used to create a new text message, reply to, or forward an existing message with a text message. When replying to or forwarding an existing message set action=ActionReply or ActionForward and include the messageId of the existing message.

Input (Literal)

The input of this method is the document element SendTextRequest having the structure defined by the following table.

Element	Type	Occurs	Description
dstNumber	ArrayOfstring	1..1	List of destination numbers. All numbering plans can be used (TETRA, FSSN, MSISDN, External). The numbering plan is determined by either the leading digit of the dstNumber or by the dstNumber being an exact match to a provisioned special number. TETRA Numbering plan is using a "-" as a leading digit and also between the 3 components of the id. For example: "-XXX-YYY-ZZZ"

Element	Type	Occurs	Description
encoding	TetraSdsEncoding (Enum): EncodingGsm EncodingLatin1 EncodingLatin9 EncodingUnicode	1..1	The encoding.
messageText	string	1..1	The message text. Each message has a maximum number of characters that it can contain. The number of characters allowed depends on the character encoding used and whether or not the message is encrypted. The DF-Client must enforce the maximum number of characters contained in a single message. Message Restrictions: • **16-bit UCS2 encoding: -** • **8-bit Latin 1 or 8-bit Latin 9 encoding: -** **7-bit GSM encoding: -** The SendTextRequest can exceed these maximums if message concatenation is used. Message concatenation allows a text message of up to 3 separate 133 character portions (for a total maximum of 399 characters). Message concatenation not permitted for all scenarios and the DF-Client is expected to enforce the restrictions on whether or not message concatenation is allowed. All of the following must be true for message concatenation to be allowed:
isDeliveryReport	boolean	1..1	Set to true for a delivery report indicating whether or not delivery was successful. An InboxItemEvent is received with the delivery report message.

Element	Type	Occurs	Description
isEncrypted	boolean	1..1	Set to true if encrypted.
isFlash	boolean	1..1	Set to true if a flash message. Flash messages are instantly displayed on the radio unit screen.
isUseSms	boolean	1..1	Set to true if using the Short Message Server to store and forward the message.
action	TetraSdsAction	1..1	The TETRA SDS action. New, reply, and forward actions are permitted.
messageId	int	1..1	The message identifier must reference an existing status or text message when the reply or forward action is specified. The message identifier is not used when the new action is specified.
clientMessageId	int	1..1	The server returns this message identifier on all messages exchanged with the DF-Client.

11.2.6 METHOD: SENDUNITALERTREQUEST

SendUnitAlertRequest sends a unit alert status message to a list of TETRA subscribers. SendItemEvent is always received regardless of the success or failure to send. An InboxItemEvent with an undeliverable message is received on failure to send.

Input (Literal)

The input of this method is the document element SendUnitAlertRequest having the structure defined by the following table.

Element	Type	Occurs	Description
dstList	ArrayOfTetraSubscriberData	1..1	List of destination Subscriber Identifiers.
isDeliveryReport	boolean	1..1	Set to true for a delivery report indicating whether or not delivery was successful. An InboxItemEvent is received with the delivery report message.
clientMessageId	int	1..1	The server returns this message identifier on all messages exchanged with the DF-Client.

11.3 TetraSdsEventsService Web Service

11.3.1 Method: InboxItemEvent

InboxItemEvent indicates a received status or text message or delivery report. An InboxItemEvent with an undeliverable message is also received on failure to send a status or text message.

The MessageId parameter is generated whenever a new status or text message or delivery report is received. The MessageId is unique accross all messages sent or received. The DF-Client is expected to use the MessageId when replying to, forwarding, or deleting an existing message.

The LastAction parameter is updated whenever an existing received message is replied to, forwarded, or deleted.

The IsRead parameter is updated whenever an existing received message is changed by a ReadItemRequest. Received status or text messages are marked as unread by default.

Input (Literal)

The input of this method is the document element InboxItemEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	InboxItemEventData	1..1	The Inbox Item Event data.

Type InboxItemEventData:

Component	Type	Occurs	Description
From	TetraSubscriberData	1..1	Originator Subscriber data.
FromNumber	string	1..1	Originator Number.
InboxItemType	ItemType	1..1	Inbox item type.
IsEncrypted	boolean	1..1	Value indicating whether encrypted (true or false).
IsFlash	boolean	1..1	Value indicating whether flash (true or false).
IsNotification	boolean	1..1	Value indicating whether notification (true or false).
IsRead	boolean	1..1	Value indicating whether read (true or false).

Component	Type	Occurs	Description
LastAction	LastAction	1..1	LastAction.
MessageId	int	1..1	MessageId.
ClientMessageId	int	1..1	Server side message Id
MessageText	string	1..1	MessageText.
NotificationCode	TetraSdsNotificationCode (Enum): NoNotificationCode DeliveryFailure DeliverySuccess DeliveryTimeout MaxLengthExceeded UnableToEncrypt UnableToForward UnableToReply UnableToSend UnknownMessageId UnknownCode InvalidNumber UnableToDecrypt	1..1	NotificationCode.
TimeStamp	dateTime	1..1	DateTime.
To	TetraSubscriberData	1..1	To subscriber data.
ToNumber	string	1..1	To Number.
NotificationReason	TetraReleaseReason	1..1	Cause of the notification.
NotificationText	string	1..1	Text describing the notification code.

11.3.2 Method: SdsFeaturesEvent

SdsFeaturesEvent identifies the features supported by the TETRA short data service.

Input (Literal)

The input of this method is the document element SdsFeaturesEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	TetraSdsFeaturesData	1..1	The supported features.

Type TetraSdsFeaturesData

Component	Type	Occurs	Description
StatusValues	ArrayOfStatusValueData	1..1	List of status values.
SupportedFeatures	ArrayOfTetraSdsFeatureType	1..1	List of features supported by the short data service.

Type ArrayOfStatusValueData:

Component	Type	Occurs	Description
StatusValueData	StatusValueData	1..*	

Type StatusValueData:

Component	Type	Occurs	Description
Status	unsignedShort	1..1	Status.
Value	string	1..1	Value.

Type ArrayOfTetraSdsFeatureType:

Component	Type	Occurs	Description
TetraSdsFeatureType	TetraSdsFeatureType (Enum): TetraSdsFeatureSms TetraSdsFeatureLip	1..*	Feature type

11.3.3 METHOD: SENDITEMEVENT

SendItemEvent indicates a sent status or text message. SendItemEvent is received upon a successful call to SendCallbackRequest, SendUnitAlertRequest, SendStatusRequest, SendTextRequest, DeleteItemRequest, ReadItemRequest.

The MessageId parameter is generated whenever a new status or text message is sent. The MessageId is unique accross all messages sent or received. The DF-Client is expected to use the MessageId when repling to, forwarding, or deleting an existing message.

The LastAction parameter is updated whenever an existing sent message is replied to, forwarded, or deleted.

The IsRead parameter is updated whenever an existing sent message is changed by a ReadItemRequest. Sent status or text messages are marked as read by default.

Input (Literal)

The input of this method is the document element SendItemEvent having the structure defined by the following table.

Element	**Type**	**Occurs**	**Description**
data	SendItemEventData	1..1	The Send Item Event Data.

Type SendItemEventData:

Component	**Type**	**Occurs**	**Description**
DstNumberList	ArrayOfstring	1..1	Send To destination number list.
IsDeliveryReport	boolean	1..1	Value indicating whether a delivery report (true or false).
IsEncrypted	boolean	1..1	Value indicating whether encrypted (true or false)
IsRead	boolean	1..1	Value indicating whether read (true or false).
ItemEventType	ItemType	1..1	ItemType.
LastAction	LastAction	1..1	LastAction.
MessageId	int	1..1	MessageId.

Component	**Type**	**Occurs**	**Description**
ClientMessageId	int	1..1	The server to DF-Client message Id.
MessageText	string	1..1	MessageText.
SendToList	ArrayOfTetraSubscriberData	1..1	Send To subscriber data list.
From	TetraSubscriberData	1..1	From field
TimeStamp	dateTime	1..1	DateTime.

11.3.4 Method: LeadingDigitsEvent

LeadingDigitsEvent indicates the mapping of leading digits to TETRA numbering plan as provisioned. This information can be used by the DF-Client to determine the numbering plan from a string of input digits.

Input (Literal)

The input of this method is the document element LeadingDigitsEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	ArrayOfLeadingDigitData	1..1	The list of leading digit data: Digit and NumberingPlan

11.3.5 Method: SpecialNumbersEvent

SpecialNumbersEvent indicates the special numbers as provisioned. This information can be used by the DF-Client to determine the numbering plan from a string of input digits that exactly match a provisioned special number.

Input (Literal)

The input of this method is the document element SpecialNumbersEvent having the structure defined by the following table.

Element	Type	Occurs	Description
data	ArrayOfSpecialNumbersData	1..1	List of special numbers data: Number and NumberingPlan

12 TetraUnitTracking & Event Service Web Service

12.1 Uses Cases

The TetraUnitTracking request and events contract is a combination of two services, one being the tracking of the units and the other one being the request and reception of unit information. Each of those two capabilities is presented by a use-case.

In all the cases, the prerequisite is to have a valid session and a resource properly allocated to proceed with those services.

12.1.1 TRACKING UNITS

Tracking a unit is a simple request an event process. In the following use-case, there is also added in the middle, the operation of forcing a unit to release a call.

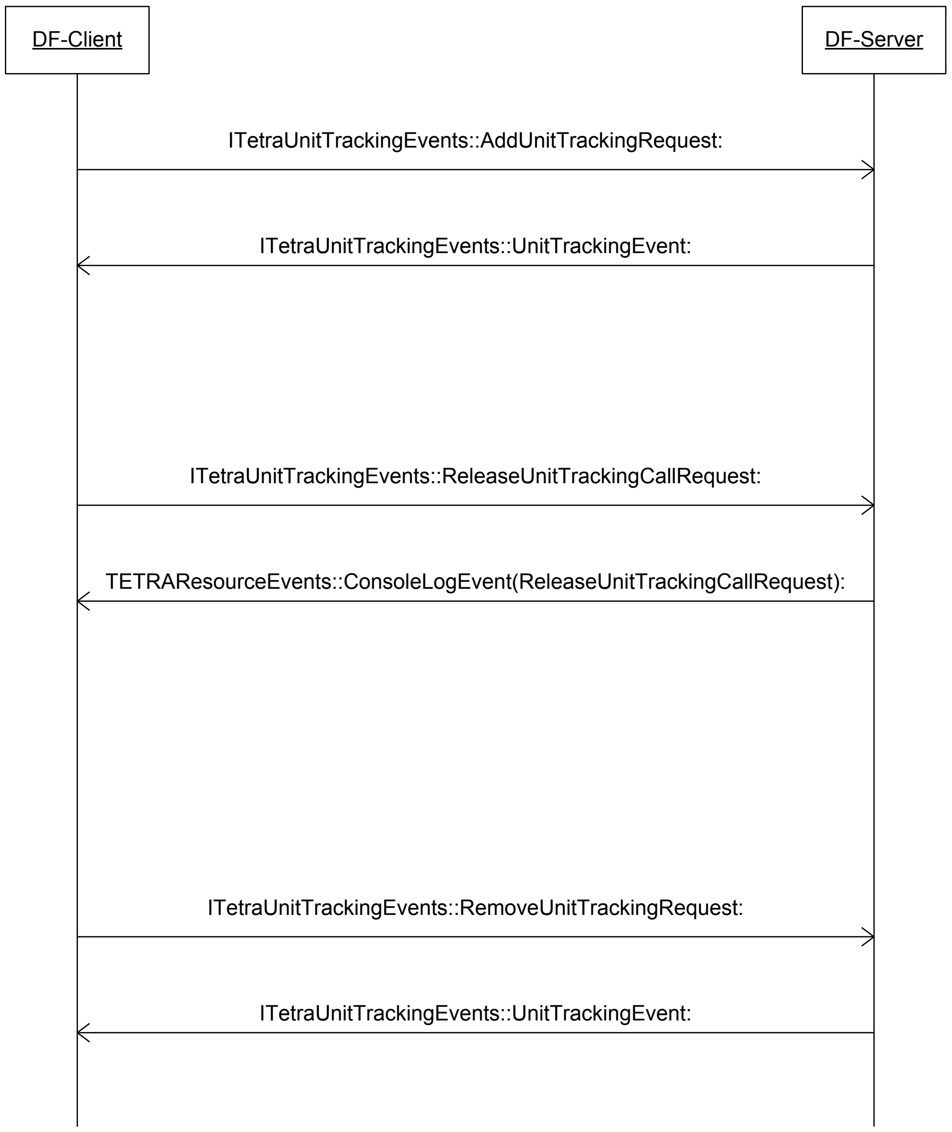

DF-Client
DF-Server
ITetraUnitTrackingEvents::AddUnitTrackingRequest:
ITetraUnitTrackingEvents::UnitTrackingEvent:
ITetraUnitTrackingEvents::ReleaseUnitTrackingCallRequest:
TETRAResourceEvents::ConsoleLogEvent(ReleaseUnitTrackingCallRequest):
ITetraUnitTrackingEvents::RemoveUnitTrackingRequest:
ITetraUnitTrackingEvents::UnitTrackingEvent:

Figure 18 - Tracking Units

12.1.2 Requesting Unit List

Requesting a unit list is a simple request and event process like demonstrated in the following figure.

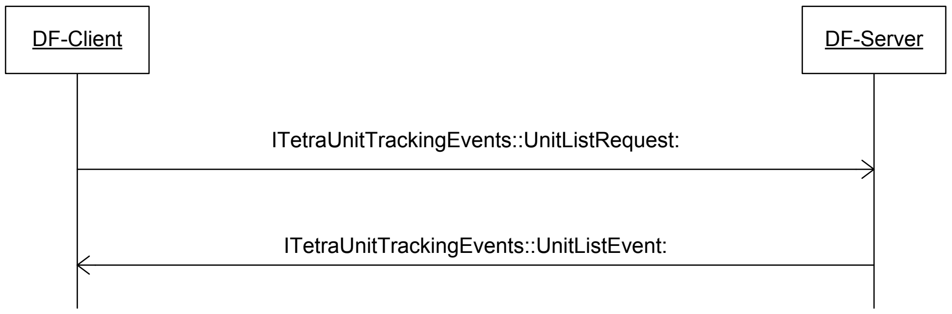

DF-Client
DF-Server
ITetraUnitTrackingEvents::UnitListRequest:
ITetraUnitTrackingEvents::UnitListEvent:

Figure 19 - Requesting Unit List

12.2 TetraUnitTracking Service Web Service

12.2.1 Method: AddUnitTrackingRequest

AddUnitTrackingRequest starts tracking of the TETRA units. UnitTrackingEvent contains the data for a tracked unit. ConsoleLogEvent is received if a unit cannot be added to the tracking list.

Input (Literal)

The input of this method is the document element AddUnitTrackingRequest having the structure defined by the following table.

Element	Type	Occurs	Description
units	ArrayOfTetraSubscriberData	1..1	List of TETRA subscribers to be tracked.

12.2.2 Method: ReleaseUnitTrackingCallRequest

ReleaseUnitTrackingCallRequest releases the current individual TETRA call of the tracked unit. A ConsoleLogEvent is returned to indicate success or failure of the request.

Input (Literal)

The input of this method is the document element ReleaseUnitTrackingCallRequest having the structure defined by the following table.

Element	Type	Occurs	Description
otherPartyId	TetraSubscriberData	1..1	TETRA subscriber information for the other party on the call.

12.2.3 METHOD: REMOVEUNITTRACKINGREQUEST

RemoveUnitTrackingRequest stops tracking of the TETRA units. UnitTrackingEvent is received with IsUnitPresent=false when a unit is removed from the tracking list. ConsoleLogEvent is received if a unit cannot be removed from the tracking list.

Input (Literal)

The input of this method is the document element RemoveUnitTrackingRequest having the structure defined by the following table.

Element	Type	Occurs	Description
units	ArrayOfTetraSubscriberData	1..1	List of TETRA subscribers to stop tracking.

12.2.4 METHOD: UNITLISTREQUEST

UnitListRequest requests a portion of the TETRA organization block hierarchy. Organization blocks are used to organize users into a hierarchical structure and to define communication and management rights for that particular organization. An organization block may contain other organization blocks in addition to users. A TETRA network supports up to six levels of nested organization blocks.

Input (Literal)

The input of this method is the document element UnitListRequest having the structure defined by the following table.

Element	Type	Occurs	Description
parentOrgBlockId	string	1..1	Parent organzation block identifier. The UnitListEvent returns the organization blocks and radio units that are contained within the specified parent organization block. The top level organization block is requested by specifying the empty string as the parentOrgBlockId. The UnitListEvent is asynchronously received whenever the organization block hierarchy changes within the TETRA network.

12.3 TetraUnitTrackingEventsService Web Service

12.3.1 Method: UnitListEvent

The Unit List is returned from the TCS based on the organization hierarchy. Unit List data is only returned when necessary (that is, not all data from every organization is returned at once). The DF-Client is expected to recursively request the data from each organization level only as needed and store the data in a tree hierarchy.

Input (Literal)

The input of this method is the document element UnitListEvent having the structure defined by the following table.

Element	Type	Occurs	Description
parentOrgBlockId	string	1..1	The parent organization block identifier.
orgBlockData	ArrayOfUnitListOrgBlockData	1..1	A list of organization block data that belongs to the parent organization block.
unitData	ArrayOfUnitListUnitData	1..1	A list of unit data that belongs to the parent organization block.

Type ArrayOfUnitListOrgBlockData

Component	Type	Occurs	Description
UnitListOrgBlockData	UnitListOrgBlockData	1..*	

Type UnitListOrgBlockData

Component	Type	Occurs	Description

Component	Type	Occurs	Description
IsDeleted	boolean	1..1	Value indicating whether the organization block is deleted.
OrgBlockId	string	1..1	Organization block identifier.
OrgBlockMnemonic	string	1..1	Organization block mnemonic.

Type ArrayOfUnitListUnitData:

Component	Type	Occurs	Description
UnitListUnitData	UnitListUnitData	1..*	

Type UnitListUnitData

Component	Type	Occurs	Description
IsDeleted	boolean	1..1	Value indicating whether the radio unit is deleted.
UnitID	TetraSubscriberData	1..1	Subscriber data for the radio unit.

12.3.2 Method: UnitTrackingEvent

UnitTrackingEvent contains the data for the tracked unit and is received whenever the data for the tracked unit changes.

Input (Literal)

The input of this method is the document element UnitTrackingEvent having the structure defined by the following table.

Element	Type	Occurs	Description
unitTrackingData	UnitTrackingData	1..1	The unit tracking data.

Type UnitTrackingData:

Component	Type	Occurs	Description
CallStatus	TrackingCallStatus (Enum): CallStatusUnknown CallStatusNone CallStatusDmo CallStatusIc CallStatusScanningOn	1..1	TrackingCallStatus of the tracked unit.
DwStatus	TrackingDualWatchStatus (Enum): DwStatusUnknown DwStatusNone DwStatusIdleDualWatch DwStatusFullDualWatch DwStatusTxInhibit	1..1	TrackingDualWatchStatus of the tracked unit.
ExchangeId	int	1..1	The tracked unit ExchangeId.
GroupMnemonic	string	1..1	The selected group mnemonic.

Component	Type	Occurs	Description
GroupSsi	int	1..1	The selected group Short Subscriber Identifier (SSI).
IsEmergencyState	boolean	1..1	Value indicating whether the tracked unit is in emergency state (true or false).
IsRegistered	boolean	1..1	Value indicating whether the tracked unit is registered in the TETRA system (true or false).
IsSelectedAndWithinGroupArea	boolean	1..1	Value indicating whether the tracked unit has selected a group and is within the selected group area (true or false).
IsUnitPresent	boolean	1..1	Value indicating whether a unit is present (true or false). False indicates the tracked unit was deleted in the TETRA system.
LastActiveTime	dateTime	1..1	Time of the last registration from the tracked unit.
OtherPartyId	TetraSubscriberData	1..1	Subscriber data for the other party on an individual call with the tracked unit.
StatusIndicator	int	1..1	Last status received from the tracked unit.
StatusIndicatorTime	dateTime	1..1	Time of the last status received from the tracked unit.
UnitId	TetraSubscriberData	1..1	Subscriber data for the tracked unit.

Technical Description of Digitalfunkstecker V0.2

Bundesverband Professioneller Mobilfunk (PMeV)

13 Unplanned Improvements

13.1 SDP Negotiation

Before including the capability to negotiate SDP parameters between the DF-Client and the DF-Server, we need to verify how and where to include SDP requests from the clients. At this point, the server is unilaterally imposing the SDP parameters. For now the SDP parameters are dictated by the server side.

13.2 BIPs Version

Currently supported version of the BIPs in this document is October 2011.

13.3 Aggregated Transmit Request

The idea is to support TransmitRequests for group and individual calls containing all the possible transmit parameters. Before including those capabilities, we need to determine which type of feedback (event) are to be sent following those type of transmit request in case of success but also in case of failure (of the transmission or of the parameters setting).

Version: Draft 1.0: 13.8.2010

History:

Version		
0.2	11.2.2007	Initial Version
0.3	30.10.2008	Deletion of “B” bad frame indicator bit and mad control bit value 4 out of it. Some other cosmetic changes.
0.4	30.09.2009	Fix in drawing with 2 audio frames of chapter 8 – 4 bytes were missing in second sub-frame.
0.5	5.11.2009	Introduction of the “Failed Crypto operation indication” bit. Cosmetic move in the order of the description of the fields so that the order of the description better matches the occurrence in the drawing. References fixed. Missing D() bit description added.
0.6	15.7.2010	Formal cleanup in order to provide the document to Frequentis external parties.
1.0	13.8.2010	Clearance of “track changes”. Fixing of RFC references. Final version for third parties.

This document is written from the formal point of view like an IETF draft so that in the future a person familiar with the tooling could create an IETF draft, which fullfills the IETF formal criteria (http://tools.ietf.org/inventory/author-tools.shtml).

Version 0.5 serves as a Frequentis RTP_SDP_IDD for customer projects with E2E encryption. All Frequentis implementations ignore the new “Failed Crypto operation indication” bit. The Frequentis DIVOS recorder records the “Failed Crypto operation indication” bit. The Frequentis DIVOS play-out console ignores any “C” bit set to 1 during play-out.

The document is written based on following instructions:

draft-ietf-avt-rtp-howto-02 - How to Write an RTP Payload Format.mht .

This winword document contains items of the template as “Hidden Text.”, which is currently not filled out yes so that the reader can see what is left to be done to polish the document so that it might get sent to the IETF.

1. Title

Real-Time Transport Protocol (RTP) Payload Format for the TETRA Audio Codec

2. Front page boilerplate

This Memo contains the RTP payload and SDP definition for TETRA coded audio within Frequentis.

3. Abstract

This document specifies a Real-time Transport Protocol (RTP) payload format to be used for TETRA encoded speech signals. The payload format is designed to be able to interoperate with existing TETRA transport formats on non-IP networks. This version of the document does not specify a file format for transport of TETRA speech data in storage mode applications such as email as would be required by the IETF. A media type registration is included, specifying the use of the RTP payload format and the storage format.

4. Table of Content

Todo for an IETF tooling guru.

5. Introduction

This document specifies the payload format for packetization of TErrestial Trunked Radio (TETRA) encoded speech signals into the Real-time Transport Protocol (RTP) [1]. The payload format supports transmission of multiple channels, multiple frames per payload, robustness against packet loss, and interoperation with existing TETRA transport formats on non-IP networks, as described in Section 3.

The payload format itself is specified in Section 8.

6. Conventions, Definitions and Acronyms

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

The following acronyms are used in this document:

ETSI	- European Telecommunications Standards Institute
TETRA	- TErrestial Trunked Radio

The byte order used in this document is network byte order, i.e., the most significant byte first. The bit order is also the most significant bit first. This is presented in all figures as having the most significant bit leftmost on a line and with the lowest number. Some bit fields may wrap over multiple lines in which cases the bits on the first line are more significant than the bits on the next line.

7. Media Format Background

The TETRA codec is used as vocoder for TETRA systems. The TETRA codec is designed for compressing 30ms of audio speech data into 137 bits. The TETRA codec is designed in such a way that on the air interface two of theses 30ms samples are transported together (sub-block 1 and sub-block 2). The codec allows that data of the first 30ms voice frame can be stolen and used for other purposes, e.g. for the exchange of dynamically updated key-material in end-to-end encrypted voice sessions. For E1 lines there are two optional formats defined [3], the first format is called FSTE (First Speech Transport Encoding Format), the other format is called OSTE (Optimised Speech Transport Encoding Format). These two formats defer mainly insofar that the OSTE format transports an additional 5 bit frame number, which provides timing information from the air interface to the receiving side in order to save the need for buffering due to different transports speed on air and in 64 kbit/s circuit switched networks. The RTP payload format is defined such that the value of this frame number can be transported.

8. Payload format

The RTP payload format is designed in such a way that it can carry the information needed to map the FSTE and OSTE format from [1]. The RTP format is defined such that both of the independent sub-blocks can be transferred separately or together within one RTP frame.

The payload format is chosen such that the TETRA data bits are octet aligned.

F bit: Frame type
1: The following frame contains a first block of two sub-blocks
0: The following frame contains a separated sub-block. A sub-block marked as such could either be a second sub-block, or an independent block, which does not have a relation with any first block.

S: Spare bit 0

C bit: Failed Crypto operation indication: This bit may be set to “1” if an encryption or a decryption operation could not be performed successfully for the specific half-block. Consequently, the encryption status of the half-block audio data is unknown. If a receiver decides to forward the TETRA audio data to OSTE or FSTE or to directly hand over the TETRA audio data to a TETRA audio decoder, the contained audio might be scrambled – depending if the audio originally was generated as a plain-override half-block or as an encrypted half-block.

CTRL: Control bit(3 bits)
The control bit contains meta information about the block.

000	Block normal
001	Block U stolen
010	Block C stolen
011	O&M ISI Block
100	Bad block

D() bits: Data bits (i.e. TETRA ACELP coded speech bits) according to Table 4 of [3].

FRAME_NR: FN (5 bits): contains an uplink frame number as defined in table 8 of [1].
If no frame number is available the FRAME_NR value SHALL be set to 00000.

Payload definition:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |F|  S  |C|CTRL |D(1)                                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           D(137)|S  |FRAME_NR |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following example shows how a first and a consecutive 30 ms frame
is combined into a single RTP packet. Note: This example shows of usage of
OSTE mapping as used by Frequentis. Other combinations might be supported in the future.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1|  S  |C|CTRL |D(1)                                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           D(137)|S  |0 0 0 0 0|0| S     |CTRL |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           D(137)|S  |0 0 0 0 0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

11.1. Media Type Definition

The media type for the TETRA codec is expected to be allocated from the IETF tree once
this draft turns into an RFC. This media type registration covers both real-time
transfer via RTP and non-real-time transfers via stored files.
 [Here the media type registration template from RFC 4288 is placed
   and filled out. This template is provided with some common RTP
   boilerplate.]

Media Type name: audio

Media Subtype name: TETRA

Required parameters: none

Optional parameters:

These parameters apply to RTP transfer only.

maxptime: The maximum amount of media which can be encapsulated
              in a payload packet, expressed as time in milliseconds.
              The time is calculated as the sum of the time that the
              media present in the packet represents. The time SHOULD
              be an integer multiple of the frame size. If this

               parameter is not present, the sender MAY encapsulate any
               number of speech frames into one RTP packet.

ptime: see RFC 4566 [4].

channels: The number of audio channels. The possible values
               (1-6) and their respective channel order is specified in
               Section 4.1 in [5]. If omitted, it has the default
               value of 1.

frqind: As long as there is no official RTP payload definition from IETF this Frequentis proprietary parameter is marked with the only possible value 1. It marks the session to be established according to this specification. If frqind is present, the payload type for the session MUST be of value 99.

Encoding considerations:

The Audio data is binary data, and must be encoded for non-binary transport; the Base64 encoding is suitable for email. When used in RTP context the data is framed as defined in [6].

Security considerations:

See Section 7 of RFC 4867.

Interoperability considerations:

Published specification:

Applications that use this media type:

This media type is used in applications needing transport or storage of encoded voice. Some examples include; Voice over IP, streaming media, voice messaging, and voice recording on recording systems.

Person & email address to contact for further information:

Stefan Wenk stefan.wenk@frequentis.com

Intended usage: (One of COMMON, LIMITED USE or OBSOLETE.)

Restrictions on usage:

When this media type is used in the context of transfer over RTP, the RTP payload format specified in Section 8 SHALL be used. In all other contexts, the file format defined in Section 5 SHALL be used.

This media type depends on RTP framing, and hence is only defined for transfer via RTP [RFC3550]. Transport within other framing protocols is not defined at this time.

11.2. Mapping to SDP

The information carried in the media type specification has a specific mapping to fields in the Session Description Protocol (SDP)[4], which is commonly used to describe RTP sessions. When SDP is used to specify sessions employing the TETRA codec, the mapping is as follows:

```
Media Type name:     audio
Media subtype name:  TETRA
Required parameters: none
Optional parameters: none
```

Mapping MIME Parameters into SDP: The information carried in the MIME media type specification has a specific mapping to fields in the Session Description Protocol [RFC2327], which is commonly used to describe RTP sessions. When SDP is used to specify sessions employing the TETRA codec, the mapping is as follows:

- The MIME type ("audio") goes in SDP "m=" as the media name.
- The MIME subtype (payload format name) goes in SDP "a=rtpmap" as the encoding name. The RTP clock rate in "a=rtpmap" MUST be 8000.
- The parameters "ptime" and "maxptime" go in the SDP "a=ptime" and "a=maxptime" attributes, respectively.
- Any remaining parameters go in the SDP "a=fmtp" attribute by copying them directly from the media type parameter string as a semicolon-separated list of parameter=value pairs.

Here is an example SDP session of usage of TETRA as used by Frequentis:

```
    m=audio 49120 RTP/AVP 99
    a=rtpmap:99 TETRA/8000
    a=maxptime:60
    a=ptime:60
    a=fmtp:99 frqind=1
```

11.2.1. Offer/Answer Considerations

The following considerations apply when using SDP Offer-Answer procedures to negotiate the use of TETRA payload in RTP:

- In most cases, the parameters "maxptime" and "ptime" will not affect interoperability; however, the setting of the parameters can affect the performance of the application. The SDP offer-answer handling of the "ptime" parameter is described in RFC 3264 [13]. The "maxptime" parameter MUST be handled in the same way.

- Any unknown parameter in an offer SHALL be removed in the answer.

13. Securtiy Considerations

[See Section Section 7.1]

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [RFC3550] , and in any applicable RTP profile. The main security considerations for the RTP packet carrying the RTP payload format defined within this memo are confidentiality, integrity and source authenticity. Confidentiality is achieved by encryption of the RTP payload. Integrity of the RTP packets through suitable cryptographic integrity protection mechanism. Cryptographic system may also allow the authentication of the source of the payload. A suitable security mechanism for this RTP payload format should provide confidentiality, integrity protection and at least source authentication capable of determining if an RTP packet is from a member of the RTP session or not.

Note that the appropriate mechanism to provide security to RTP and payloads following this memo may vary. It is dependent on the application, the transport, and the signalling protocol employed. Therefore a single mechanism is not sufficient, although if suitable the usage of SRTP [RFC3711] is recommended. Other mechanism that may be used are IPsec [RFC4301] and TLS [RFC4346] (RTP over TCP), but also other alternatives may exist.

14. References

[1] Schulzrinne, H. and S. Casner, " RTP: A Transport Protocol for Real-Time Applications ", RFC 3550, July 2003.
[4] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 4566, July 2006.
[5] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 3548, July 2003.
[6] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

14.1. Normative References

[2] ETSI TS 100 392-3-6 V1.1.1 (2003-12) Part 2: TETRA Interworking at the Inter -System interface (ISI) Sub-part 6: Speech format implementation for circuit mode transmission
[3] ETSI TS 300 395-2 February 1998 Terrestrial Trunked Radio (TETRA); Speech codec for full-rate traffic channel; Part 2: TETRA codec

14.2. Informative References

15. Author Addresses

Stefan Wenk<stefan.wenk@frequentis.com>

16. IPR Notice

17. Copyright Notice