„Digitalradio-Gateway-Interface“ (DF-Stecker)

Technical Description

Publication of AK BOS-Leitstellen
(Proposed Baseline DR-GW-Interface V1.1)

09.10.2018

Document History

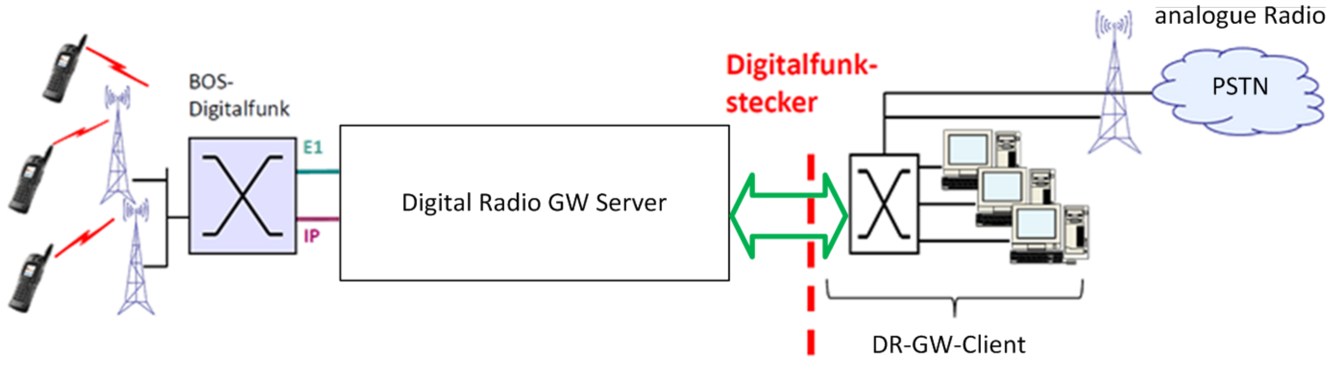
Version	Reference	Date
Version 1.0		15.09.2015
Version 1.0.1	Fixed Bug CR-64 CR-74: Use Case 5.12, Event: Group_RadioMemberEvent CR-65: 4.2.4 upgraded to 4.3, clarification CR-76: chapter 5.11 SDS Handling introduced CR-77: chapter 7.2 SIP Transport layer inserted CR-66: exchange chapter 9 completely	10.04.2017
Version 1.1.0	CR-70: hint to mandatory rtpmap for ACELP Codec CR-71: call events for individual calls CR-69: added re-INVITE as the only solution CR-67: added description for call events via SOAP CR-79: added new event Radio_GroupsEvent CR-82: link to github repository of xsd added CR-85: table added CR-87: heading corrected	09.10.2018

Table of Contents

Table of Figures

1 INTRODUCTION

This document describes the Digitalradio-Gateway-Interface (DF-Stecker) which is used to interface third-party applications (Digitalradio-Gateway-Clients) to the Digitalradio-Gateway (DR-GW).



This interface provides access to the TETRA System for third-party applications without being familiar with TETRA Connectivity Server API or Microsoft Component Object Model (COM) or Distributed Component Object Model (DCOM).
This interface is using SIP and RTP for audio signaling and audio real time transfer, and SOAP for data manipulation. One of the goals of this interface is to be network administrator friendly, so that configuration of network elements in different network environments would be easy and almost entirely automatic using the latest network technologies.

This document serves as a reference manual for third parties who develop applications to access the Digitalradio-Gateway.

This document is not a description of a DR-GW server product and it may not be considered as a document of vendor specific products and functionalities. For further information about Concentrator products, please refer to appropriate and specific product documentation.

How to use this document

This document comprises of following chapters:

- Chapter 1: *Introduction:* introduce the purpose of this document
- Chapter 2: *Glossary:* list of used abbreviations
- Chapter 3: *Applicable documents*
- Chapter 4: *Overview*
- Chapter 5: *Use Case Description*
- Chapter 6: *Security Aspects*
- Chapter 7: *Profile Standard for the use of SIP*
- Chapter 8: *Profile Standard for the use of SOAP*
- Chapter 9: *Request, Response and Event system*
- Chapter 10: *Interface Definition:* complete DR-GW-API description

Functional requirements described in this document may use the terms MUST, SHALL and may in the style of RFC 2119.

2 GLOSSARY

Term	Definition
ACELP	Algebraic Code Excited Linear Prediction
API	Application Programming Interface
BDBOS	Bundesanstalt für den Digitalfunk der Behörden und Organisationen mit Sicherheitsaufgaben
BOS	Behörden und Organisationen mit Sicherheitsaufgaben
BSI	Bundesamt für Sicherheit in der Informationstechnik
COM	Component Object Model
Cseq	Command Sequence
DCOM	Distributed Component Object Model
DR-GW-Client (DF-Stecker / Clientapplikation)	Client application of a Digital Radio-Gateway-Server
DR-GW (DF-Stecker Server)	Digital Radio-Gateway (from an API perspective)
E2E encrypted	End-to-end-encrypted (specific BSI)
FSTE	First Speech Transport Encoding Format
G.711	G.711 is an ITU-T standard for audio companding
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol
ISSI	Individual Short Subscriber ID
ITSI	Internet Engineering Task Force
LAN	Local Area Network

Term	Definition
LIP	Location Information Protocol
MSC	Message Sequence Chart
NAT	Network Address Translation
OSTE	Optimized Speech Transport Encoding Format
PCM	Pulse Code Modulation.
PCMA	Pulse Code Modulation A-Law format
PCMU	Pulse Code Modulation µ-Law format
PSTN	Public Switched Telephone Network
RTCP	Real-Time Control Protocol
PTT	Push To Talk
RTP	Real Time Transport Protocol
Rx	Receive/Receiving
SDP	Session Description Protocol
SDS	Short Data Services
SIP	Session Initiation Protocol
SIPS	Session Initiation Protocol Security
SOAP	Simple Object Access Protocol
SRTP	Secure Real-Time Transport 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
TLS	Transport Layer Security
Tx	Transmit/Transmission/Transmitting
UAC	User Account Control

Term	**Definition**
UDP	User Datagram Protocol
VoIP	Voice Over IP
WAN	Wide Area Network
WSDL	Web Service Definition Language
WWW	World Wide Web
XML	Extensible Markup Language
XSD	XML Schema Definition

3 APPLICABLE DOCUMENTS AND STANDARDS

3.1 REF. TO RFC OR ETSI

RFC	Content
RFC 2046	MIME: Multipurpose Internet Mail Extensions Part Two: Media Types
RFC 2617	HTTP: HTTP Authentication
RFC 2976	SIP: SIP INFO
RFC 3261	SIP: Session Initiation Protocol
RFC 3263	SIP: Locating SIP Servers
RFC 3550	RTP: Real-Time Transport Protocol
RFC 3326	SIP: The Reason Header Field for the Session Initiation Protocol
RFC 2976	SIP: SIP INFO Method
RFC 3262	SIP: Reliability of Provisional Responses in the Session Initiation Protocol
RFC 3311	SIP: UPDATE Method
RFC 3389	Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)
RFC 3515	SIP: REFER Method
RFC 4028	SIP: Session Timer
RFC 3265	SIP: Specific Event Notification
RFC 3911	SIP: Join Header
RFC 3891	SIP: Replaces Header
RFC 4566	SDP: Session Description Protocol
RFC 6141	Re-INVITE Handling in SIP
ETSI EN 300 392	ETSI EN 300 392-2 V2.3.2 (2001) Terrestrial Trunked Radio

3.2 Ref. Documents

Document	Content	Date
https://github.com/DF-Stecker-Expert-Panel/df-stecker-xsd/tree/1.1	git repository tag: 1.1	2018-08-14
DR_GW_payload_ref_vxx.pdf	Payload Format for voip transport	
EADS TETRA System Release 6.0–6.5	TCS API Description	1/2013

4 OVERVIEW

4.1 CONTRACTS OVERVIEW

Description

This document describes two different layers of services interacting with TETRA systems.

- application layer control protocol for all services concerning speech services. The referred protocol is described in RFC 3261 SIP. For different call types and scenarios refer to the use cases in this document. The login method is via SIP typical INVITE / REGISTER.
- web service description for the non-speech services (e.g. SDS) .

An understanding about the basics of BOS TETRA systems and the TCS API are not required (though helpful) for understanding this document. (Ref: TCS-Description - EADS TETRA System Release 6.0–6.5)

4.2 API WORKFLOW

Depending on the used service a DR-GW-Client establishes a connection either with a login, or with a SIP invite/register.

Login and invite / register logic refer respectively to the SIP and the SOAP part of the interface, these two parts work independently and can also be used separately.

Resource management is fully under control of the DR-GW and its performance depends on the manufacturer of the DR-GW.

4.2.1 DR-GW BUSINESS LOGIC

The DR-GW manages states for resource management (ISSIs, B channels and M channels). It is responsible for managing sessions, DR-GW- Clients and resource distribution; the way this is done is product specific.

4.2.2 DR-GW-CLIENT SESSION ESTABLISHMENT AND MANAGEMENT

The DR-GW-Client has to be provided with IP addresses in order to establish a connection and a SIP session with the DR-GW. At least, the DR-GW-Client shall be provisioned with a single IP address.

In group communication sessions are not necessarily disconnected by the DR-GW if the “real TETRA call” on the air is disconnected. Normally a session lasts until the DR-GW-Client does finish the session.

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

In case of a DR-GW-Client losing its session with the DR-GW, for example by having a failure at the HTTP or TCP/UDP layer or by not receiving Keep Alive Events regularly, the DF Client has to resend a login request to that IP address periodically in order to restore the session until it receives a valid answer. It is then the responsibility of the DR-GW-Client to restore the session by requesting resources and group monitoring.

4.2.3 B and M Channel

The concept of B and M Channels in the Tetra system and their handling is explained in the document "Hinweise und Handreichungen zur Schnittstelle Digitalfunkstecker (DF-Stecker) und ihrer Verwendung" published by PMeV.

B and M Channels are useful to support resource management by the DR-GW. Technically the DR-GW Client needs no further information about the channels and resource management in the TETRA network and this document will not make use of them. Yet knowing the concept may be helpful for implementors of control room applications.

4.3 API Versioning

There are 3 major interfaces making up the "DR-GW-Interface":

- RTP for Audio Transport
- SIP for Speech Communication Control
- SOAP for any other Services like data, group management, …

The explicit version announcement permits version management and version negotiation because of an incompatible version change.

4.3.1 RTP for Audio Transport Versioning

There is no other version information besides 2 Bit field in the RTP header defined in RFC 3550. At the time of writing this document, the version is set to 2.

4.3.2 SIP Interface Versioning

In order to give incompatible changes of the interface characteristics a chance in the future, DR-GW-Interface introduces a specific SIP Header field for versioning purposes. The present document SHALL be referred to as "radio.1.1" in the DR-GW-Version SIP Header Field as described in 7.8.1.

4.3.2.1 Prelude

The initial version of this document didn't specify how DR peers can indicate and agree upon different versions of the DR-Call protocol (subsequently called "SIP interface" or simply "interface") to use in a specific SIP dialog. For the purposes of indicating supported interface versions, a SIP header “DR-GW-Version” was introduced but its usage was not properly explained. Furthermore, there were uncertainties on how and when to use the “DR-GW-Version” header.

This chapter aims to rectify this.

4.3.2.2 SIP Interface Version

The initial interface version was “radio.01”. Future version **X.Y** will be “radio.X.Y”. A higher number indicates a newer interface.

If possible, DR peers should attempt to negotiate and use the newest interface version when communicating with each other in an attempt to enable features which might not be available in earlier versions of said interface and to improve user experience.

However, there may be valid reasons not to use a newer version of the interface, even when it would otherwise be available.

Before either DR-Client or DR-Gateway can place a new SIP call, a decision must be made in advance on what version of the interface is going to be used for that particular call. Depending on whether DR-Client chooses to receive incoming individual calls or not, it REGISTERs itself at DR-Gateway or not (see chapter “5.1 Resource allocation”).

There is both, common and specific behavior to DR-Client and DR-Gateway.

4.3.2.2.1 Common behavior:

- When DR-Peer sends an INVITE request, it MUST include exactly one “DR-GW-Version” header with exactly one parameter that indicates the interface version that is used in this call and to which the DR-GW-Call PDU, which is included in the (multipart) body, belongs to. If multiple versions were allowed to be listed, the meaning would be ambiguous leaving the receiver clueless as to what version the included DR-GW-Call PDU belongs to.
- When DR-Peer receives an INVITE request, it extracts the interface version from the “DR-GW-Version” header found therein and MUST check whether it supports that particular version of the interface. If the check fails, because the version listed is not supported, it MUST reject the call with 488 (Not acceptable here) and MAY include a “DR-GW-Version” header with a list of interface versions it is willing or able to support.
- When DR-Peer receives an INVITE request without a “DR-GW-Version” header but the message body contains content of type “application/DR-GW+xml”, it SHOULD assume version “radio.01” as being used for that call. This has become necessary because implementation experience showed that the “DR-GW-Version” header is not used consistently among initial implementations of this standard which mainly stems from the fact that the versioning mechanism wasn’t fully specified at the time.
- Once a SIP dialog is established, the interface version (from the initial INVITE) is fixed for that dialog and cannot change.

4.3.2.2.2 DR-Client behavior

DR-Client may query available interface versions from DR-Gateway by sending an OPTIONS request.

- When DR-Client sends an OPTIONS or REGISTER request to DR-Gateway, it MUST include a DR-GW-Version header, indicating the interface versions it is willing or able to support.
- When DR-Client receives a response to an OPTIONS or REGISTER request, it extracts the “DR-GW-Version” header from that response and associates it with the corresponding SIP endpoint. Subsequent calls by DR-Client to that endpoint MAY use any of the versions listed in the response.
- When DR-Client receives a 2XX response to an OPTIONS or REGISTER request that contains no “DR-GW-Version” header, it MAY assume that version “radio.01” is available. This has become necessary to support early implementations of DR-Gateways who don’t support the mechanism specified here.
- When DR-Client learns the supported interface versions from DR-Gateway, it SHOULD use the highest version available in subsequent calls to that DR-Gateway.
- When DR-Client doesn’t REGISTER itself to DR-Gateway but uses the OPTIONS request to find out which interface versions DR-Gateway supports, it SHOULD repeat the OPTIONS request periodically. This doesn’t serve as a heartbeat only, but it’s the only way to detect changes of the interface versions supported by DR-Gateway.

4.3.2.2.3 DR-Gateway behavior

DR-Gateway can act as a registrar or as a simple User Agent Server (UAS). When it acts as a registrar, it does so to allow DR-Clients to receive incoming individual calls in which case it too, must know in advance what interface versions the registered DR-Client supports in order to make a decision on what version to use for future calls.

The only way for DR-Gateway to learn the interface versions supported by DR-Client, is through examination of the “DR-GW-Version” header of the REGISTER request from DR-Client.

- When DR-Gateway receives an OPTIONS request, it examines the “DR-GW-Version” header to find out what versions are supported by DR-Client and creates an intersection against the set of versions it is willing or able to support. The resulting subset of interface versions MUST be included in a “DR-GW-Version” header when responding to the OPTIONS request. If the subset is empty (i.e. there are no compatible versions) then DR-Gateway SHOULD respond to the OPTIONS request with 488 (Not acceptable here).
- When DR-Gateway receives a REGISTER request, it follows the same procedures as described above for OPTIONS requests. In addition, when receiving a REGISTER request, DR-Gateway examines the subset of compatible interface versions and MAY choose any of the versions from that set for future calls to its registrar though it is RECOMMENDED to choose the highest version.
- DR-Gateway MUST be prepared to receive subsequent registration requests/refreshes in which the “DR-GW-Version” header differs from those in the previous requests. Especially, DR-Gateway MUST NOT permanently fixate a specific interface version because of a REGISTER or OPTIONS request. Without this requirement, interoperability would be severely limited and could prevent communication in mixed environments.

4.3.3 SOAP Interface Versioning

In order to give incompatible changes of the interface characteristics a chance in the future, DR-GW- Soap Interface introduces a specific version argument for sessionlogin in SOAP requests (see chapter 10.2.1 and 10.3.2).

4.3.3.1 General

In all the scenarios with SOAP the behavior is straight forward:

There is always a login from client to gateway.

The login MUST always carry a version indication provided by the client.

Either the GW can deal with that version or it cannot. In the latter case the GW has to reject the request – no further communication is possible.

In any case after login procedure the version is strictly defined and non-ambiguous situation is given.

4.3.3.2 Multiversion DR-GW

The gateway supporting multiple versions of DR-GW-Interface selects the version depending on the version attribute provided by the DR-Client with the login request. Another possibility would be to distinguish between different SOAP versions by using different URIs. Clients should be aware of this possibility and allow the configuration of the URI to be used.

4.3.3.3 Multiversion DR-Client

The Multiversion DR-Client is required only if the DR-GW does not support all the major versions defined at that moment. Preferably the interface described in DR-GW-Interface should simplify life for DR-Client and hide as much as possible complexity like version selection and of course mismatch as well. This dogma is triggered by the fact that there are more vendors for DR-Clients expected than suppliers of DR-GW. This dogma tries to optimize the overall effort.

If a multiversion DR-Client is needed two approaches are feasible:

- DR-Client is configured via a parameter to a certain version which is known to be supported by the particular DR-GW.
- DR-Client does some sort of version discovery by e.g. trying to login with the highest version the clients itself supports. In case of reject it tries the next lower version until there is a version discovered which is supported by both the DR-Client and the gateway. For optimized latency the discovery process should cache the result of discovery for consecutive re-connects.
- DR-Clients should allow configuration of the SOAP URI to be used for each particular version.

4.4 NETWORKING ASPECTS

4.4.1 FIREWALL

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

Due to the application of standard SIP with standard SDP each Voice-Over-IP can be introduced with SIP aware stateful firewall including Session Border Controllers to ensure the highest level of security and separation between different network domains. No static setting for RTP Ports is required as these devices keep track of sessions and explicitly and dynamically enable peer connectivity on a peer source IP Address, source port, destination IP Address, destination port.

4.4.2 NAT

For support of Network Address Translation in the network between the DR-GW-Client and the DR-GW a “SIP aware” application layer gateway has to be introduced such as the NATting device.

4.4.3 TRAFFIC TAGGING

Traffic tagging is not covered by this DR-GW-Interface (API definition) but it is considered an important point while deploying a Concentrator/Control room combination. To be able to configure those parameters on both sides this feature should be present and each application should support a predefined tagging.

4.4.4 MULTICAST IGMP

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

4.5 MANDATORY AND OPTIONAL FEATURES

4.5.1 AUDIO COMPRESSION

This section is listing which codecs a DR-GW Server side implementation may contain to be compliant with the standard.

Codec	DR-GW	DR-GW-Client
G.711	Mandatory	Optional

ACELP	Mandatory	Optional
FSTE / OSTE	Optional	Optional
Any other codecs	Optional	Optional

4.5.2 BOS END-TO-END AUDIO ENCRYPTION

Encryption state	DR-GW	DR-GW-Client
Unencrypted	Mandatory	Mandatory
End-to-end-encrypted	Optional	Optional

4.5.3 AUDIO IP DISTRIBUTION

Audio IP Distribution	DR-GW	DR-GW-Client
Unicast	Mandatory	Mandatory
Multicast	Optional	Optional

4.5.4 CONTROL METHOD SOAP-REQUEST

Method	**Mandatory Client**	**Mandatory DR-GW**	**Optional DR-GW**
DR-GW-Session			
Session_Login	X	X	
Session_Logout	X	X	
Session_Supervise	X	x	
DR-GW-Session.Events			
Session_Response		X	
Session_LoginEvent		X	
Session_LogoutEvent		X	
Session_SuperviseEvent		X	
Session_Check		X	
DR-GW-SDS			
SDS_Send		X	

Method	Mandatory Client	Mandatory DR-GW	Optional DR-GW
SDS_SendReport		X	
DR-GW-SDS.Events			
SDS_Response		X	
SDS_SendEvent		X	
SDS_ReceiveEvent		X	
SDS_ReportEvent		X	
DR-GW-Status			
Status_Send		X	
DR-GW-Status.Events			
Status_Response		X	
Status_SendEvent		X	
Status_ReceiveEvent		X	
DR-GW-OrganisationBlock			
Org_Get			X
Org_GetList			X
DR-GW-OrganisationBlock.Events			
Org_Response			X
Org_GetEvent			X
Org_GetListEvent			X
Org_Event			X
DR-GW-Group			
Group_Get		X	
Group_GetList		X	
Group_GetRadioMembers		X	
Group_GetAppMembers			X
Group_Track			X
Group_AddRadioMember			X
Group_RemoveRadioMember			X
Group_GetCombinations			X
Group_AddCombination			X
Group_RemoveCombination			X
Group_SubscribeData		X	
DR-GW-Group.Events			

Method	Mandatory Client	Mandatory DR-GW	Optional DR-GW
Group_Response		X	
Group_GetEvent		X	
Group_GetListEvent		X	
Group_GetRadioMembersEvent			X
Group_GetAppMembersEvent			X
Group_Event			X
Group_RadioMemberEvent			X
Group_AppMemberEvent			X
Group_GetCombinationsEvent			X
Group_CombinationEvent			X
Group_Group_SubscribeDataEvent			X
Group_TrackSubscriptionEvent			X
Group_AddRadioMemberEvent			X
Group_RemoveRadioMemberEvent			X
Group_AddCombinationEvent			X
Group_RemoveCombinationEvent			X
DR-GW-Application			
App_Get			X
App_GetList			X
DR-GW-Application.Events			
App_Response			X
App_GetEvent			X
App_GetListEvent			X
DR-GW-System.Events			
Sys_TETRAStatesEvent			X
Sys_LogEvent			x
DR-GW-Radio			
Radio_Get			X
Radio_GetList			X
Radio_GetGroups			X
Radio_Track			X
Radio_ChangeOpta			X
Radio_EnDisabled			X

Method	Mandatory Client	Mandatory DR-GW	Optional DR-GW
DR-GW-Radio.Events			
Radio_Response			X
Radio_GetEvent			X
Radio_GetListEvent			X
Radio_GetGroupsEvent			X
Radio_GroupsEvent			X
Radio_Event			X
Radio_TrackEvent			X
Radio_TrackSubscriptionEvent			X
Radio_OptaChanged			X
Radio_EnDisableEvent			X
DR-GW-Call			
Call_Select		X[1]	
DR-GW-Call.Events			
Call_SelectEvent		X	
Call_Event			X
Call_PTTEvent			X
Call_UnitInEmergencyEvent		X	

4.5.5 Load And Resource Sharing

This DR-GW switchover functionality is optional.

Based on the assumption that multiple servers could exist in the DR-GW 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 scope of this API document.

If a DR-GW-Client requests a resource on a server which does not have enough available resources to accommodate the DR-GW-Client, the server may request other server of the DR-GW for resource availability and return the IP address of the server which could provide the service requested in the resource request event.

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

[1] support for Call_Select limited to selection level “no” and “event” via SOAP interface required

4.5.6 DR-GW Failover functionality

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

The following DR-GW failover functionality is optional.

Based on the assumption that a DR-GW 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 scope of this API document.

If a server fails while in service, the peer server (redundancy partner) shall be able to take over the role and resources (including IP Address) of the failed server. In this case the failover would be entirely transparent to the DR-GW-Client.

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

4.5.7 DR-GW-Client Redundancy

The interface described in here does not explicitly foresee any redundant client implementation. In scenarios where the DR-GW-Client is realized as a centralized element such a redundancy might makes sense to achieve high reliability. As an implementation hint these redundant elements shall perform a local communication in a way that only one DF-GW-Client service is active at one moment. The DF-GW server strictly follows a first come first serve strategy considering the specified priorities for arbitration.

5 USE CASE DESCRIPTION

This document describes use cases for using TETRA with a gateway to TETRA net. The digital radio gateway was named DigitalRadio-Gateway (or short form DR-GW).

The context of the use cases is always the DR-GW, focused on the interface to the User. The user could be a more or less complex system or a dispatcher including some systems.

Lean diagram conditions of extensions are not drawn in a diagram, but described in corresponding texts above diagrams.

The user is connected to the DR-GW. The DR-GW is connected to the radio exchange system. The TETRA subscriber subscribes over a base station of the radio exchange system data and call services.

Utilization of the SIP Message Header Fields as well as SIP Message Bodies is explained in the specific Message Sequence Charts. Chapter 7 "Profile Standard for the use of SIP" normatively describes the use of the respective fields.

5.1 RESOURCE ALLOCATION

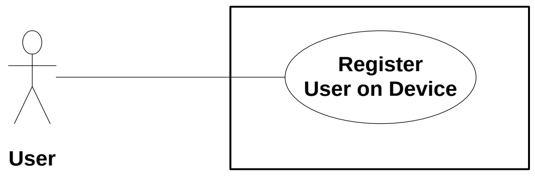


Figure 1 – Use Case Resource Allocation

If a user wants to receive individual calls from the DR-GW or a TETRA Subscriber the user has to register to the device inside the DR-GW. After registration the user is able to receive calls from the TETRA Subscriber.

Depending on the implementation of the DR-GW this registration may be used to implicitly allocate the according resources for that specific client either in an exclusive or a shared manner. Especially for the allocation of the so called B-Channel with its associated ISSI this way of resource reservation shall be treated as a strong recommendation for implementation in the DR-GW.

Before expiration of the registration Timer, the user has to refresh the registration. The user finishes his resource allocation by registering a timeout value of 0 which is the SIP "UNREGISTER" method.

5.1.1 Message Sequence Chart Description For Resource Allocation

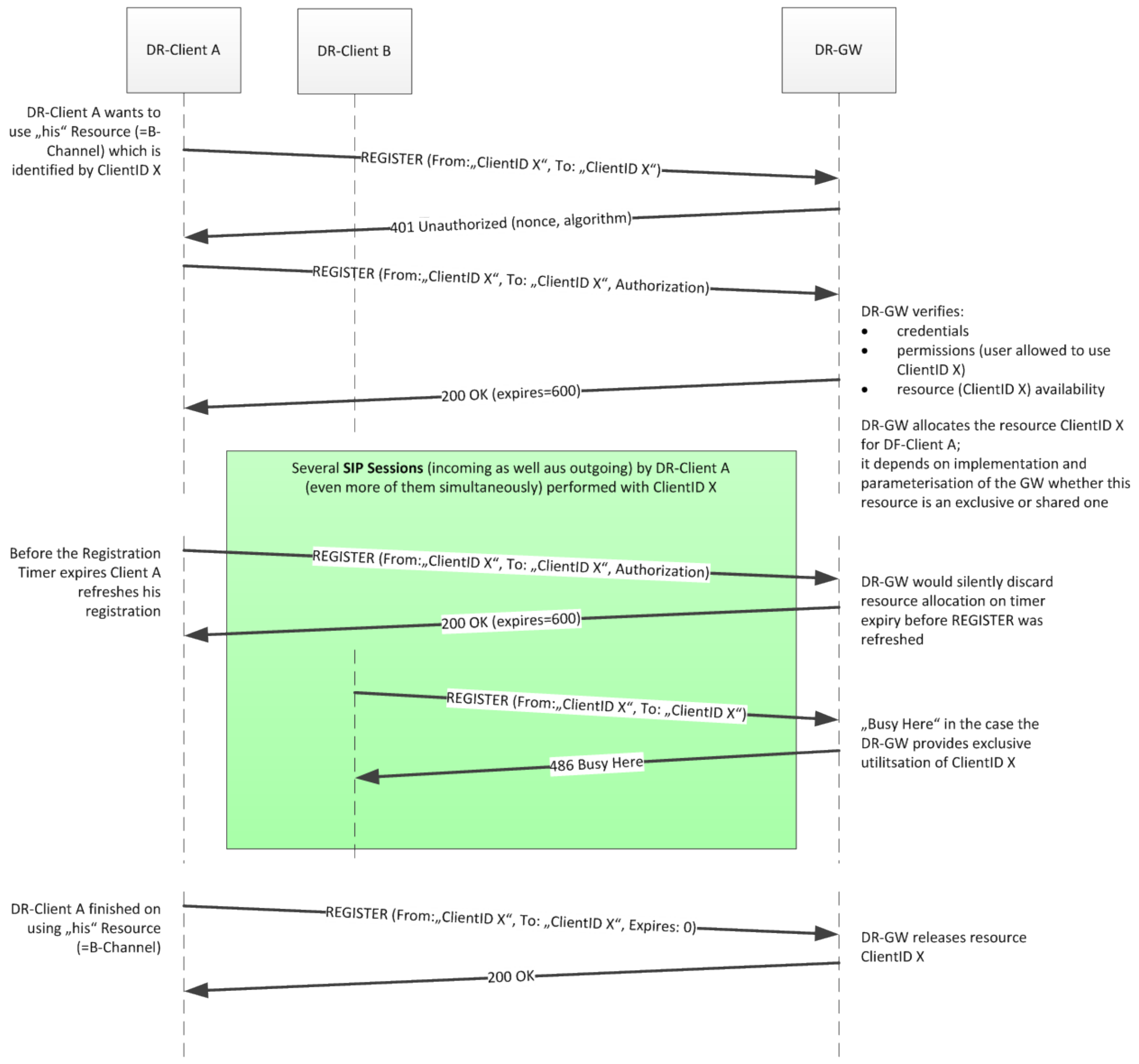

DR-Client A
DR-Client B
DR-GW
DR-Client A wants to use „his“ Resource (=B-Channel) which is identified by ClientID X
REGISTER (From:„ClientID X“, To: „ClientID X“)
401 Unauthorized (nonce, algorithm)
REGISTER (From:„ClientID X“, To: „ClientID X“, Authorization)
DR-GW verifies:
• credentials
• permissions (user allowed to use ClientID X)
• resource (ClientID X) availability
200 OK (expires=600)
DR-GW allocates the resource ClientID X for DF-Client A; it depends on implementation and parameterisation of the GW whether this resource is an exclusive or shared one
Several SIP Sessions (incoming as well aus outgoing) by DR-Client A (even more of them simultaneously) performed with ClientID X
Before the Registration Timer expires Client A refreshes his registration
REGISTER (From:„ClientID X“, To: „ClientID X“, Authorization)
DR-GW would silently discard resource allocation on timer expiry before REGISTER was refreshed
200 OK (expires=600)
REGISTER (From:„ClientID X“, To: „ClientID X“)
„Busy Here“ in the case the DR-GW provides exclusive utilitsation of ClientID X
486 Busy Here
DR-Client A finished on using „his“ Resource (=B-Channel)
REGISTER (From:„ClientID X“, To: „ClientID X“, Expires: 0)
DR-GW releases resource ClientID X
200 OK

Figure 2 – MSC for Resource Allocation

5.2 INCOMING INDIVIDUAL CALLS

Incoming call in this context means calls originated somewhere within the “Digitalfunk” propagated via the “DR-GW-Server” to the “DR-GW-Client”. As a precondition before being able to receive any incoming calls the DR-GW-Client has to allocate the resource via the SIP REGISTER method as described in chapter 5.1 “Resource allocation”

5.2.1 ESTABLISH AND TERMINATE INCOMING FULL DUPLEX INDIVIDUAL CALL

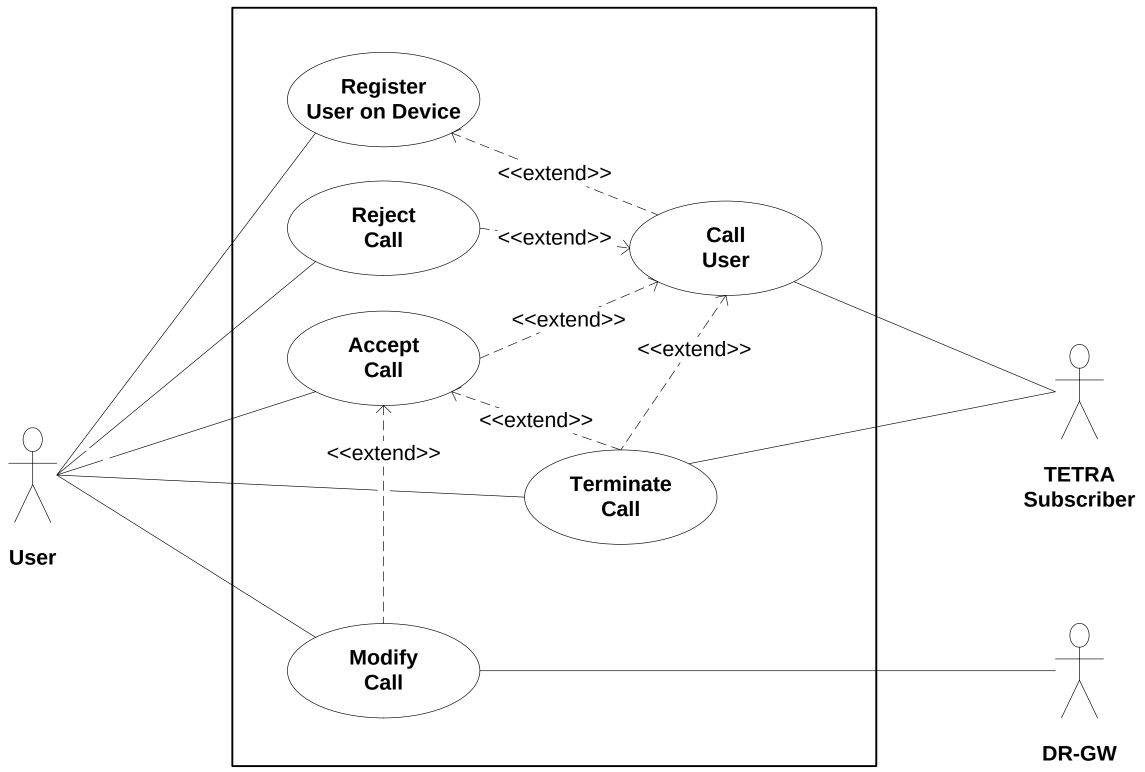


Figure 3 – Use Case establish and terminate Incoming Full Duplex Individual Call

To get signaling of calls from the TETRA subscriber a user has to register. If the user has registered, the TETRA subscriber is able to call the registered user.

The user can reject or accept a call alerted by the DR-GW. Also, an incoming and not yet accepted call can be prematurely terminated by the TETRA subscriber.

If the user accepts the call, both parties, the caller DR-GW and the callee User are considered to be in a dialog. Both, the user and DR-GW can modify session parameters inside the dialog.

The call can be torn down by either participant, the user or the TETRA subscriber (represented by DR-GW).

In case of non-hook calls, acceptance of the call was done automatically by the radio exchange system.

5.2.2 Establish and Terminate Incoming Half Duplex Individual Call

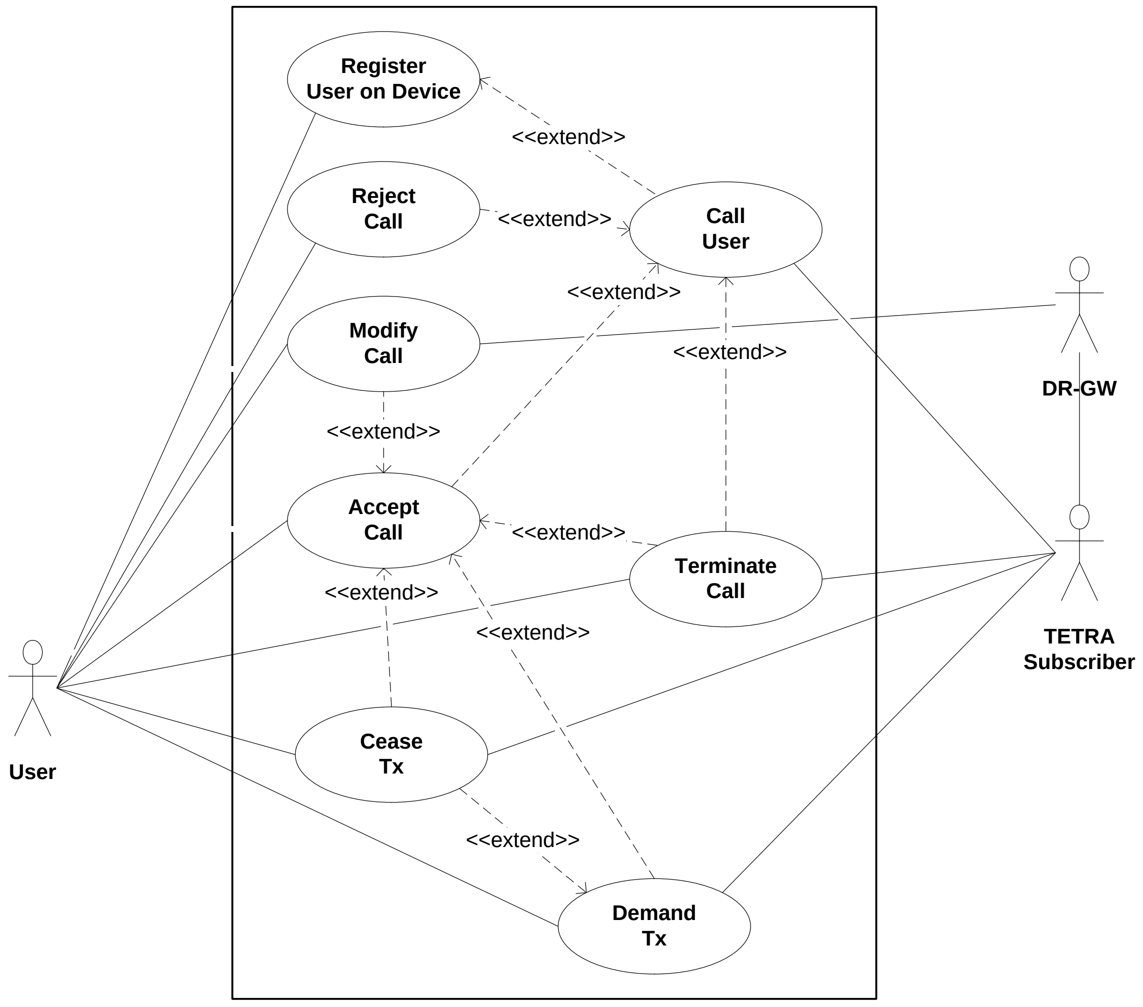


Figure 4 – Use Case establish and terminate Incoming Half Duplex Individual Call

To receive signaling of incoming calls from a TETRA subscriber the user has to register first. If the user is registered, the TETRA subscriber is able to call the registered user.

The user can reject or accept a call alerted by the DR-GW. Also, an incoming and not yet accepted call can be prematurely terminated by the TETRA subscriber.

After the call is accepted by the user, either side can demand Tx – the radio exchange system queues or grants Tx after demanding. After Tx has been granted to one participant the subscriber with the speech item is able to talk. The party with the speech item can cease Tx. Both, the user and DR-GW can modify session parameters inside the dialog.

The call can be torn down by either participant, the user or the TETRA subscriber (represented by DR-GW).In case of non-hook calls, acceptance of the call is done automatically by the radio exchange system.

“Modify” in this chapter’s sense is related to the SIP sessions only. It is used to negotiate new IP address/port pairs and it will be performed via SIP UPDATE or re-INVITE method.

5.2.3 MESSAGE SEQUENCE CHART DESCRIPTION FOR INCOMING INDIVIDUAL CALLS

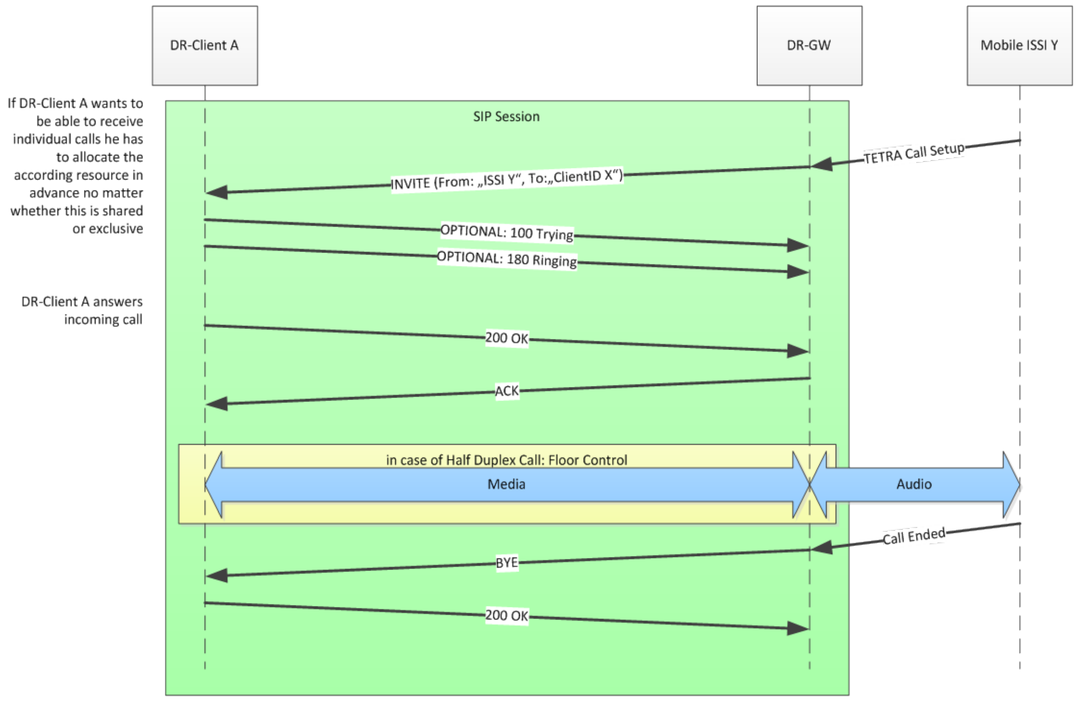

DR-Client A
DR-GW
Mobile ISSI Y
If DR-Client A wants to be able to receive individual calls he has to allocate the according resource in advance no matter whether this is shared or exclusive
SIP Session
TETRA Call Setup
INVITE (From: „ISSI Y“, To:„ClientID X“)
OPTIONAL: 100 Trying
OPTIONAL: 180 Ringing
DR-Client A answers incoming call
200 OK
ACK
in case of Half Duplex Call: Floor Control
Media
Audio
Call Ended
BYE
200 OK

Figure 5 – MSC for Incoming Individual Call

Both TETRA call establishment and key exchange process within the TETRA system may run concurrently to the SIP Session establishment and may not complete before the SIP session has been successfully established (with the ACK message). In order to enable DR-Client to cope with this situation “Call_Event” and “Call_KeyExchangeEvent” notification can be provided as an optional DR-GW feature.

Depending on which party releases the call first, the BYE will be sent by either the DR-GW (as shown in Figure 6) or by the DR-GW-Client.

5.3 USE CASES FOR OUTGOING INDIVIDUAL CALLS

5.3.1 ESTABLISH AND TERMINATE OUTGOING FULL DUPLEX INDIVIDUAL CALL

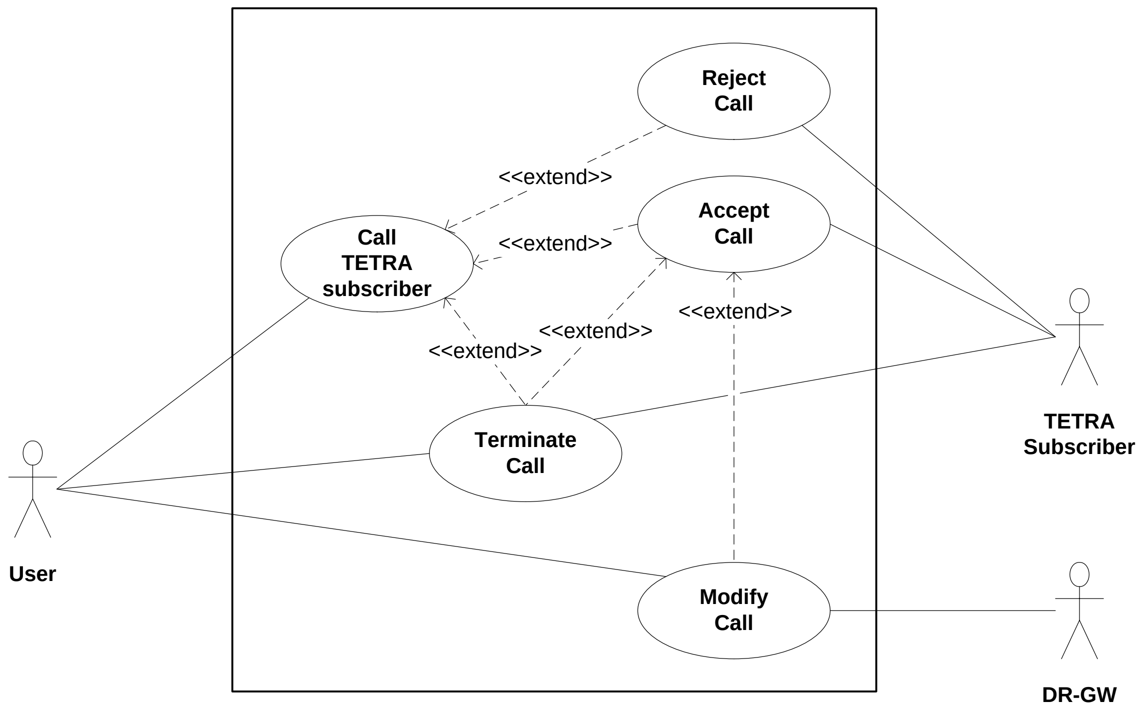


Figure 6 – Use Case establish and terminate Outgoing Full Duplex Individual Call

The user can call a TETRA subscriber.

The TETRA subscriber can reject or accept the calls alerted by the DR-GW. The user can terminate calls before they have been accepted/rejected by the TETRA subscriber.

If the TETRA subscriber accepts the call both parties, the caller user and the callee TETRA subscriber represented by the DR-GW, have a dialog and can talk to each other. The user or the DR-GW itself can modify the accepted call – the TETRA subscriber cannot modify the call.

The user and the TETRA subscriber represented by the DR-GW can terminate the established dialog.

For an outgoing individual call, registration of the user to a device is not necessary.

In case of non-hook calls, acceptance of the call will be done automatically by the radio exchange system.

5.3.2 ESTABLISH AND TERMINATE OUTGOING HALF DUPLEX INDIVIDUAL CALL

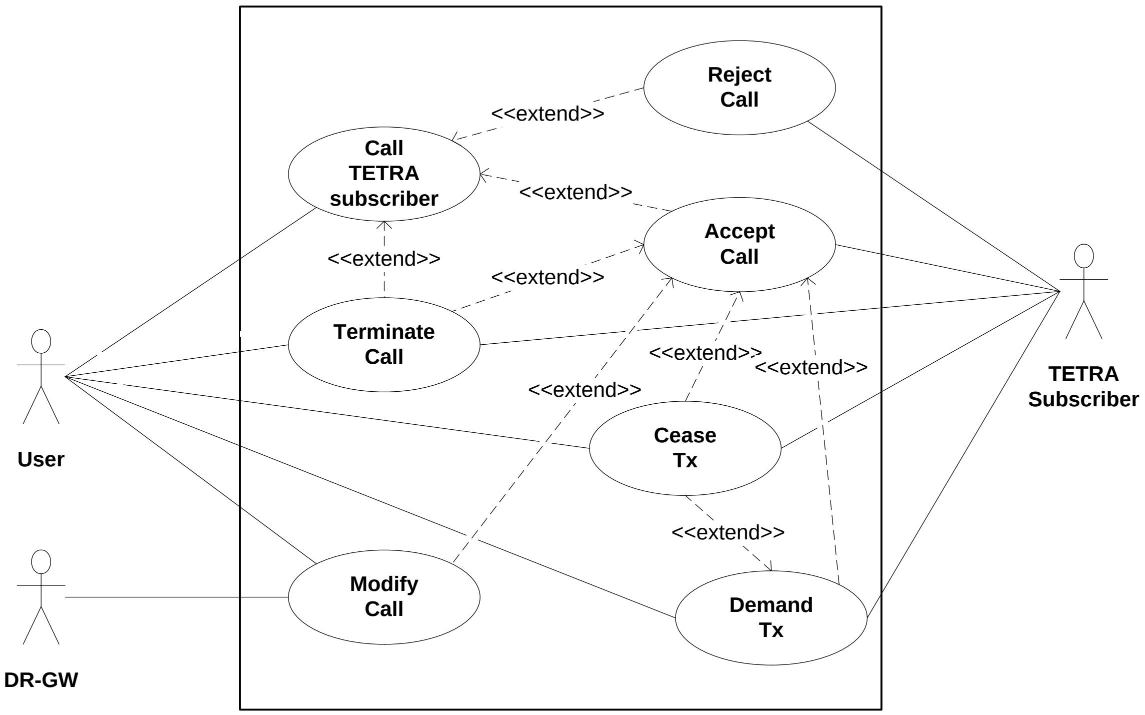


Figure 7 – Use Case establish and terminate Outgoing Half Duplex Individual Call

User could call the TETRA subscriber.

The TETRA subscriber could either reject or accept the call alerted by the DR-GW. When the TETRA subscriber has not rejected or accepted the call, the user could terminate the alerted call.

After call is accepted by the TETRA subscriber, he and the user could demand Tx – radio exchange system queue or grant Tx after demanding. After Tx has been granted to one call party the subscriber with the speech item could talk. The party with the speech item could cease Tx. Accepted calls could be modified by the user or the DR-GW.

The user and the TETRA subscriber represented by the DR-GW could terminate the established dialog.

For an outgoing individual call registration of user for a device is not necessary.

In case of non-hook calls, acceptance of the call was done automatically by the radio exchange system.

5.3.3 MESSAGE SEQUENCE CHART FOR OUTGOING INDIVIDUAL CALL

The example message sequence chart below shows an individual call initiated by the user. It depicts the establishment of the SIP session including the digest authentication (“407

Proxy Authentication Required”). If the call is a half-duplex call, floor control is used for arbitration of the media. Half or full-duplex call is announced either via SDP and / or via xml message body as SIP INVITE’s message body.

The example shows the call being torn down by the TETRA subscriber.

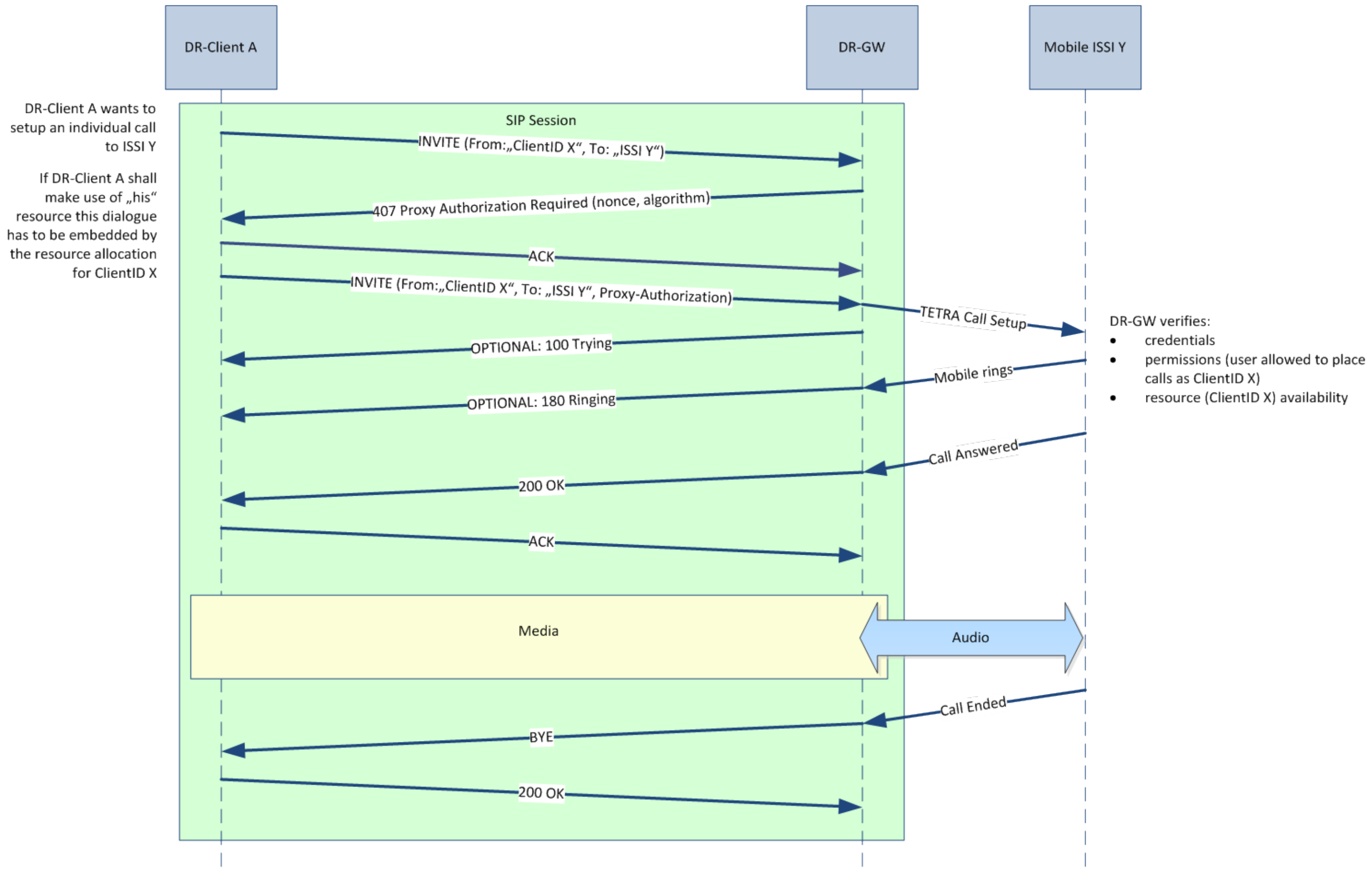
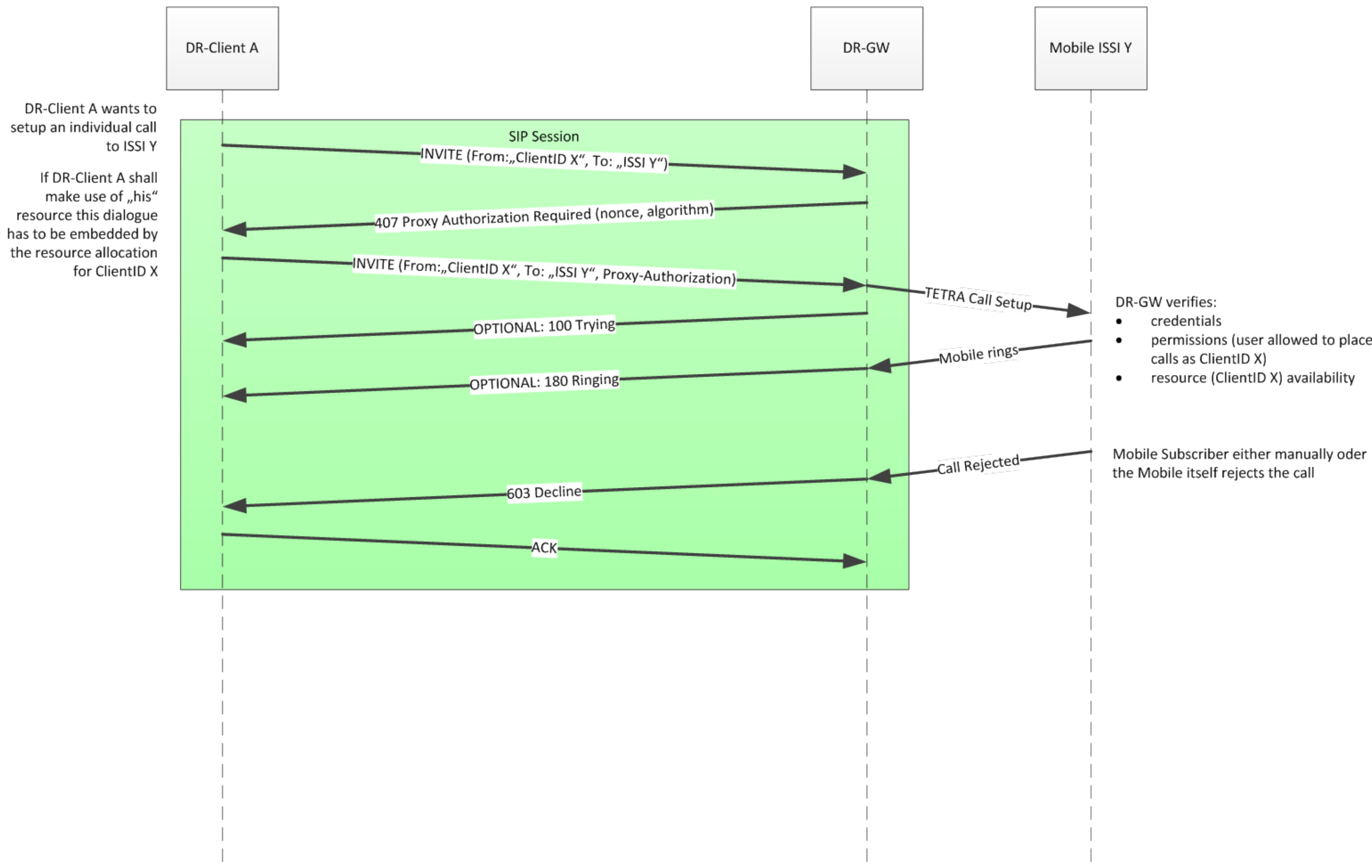

DR-Client A
DR-GW
Mobile ISSI Y
DR-Client A wants to setup an individual call to ISSI Y
If DR-Client A shall make use of „his“ resource this dialogue has to be embedded by the resource allocation for ClientID X
SIP Session
INVITE (From:„ClientID X“, To: „ISSI Y“)
407 Proxy Authorization Required (nonce, algorithm)
ACK
INVITE (From:„ClientID X“, To: „ISSI Y“, Proxy-Authorization)
TETRA Call Setup
DR-GW verifies:
credentials
permissions (user allowed to place calls as ClientID X)
resource (ClientID X) availability
OPTIONAL: 100 Trying
Mobile rings
OPTIONAL: 180 Ringing
Call Answered
200 OK
ACK
Media
Audio
Call Ended
BYE
200 OK

Figure 8 – MSC for Outgoing Individual Call

Before answering the TETRA subscriber may reject the offered call which is shown in the following message sequence chart.



DR-Client A
DR-GW
Mobile ISSI Y
DR-Client A wants to setup an individual call to ISSI Y
If DR-Client A shall make use of „his“ resource this dialogue has to be embedded by the resource allocation for ClientID X
SIP Session
INVITE (From:„ClientID X“, To: „ISSI Y“)
407 Proxy Authorization Required (nonce, algorithm)
INVITE (From:„ClientID X“, To: „ISSI Y“, Proxy-Authorization)
TETRA Call Setup
DR-GW verifies:
• credentials
• permissions (user allowed to place calls as ClientID X)
• resource (ClientID X) availability
OPTIONAL: 100 Trying
Mobile rings
OPTIONAL: 180 Ringing
Call Rejected
Mobile Subscriber either manually oder the Mobile itself rejects the call
603 Decline
ACK

Figure 9 – MSC for GW Rejected Outgoing Individual Call

5.4 USE CASES FOR GROUP CALLS

5.4.1 EVENT MONITORING

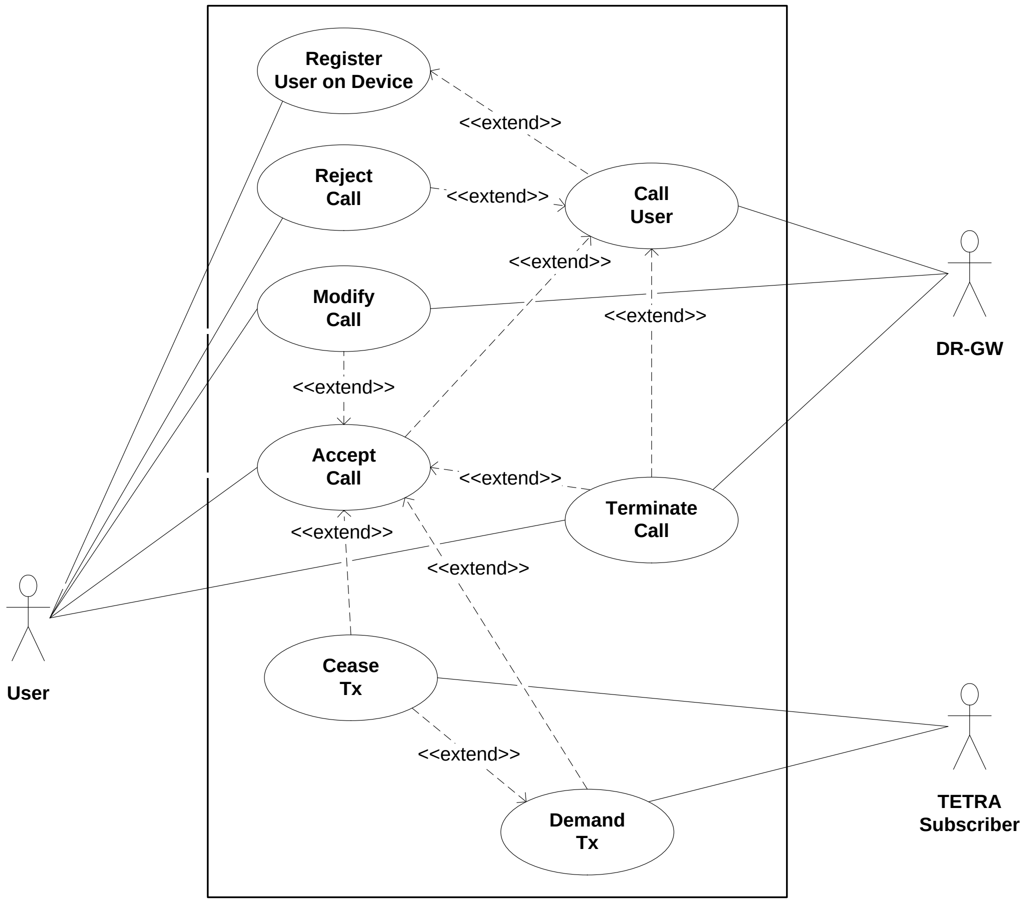


Figure 10 – Use Case Event Monitoring of TETRA Group

To get events of a TETRA group the user has to invite the group to a non-audio SIP Session. The session has to be established to the group in terms of a device. A device can be an identifier, as well as a calling line or number. If the device is an identifier, the DR-GW maps the identifier to a calling line or number. By calling the group the user defines the Selection Mode for group usage: in this case event monitoring so that no audio has been transported between DR-GW and the user.

If the user has established a session for a device, the DR-GW can notify in case of events.

The user can reject or accept a call alerted by the DR-GW.

After the session is established, the DR-GW signals the User, if any TETRA-Subscriber on that particular group has demanded Tx–radio exchange system queue or grant Tx after demanding. After Tx has been granted to one call party the subscriber with the speech item could talk – only signaling would be transported in case of event monitoring. The party with the speech item could cease Tx. Accepted calls could be modified by the user or by the DR-GW.

Regularly the user terminates the established dialog while the DR-GW tears down the session only in cases where an error occurs.

5.4.2 AUDIO MONITORING

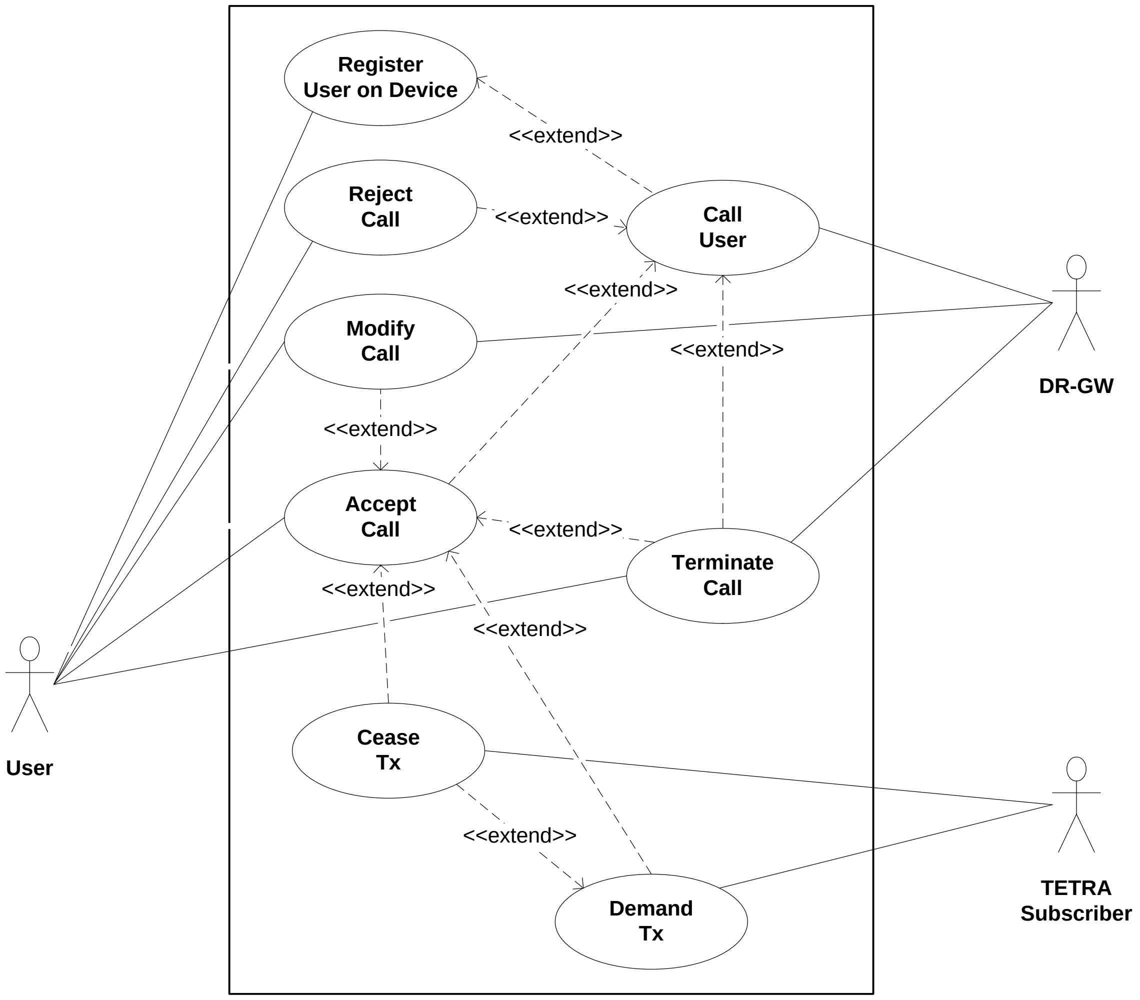


Figure 11 – Use Case Audio Monitoring of TETRA Group

To monitor audio of a TETRA group the user has to call the group. By calling the group the user defines the Selection Mode for group usage: in this case audio monitoring – so that signaling and audio can be transported between the DR-GW and the User. The DR-GW has to accept the call in order to be able to propagate the TETRA groups audio to the user.

After the call is accepted, the DR-GW signals the User when the TETRA-Subscriber has demanded Tx – radio exchange system queue or grant Tx after demanding. After Tx has been granted to one call party the subscriber with the speech item could talk–signaling and audio would be transported in case of audio monitoring. The party with the speech item could cease Tx. Accepted calls could be modified by the user or by the DR-GW.

The user and the DR-GW could terminate the established dialog. The user terminates the dialog if monitoring of the group is not wanted anymore. In case of error occurrence the DR-GW tears down the SIP session by terminating the dialog.

5.4.3 Use TETRA Group

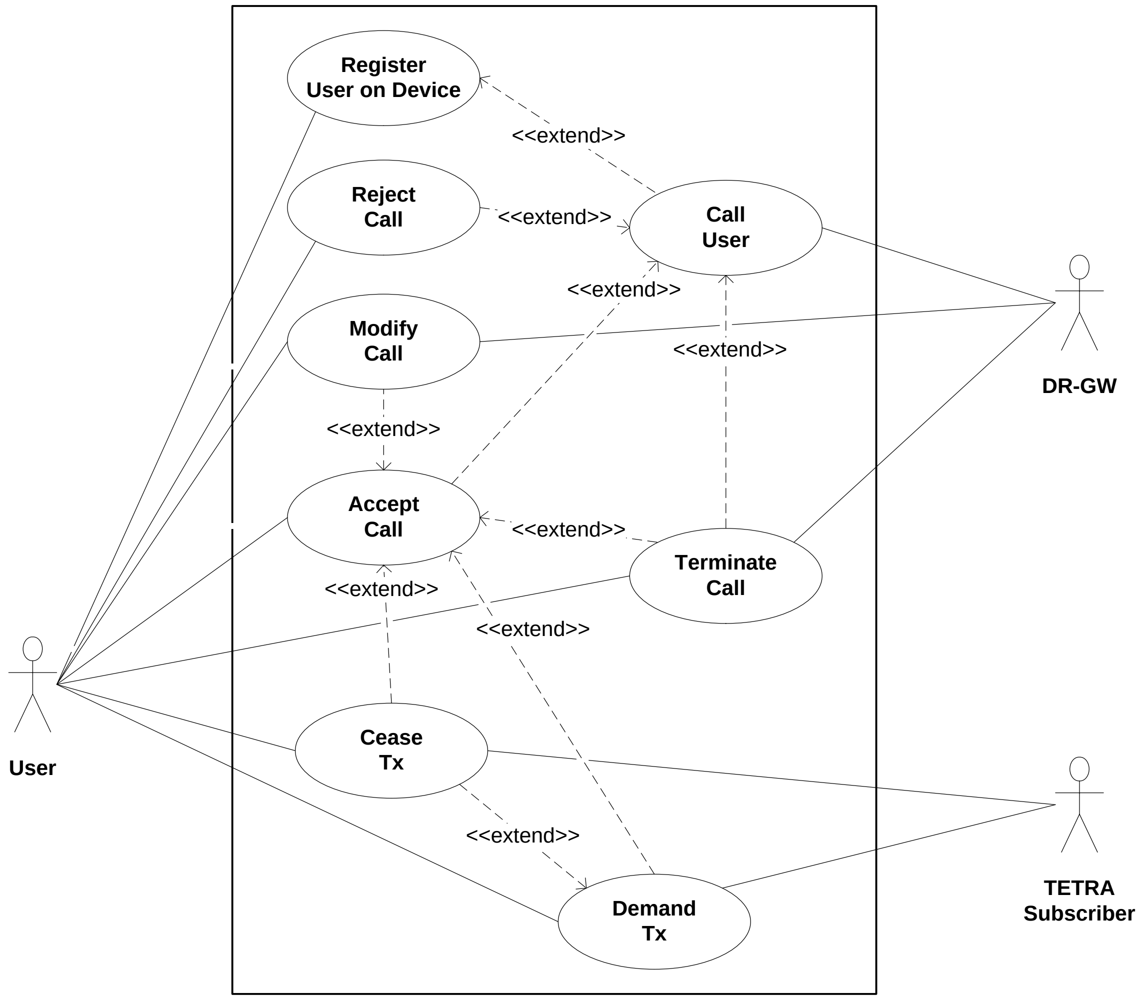


Figure 12 – Use Case Use TETRA Group

For using a TETRA group the use cases are similar to the use case of monitoring audio of a TETRA group. When using the TETRA not only the TETRA subscriber could demand and cease Tx, but the user as well.

To use a TETRA group the user has to invite this group to a SIP session. If the user wants to use his exclusive B-Channel for this purpose, he has to register himself in terms of a device. A device could be an identifier, as well as a calling line or number. If the device is an identifier, the DR-GW maps the identifier to a calling line or number. Without any registration it is a matter of DR-GW implementation which identification within the digital radio network will be used when the user demands (and is granted for) Tx.

By calling the group the user defines the Selection Mode for group usage: in this case use of the talk group– so that signaling and audio can be transported between the DR-GW and the User. The DR-GW has to accept the call in order to enable the audio path from the user to the TETRA talk group and vice versa.

After the call is established, the DR-GW signals the User when the TETRA-Subscriber has demanded Tx – radio exchange system queue or grant Tx after demanding. After Tx has been granted to one call party the subscriber with the speech item could talk–signaling and audio would be transported in this case. The party with the speech item could cease Tx. Accepted calls could be modified by the user or by the DR-GW.

The user and the DR-GW could terminate the established dialog. Regularly the user terminates the dialog if he does not want to use the talk group anymore. In case of error occurrence the DR-GW tears down the SIP session which is equal to terminate the dialog.

5.4.4 CHANGE SELECTION MODE FOR GROUP CALLS

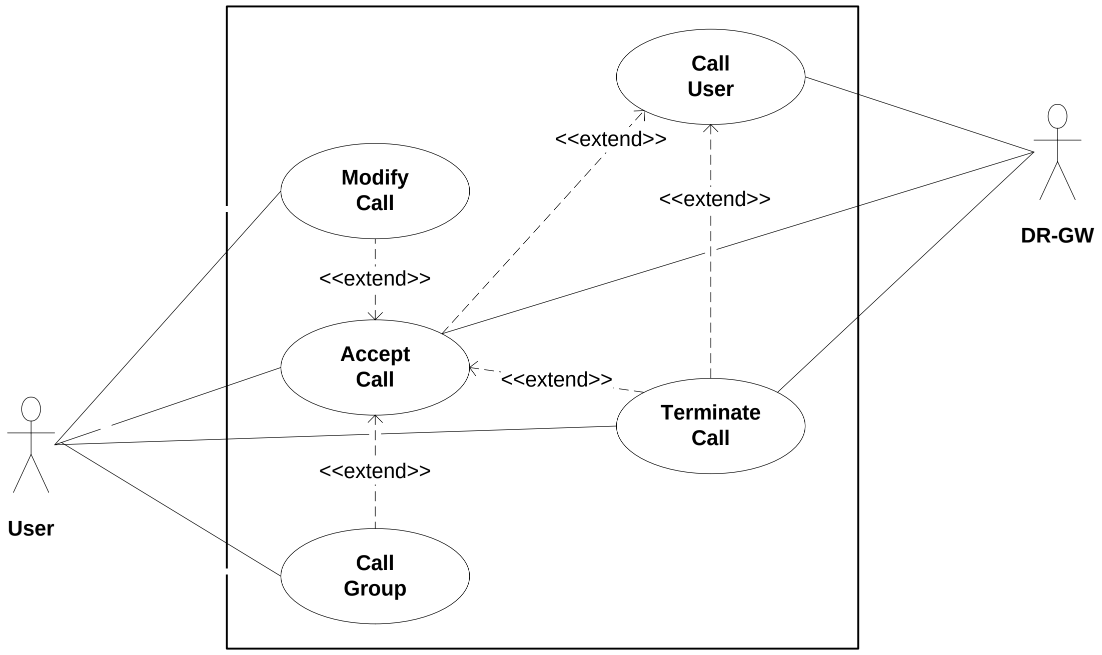


Figure 13 – Use Case SELECTION MODE FOR GROUP CALLS

To change the Selection Mode for group calls the user could modify the call. Modification of the call is possible only, when the dialogue is confirmed already. For this reason clients can use re-INVITE to change selection (see chapter 5.1 of RFC3311, which RECOMMENDS re-INVITE as the method of choice for modifying established sessions).

Note that UPDATE may also be used if support for this method is indicated by either side.

5.4.5 DISCONNECT GROUP CALL

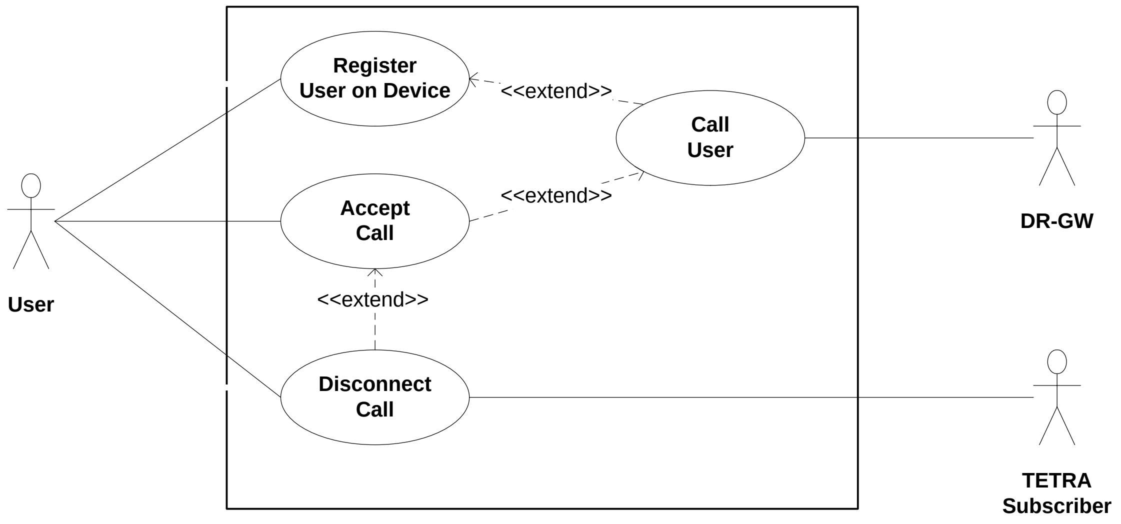
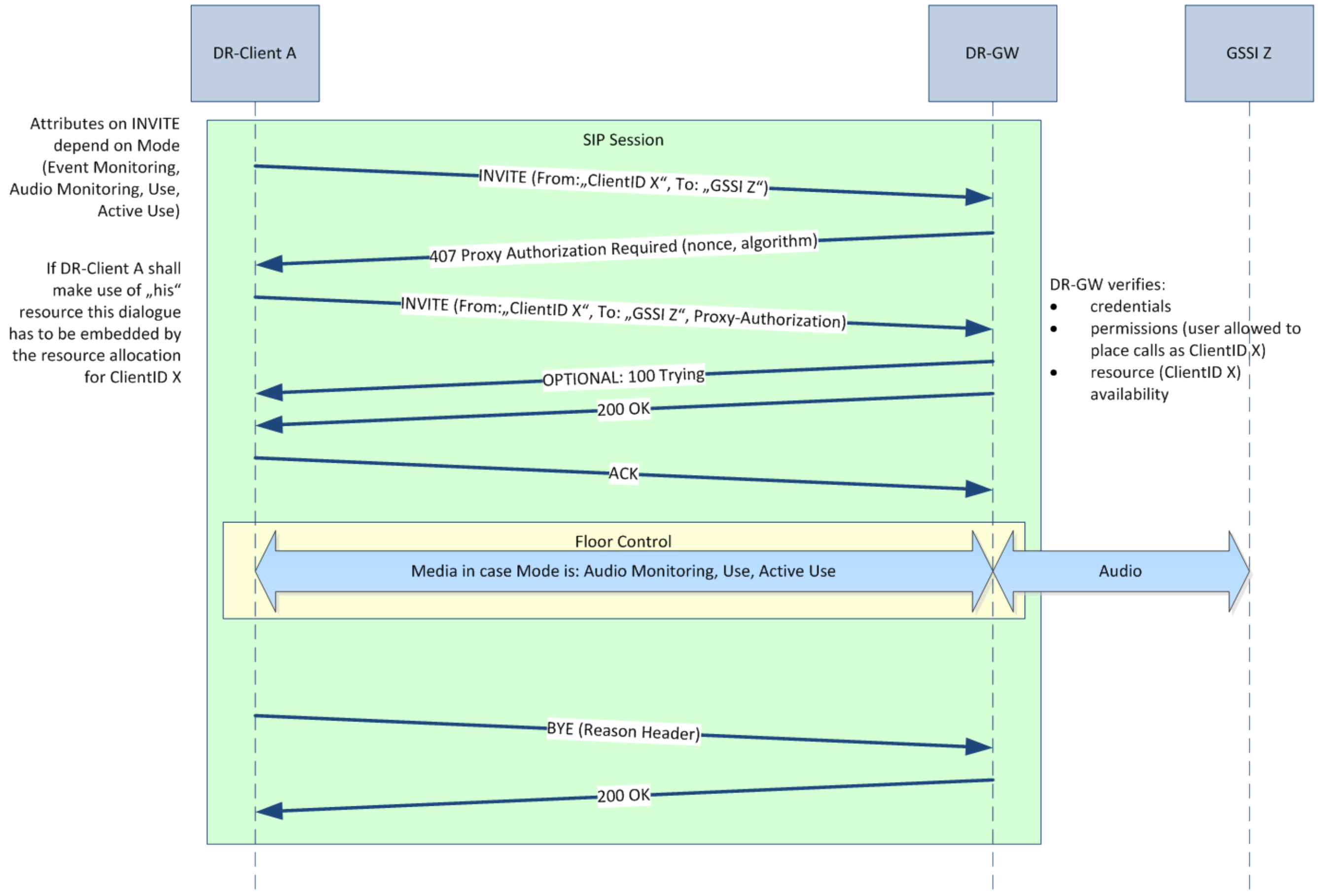


Figure 14 – Use Case Disconnect GROUP CALLS

Once the SIP session for the group call is established the user could disconnect the call. The TETRA subscriber or the DR-GW (including exchange) could disconnect the call as well.

5.4.6 Message Sequence Chart Description for Group Call



DR-Client A
DR-GW
GSSI Z
Attributes on INVITE depend on Mode (Event Monitoring, Audio Monitoring, Use, Active Use)
SIP Session
INVITE (From:„ClientID X“, To: „GSSI Z“)
407 Proxy Authorization Required (nonce, algorithm)
If DR-Client A shall make use of „his“ resource this dialogue has to be embedded by the resource allocation for ClientID X
INVITE (From:„ClientID X“, To: „GSSI Z“, Proxy-Authorization)
DR-GW verifies:
• credentials
• permissions (user allowed to place calls as ClientID X)
• resource (ClientID X) availability
OPTIONAL: 100 Trying
200 OK
ACK
Floor Control
Media in case Mode is: Audio Monitoring, Use, Active Use
Audio
BYE (Reason Header)
200 OK

Figure 15 – MSC for successful Group Call

In standard behavior the call is cleared by the DR-GW-Client with the SIP BYE message as shown in Figure 15. The DR-GW shall tear down the SIP Session only in case of an error. In such case the DR-GW Reason Header has to be provided to indicate the specific failure according to [0].

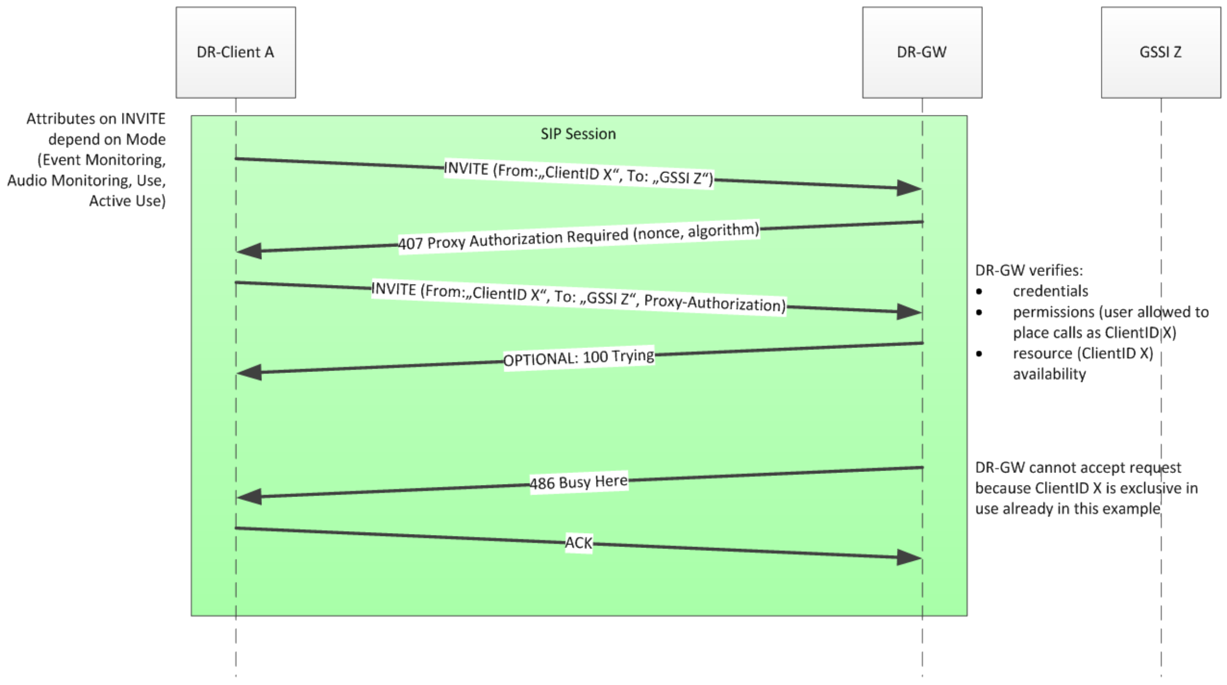
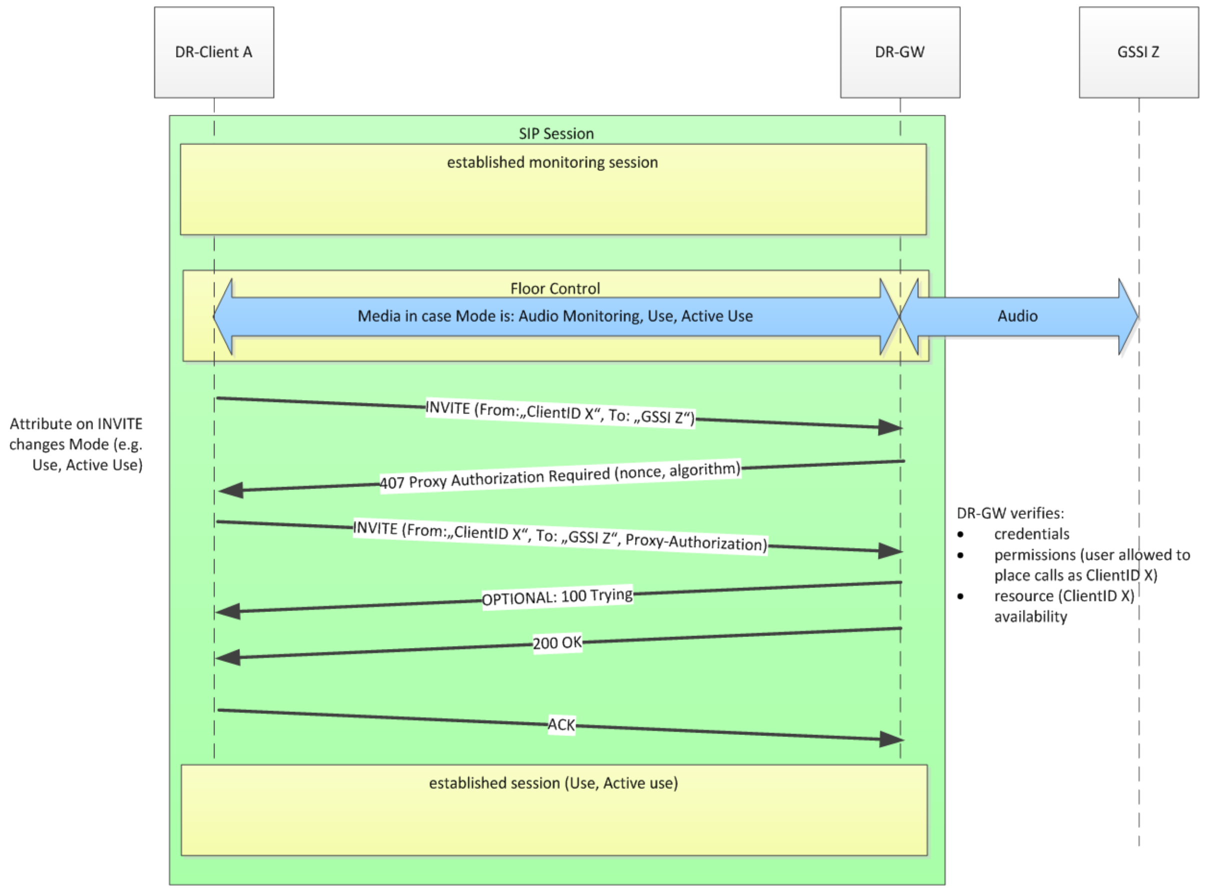

DR-Client A
DR-GW
GSSI Z
Attributes on INVITE depend on Mode (Event Monitoring, Audio Monitoring, Use, Active Use)
SIP Session
INVITE (From:„ClientID X“, To: „GSSI Z“)
407 Proxy Authorization Required (nonce, algorithm)
INVITE (From:„ClientID X“, To: „GSSI Z“, Proxy-Authorization)
OPTIONAL: 100 Trying
486 Busy Here
ACK
DR-GW verifies:
• credentials
• permissions (user allowed to place calls as ClientID X)
• resource (ClientID X) availability
DR-GW cannot accept request because ClientID X is exclusive in use already in this example

Figure 16 – MSC for GW rejected Group Call



DR-Client A
DR-GW
GSSI Z
SIP Session
established monitoring session
Floor Control
Media in case Mode is: Audio Monitoring, Use, Active Use
Audio
Attribute on INVITE changes Mode (e.g. Use, Active Use)
INVITE (From:„ClientID X“, To: „GSSI Z“)
407 Proxy Authorization Required (nonce, algorithm)
INVITE (From:„ClientID X“, To: „GSSI Z“, Proxy-Authorization)
OPTIONAL: 100 Trying
200 OK
ACK
DR-GW verifies:
• credentials
• permissions (user allowed to place calls as ClientID X)
• resource (ClientID X) availability
established session (Use, Active use)

Figure 17 – MSC for GW accepted Selection Mode change

Refer to RFC 6141 for rainy day scenarios and race conditions. Please note that a failed change of selection mode SHOULD not tear down the existing monitoring (even monitoring) session.

5.5 Floor Control

5.5.1 Floor Life Cycle

Once a Talkgroup is selected the DR-GW-Client can start opening the floor. Explicitly (as shown in Figure 18) or implicitly (on first Tx Demand) the TETRA call SHALL be established by the DR-GW. After granting Tx by the DR-GW the floor is opened – the DR-GW-Client is able to use the audio connection. This audible connection is closed when the Tx is ceased. After the TETRA timeout or when requested by the DR-GW-Client the DR-GW optionally tears down the TETRA call.

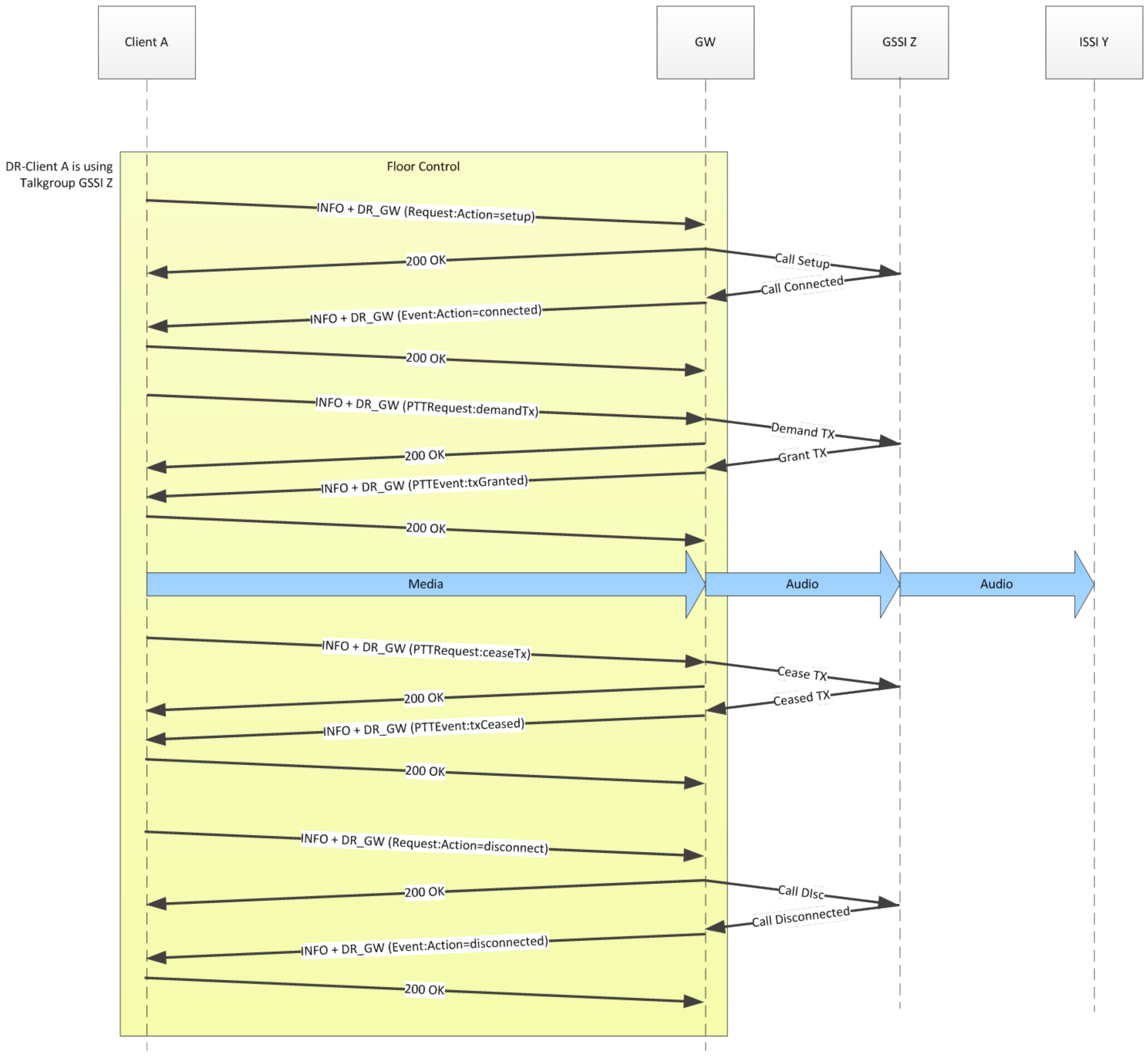

Client A
GW
GSSI Z
ISSI Y
DR-Client A is using Talkgroup GSSI Z
Floor Control
INFO + DR_GW (Request:Action=setup)
200 OK
Call Setup
Call Connected
INFO + DR_GW (Event:Action=connected)
200 OK
INFO + DR_GW (PTTRequest:demandTx)
Demand TX
Grant TX
200 OK
INFO + DR_GW (PTTEvent:txGranted)
200 OK
Media
Audio
Audio
INFO + DR_GW (PTTRequest:ceaseTx)
Cease TX
Ceased TX
200 OK
INFO + DR_GW (PTTEvent:txCeased)
200 OK
INFO + DR_GW (Request:Action=disconnect)
200 OK
Call Disc
Call Disconnected
INFO + DR_GW (Event:Action=disconnected)
200 OK

Figure 18 – MSC Floor Life Cycle

5.5.2 Floor Control Outgoing

Before the DR-GW-Client is able to transmit audio the Client has to demand the floor which has to be granted by the DR-GW. There is no need for the client to explicitly set up the TETRA call even though there is a possibility to do so. If the TETRA call is not yet established and the DR-GW-Client does not explicitly set it up, the DR-GW will set up the TETRA call implicitly.

To allow the DR-GW a safety feature such as “stuck-PTT detection” (in case of simple DR-GW-Clients similar to a plain SIP Phone with latching PTT activation) the DR-GW MAY announce a required refresh timeout in his “TXGranted” messages. If the DR-GW-Client does not refresh his Tx demand within the required timeout the DR-GW should revoke the Tx demand automatically.

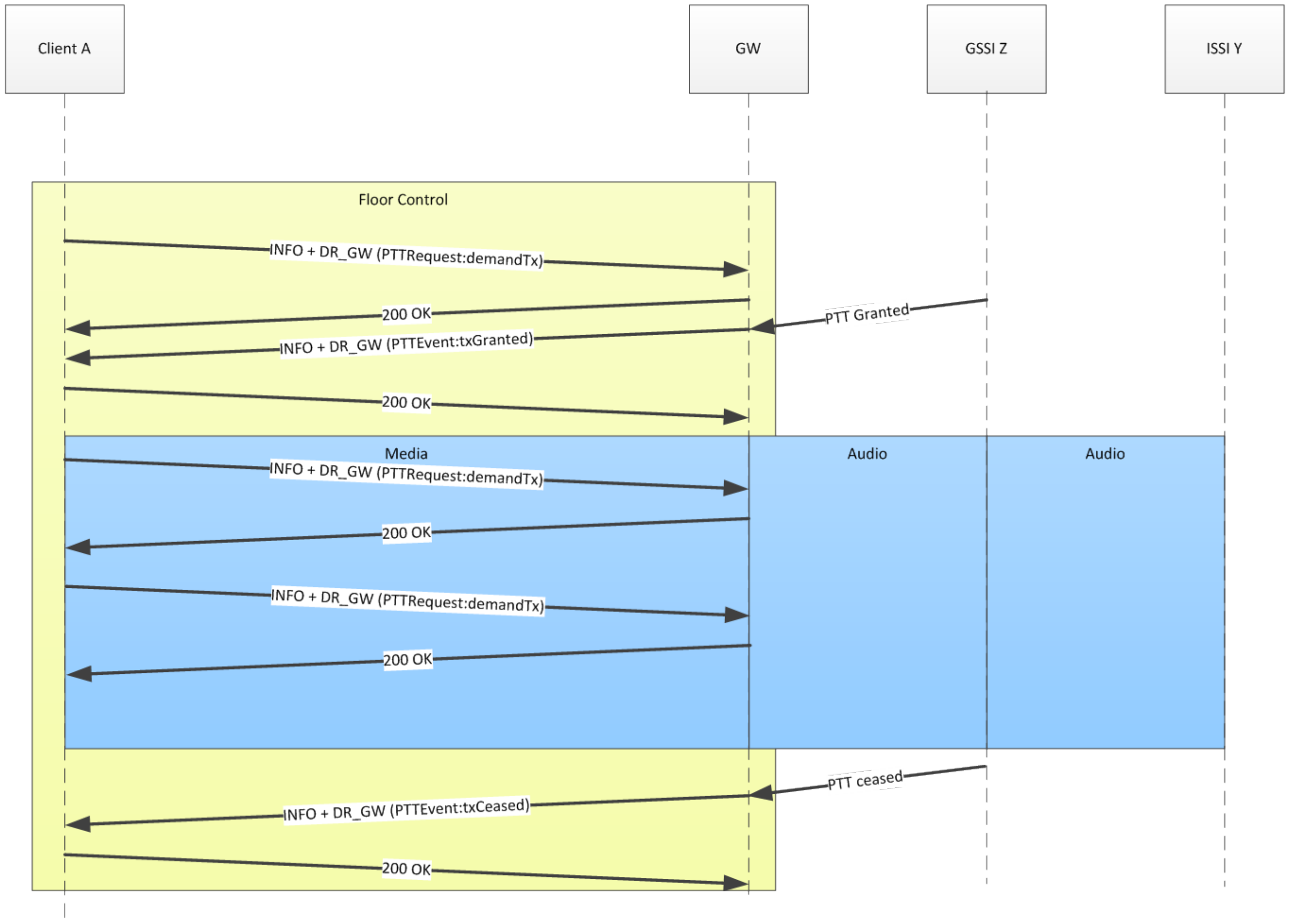

Client A
GW
GSSI Z
ISSI Y
Floor Control
INFO + DR_GW (PTTRequest:demandTx)
200 OK
PTT Granted
INFO + DR_GW (PTTEvent:txGranted)
200 OK
Media
Audio
Audio
INFO + DR_GW (PTTRequest:demandTx)
200 OK
INFO + DR_GW (PTTRequest:demandTx)
200 OK
PTT ceased
INFO + DR_GW (PTTEvent:txCeased)
200 OK

Figure 19 – MSC Floor Control Out

5.5.3 Floor Control Incoming

When the DR-GW-Client at least monitors events of a talkgroup each change of audio arbitration on that talkgroup is indicated via SIP INFO.

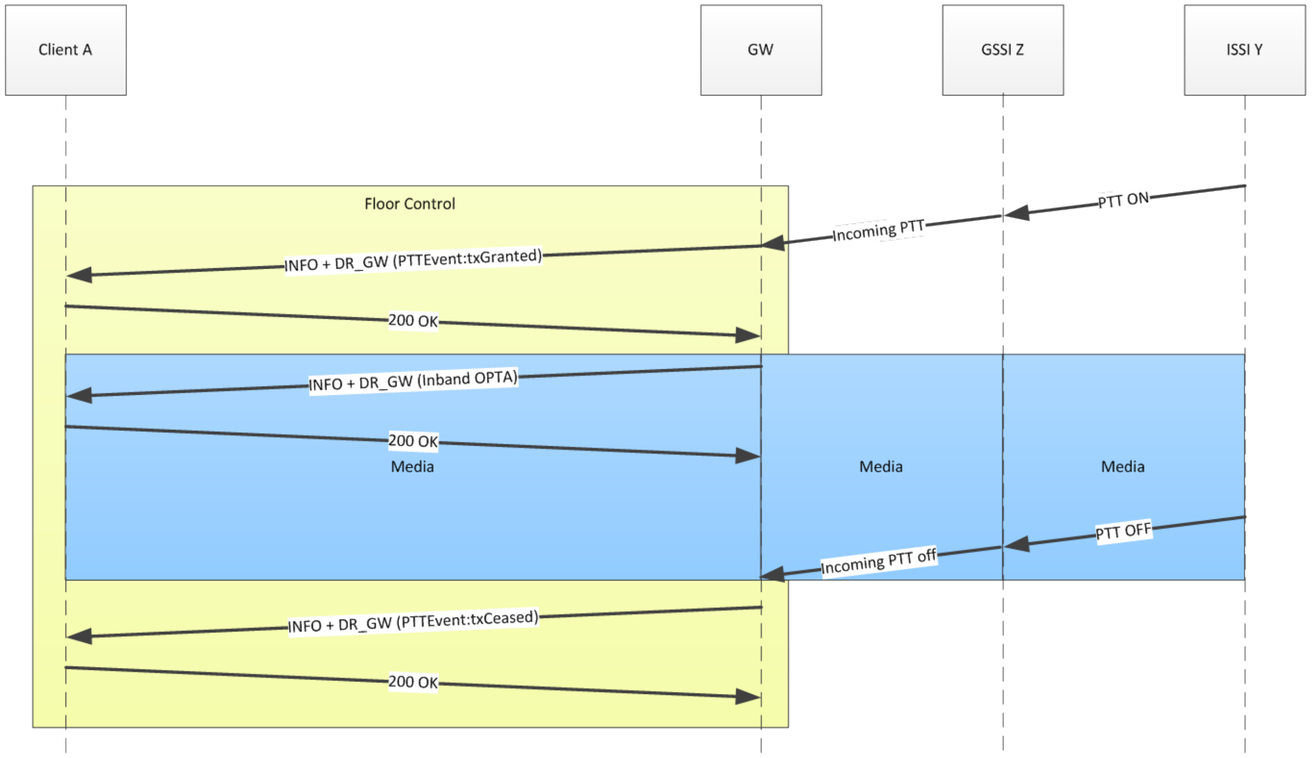


Figure 20 – MSC Floor Control In

5.6 Triggering of Key Exchange

5.6.1 Demanding new key for a TETRA Group

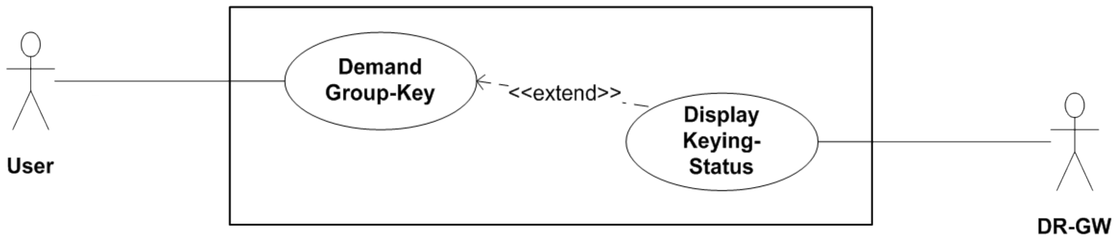


Figure 21 – Use Case Demanding a new key for a TETRA Group

The user may demand a new Key for a TETRA group. In case of rekeying the DR-GW informs the user about the keying status.

5.6.2 Providing a new key for a TETRA Group

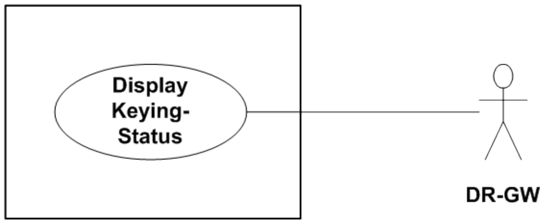


Figure 22 – Use Case Providing a new key for a TETRA Group

In the use case description, the presentation of the key status of a group or its change is shown from the direction of DR-GW.

5.6.3 Message sequence charts

When one client demands a new key for a specific TETRA Group, each client which at least monitors events of that TETRA Group will be notified about the key exchange. To prevent the TETRA infrastructure from a denial-of-service attack (even if it is performed by mistake) the DR-GW shall accept and forward Key Exchange Request with a limited maximum rate.

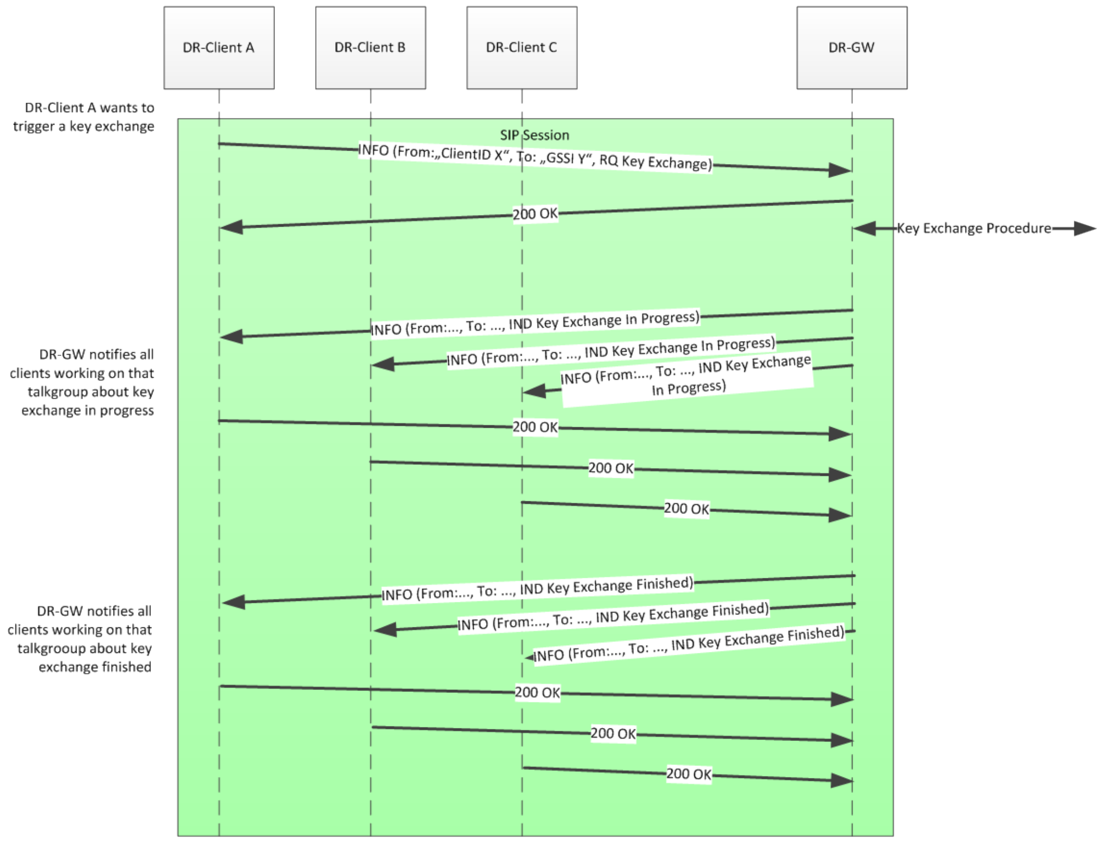

DR-Client A
DR-Client B
DR-Client C
DR-GW
DR-Client A wants to trigger a key exchange
SIP Session
INFO (From:„ClientID X“, To: „GSSI Y“, RQ Key Exchange)
200 OK
Key Exchange Procedure
INFO (From:..., To: ..., IND Key Exchange In Progress)
INFO (From:..., To: ..., IND Key Exchange In Progress)
INFO (From:..., To: ..., IND Key Exchange In Progress)
DR-GW notifies all clients working on that talkgroup about key exchange in progress
200 OK
200 OK
200 OK
INFO (From:..., To: ..., IND Key Exchange Finished)
INFO (From:..., To: ..., IND Key Exchange Finished)
INFO (From:..., To: ..., IND Key Exchange Finished)
DR-GW notifies all clients working on that talkgroup about key exchange finished
200 OK
200 OK
200 OK

Figure 23 – MSC for DR-GW-Client demanding a new key

While Figure 23 shows successful demand for a new key for a certain group by one DR-GW-Client, Figure 24 shows demand which does not lead to a successful result.

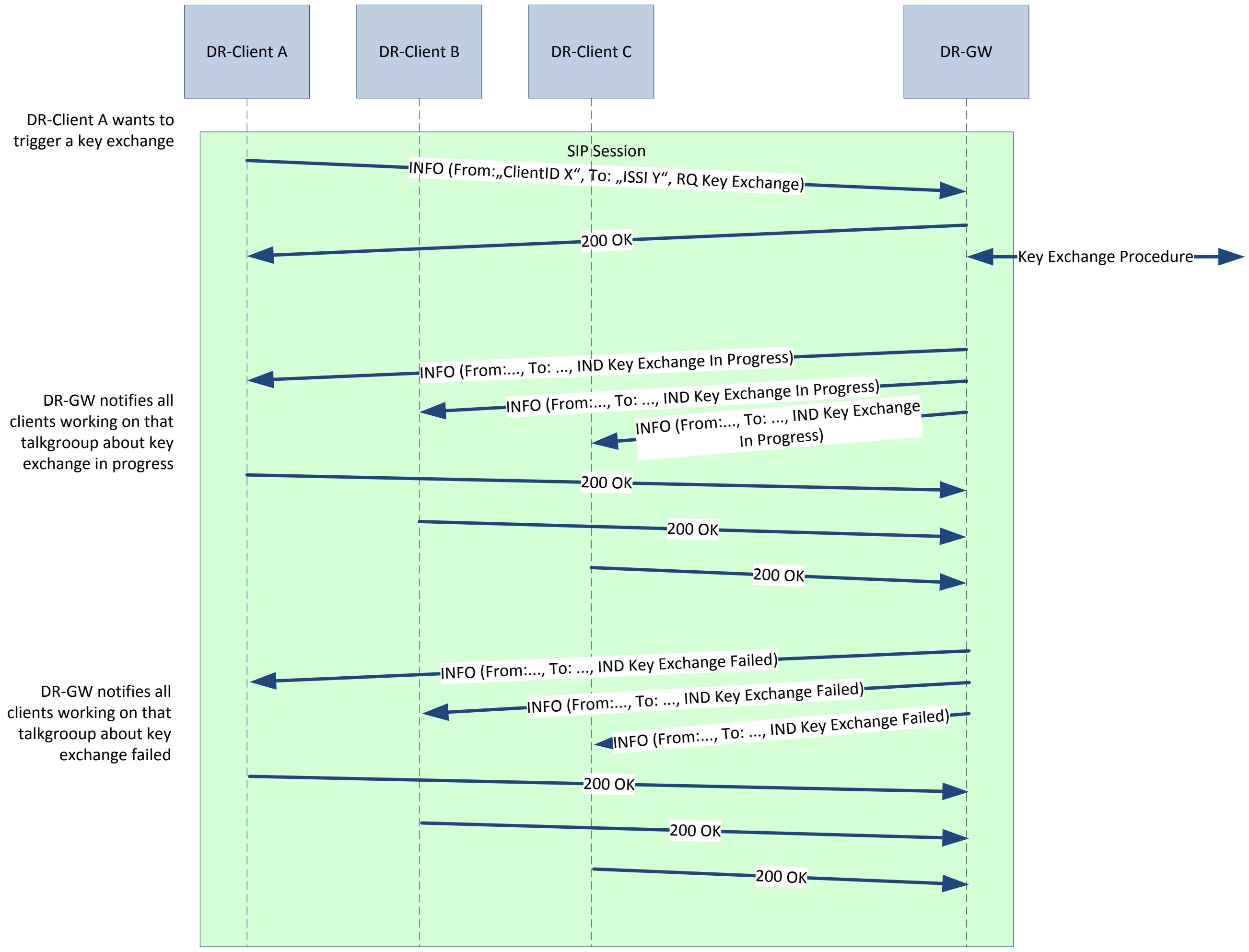

DR-Client A
DR-Client B
DR-Client C
DR-GW
DR-Client A wants to trigger a key exchange
SIP Session
INFO (From:„ClientID X“, To: „ISSI Y“, RQ Key Exchange)
200 OK
Key Exchange Procedure
INFO (From:..., To: ..., IND Key Exchange In Progress)
INFO (From:..., To: ..., IND Key Exchange In Progress)
INFO (From:..., To: ..., IND Key Exchange In Progress)
DR-GW notifies all clients working on that talkgrooup about key exchange in progress
200 OK
200 OK
200 OK
INFO (From:..., To: ..., IND Key Exchange Failed)
INFO (From:..., To: ..., IND Key Exchange Failed)
INFO (From:..., To: ..., IND Key Exchange Failed)
DR-GW notifies all clients working on that talkgrooup about key exchange failed
200 OK
200 OK
200 OK

Figure 24 – MSC for unsuccessful new key demand

As the progress of key exchange is reported to each client which is at least event monitoring the specific talkgroup, a client can abort an ongoing key exchange. Such a scenario is shown in Figure 25.

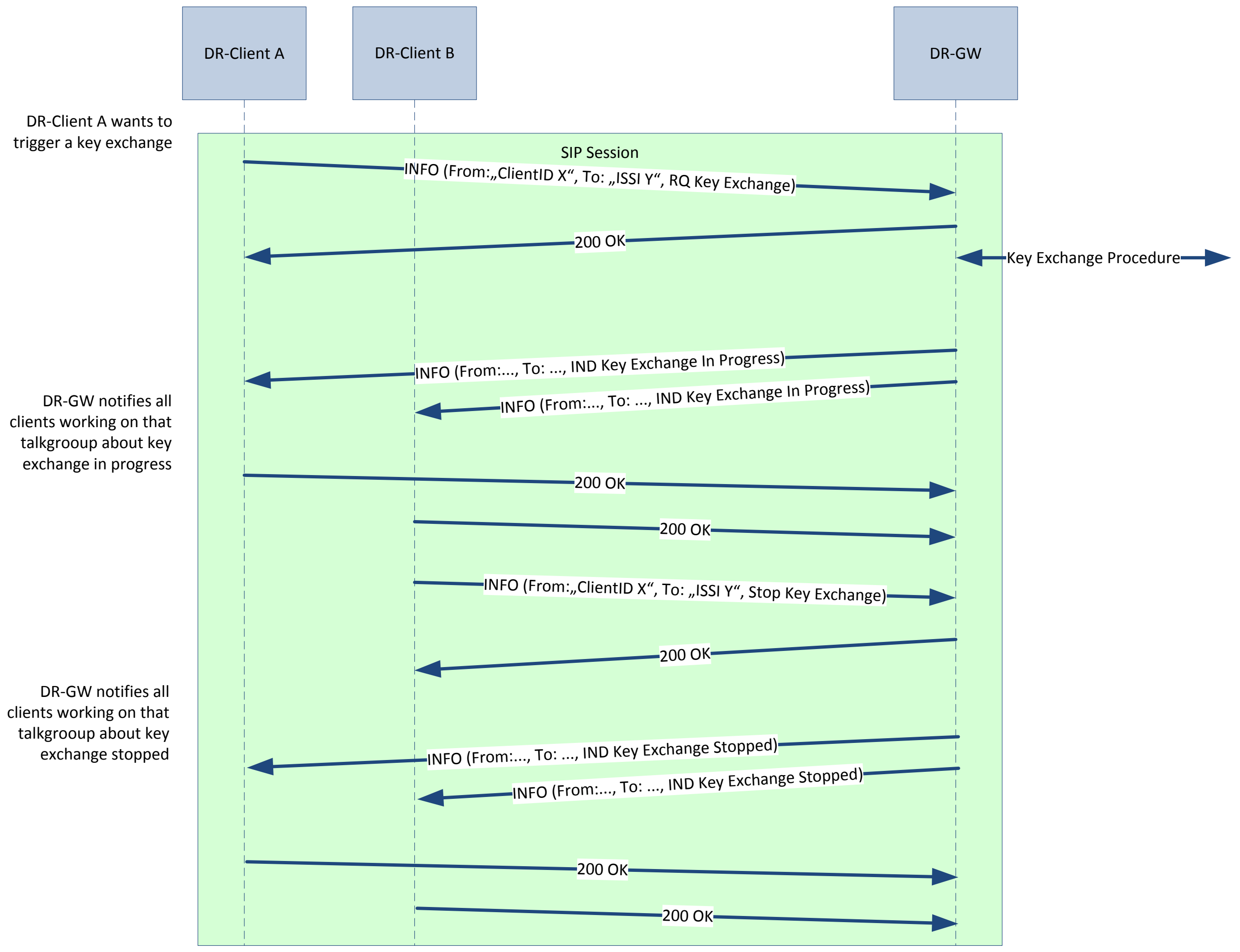

DR-Client A
DR-Client B
DR-GW
DR-Client A wants to trigger a key exchange
SIP Session
INFO (From:„ClientID X“, To: „ISSI Y“, RQ Key Exchange)
200 OK
Key Exchange Procedure
INFO (From:..., To: ..., IND Key Exchange In Progress)
INFO (From:..., To: ..., IND Key Exchange In Progress)
DR-GW notifies all clients working on that talkgrooup about key exchange in progress
200 OK
200 OK
INFO (From:„ClientID X“, To: „ISSI Y“, Stop Key Exchange)
200 OK
DR-GW notifies all clients working on that talkgrooup about key exchange stopped
INFO (From:..., To: ..., IND Key Exchange Stopped)
INFO (From:..., To: ..., IND Key Exchange Stopped)
200 OK
200 OK

Figure 25 – MSC for client aborting new key demand

It is a matter of the rights management and implementation specifics of the DR-GW whether another client than the initiator of the key exchange can abort the key exchange or not.

5.7 Call Hold

Even though it is obvious how the best practice for putting a SIP call on hold would look like, this version of the DR-GW Interface Specification does not yet include this feature. The detailed specification of this feature will be provided in the later version of this interface specification.

5.8 Call Transfer

Even though there are some existing practices for transferring a call in the SIP world, for detailed selection of the model (attended call transfer, unattended transfer, blind transfer and supervised transfer) additional clarification is needed. For this reason, the detailed specification of this feature will be provided in the later version of this interface specification.

5.9 Error Congestion

Within the SIP Protocol part of this interface, errors are handled by rejecting requests, tearing down SIP Sessions and closing other dialogues (e.g. REGISTER) mainly by the DR-GW. There is no need to re-invent the wheel and to make the root cause on how to analyze failures and mis-behaviors clear, there shall be a way to propagate TCS error without any need for remapping to the DR-Client.

This goal is achieved by putting the SIP Reason Header in the respective requests and responses. The same way as the SIP community propagates errors originated within a PSTN (public switched telephone network) the SIP Gateway is selected. Therefore the RFC3326 Reason Header Field definition will be extended according to the chapter “7.8.2 DR-GW-Reason Header“.

In case the error condition may not be handled transparently for the client by the DR-GW, the DR-GW shall tear down the affected SIP sessions and close all other affected dialogues. For redundancy switchover the DR-Client has to perform the necessary action at the backup DR-GW as described in the chapter “4.5.6 DR-GW Failover functionality”.

The SOAP Part will handle failures in a similar fashion. Established connections shall be torn down by the DR-GW. The DR-Client has to switch to the backup DR-GW.

5.10 Unit in Emergency

Unit in Emergency is a special procedure within the DR network resulting in the demand of propagating few events towards the DR-Client. Those event propagations are not atomic. Further description of this functionality is to be taken out of the documentation provided by the Digital Radio System.

When information (definition see chapter 10.18.5 “Event: Call_UnitInEmergency”) is provided by the DR-GW via SIP, SIP INFO method shall be used where the information itself is propagated via the message body.

The following message sequence diagram shows an usage example of this event during an emergency activation of a mobile (only relevant information is shown).

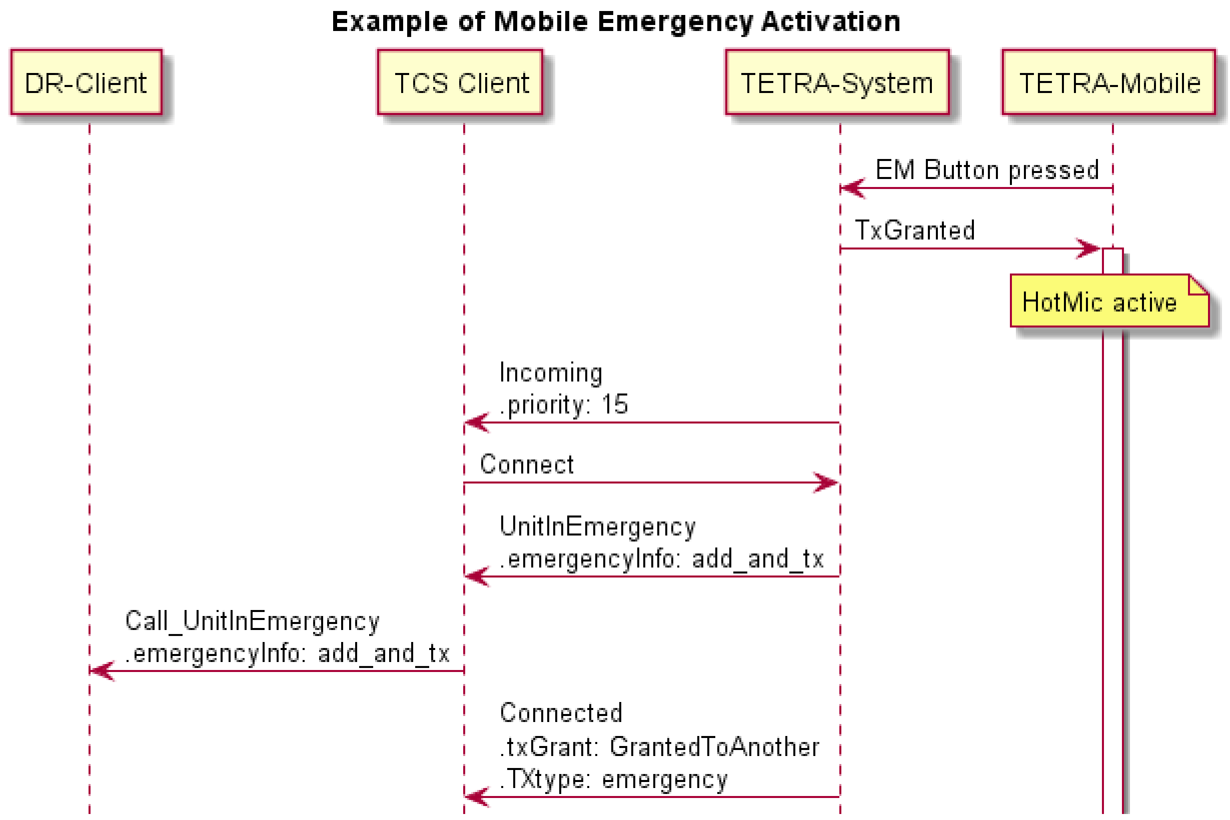


Figure 26 – MSC for Unit in Emergency

5.11 SDS Handling

SDS is handled via DR-GW Interface's SOAP part as described in chapter 10.4 "Request Interface: DR-GW-SDS" and 10.5 "Response and Event Interface: DR-GW-SDS.Events".

5.11.1 Send SDS

The DR-GW Client requests "SDS_Send" for sending an SDS-TL.

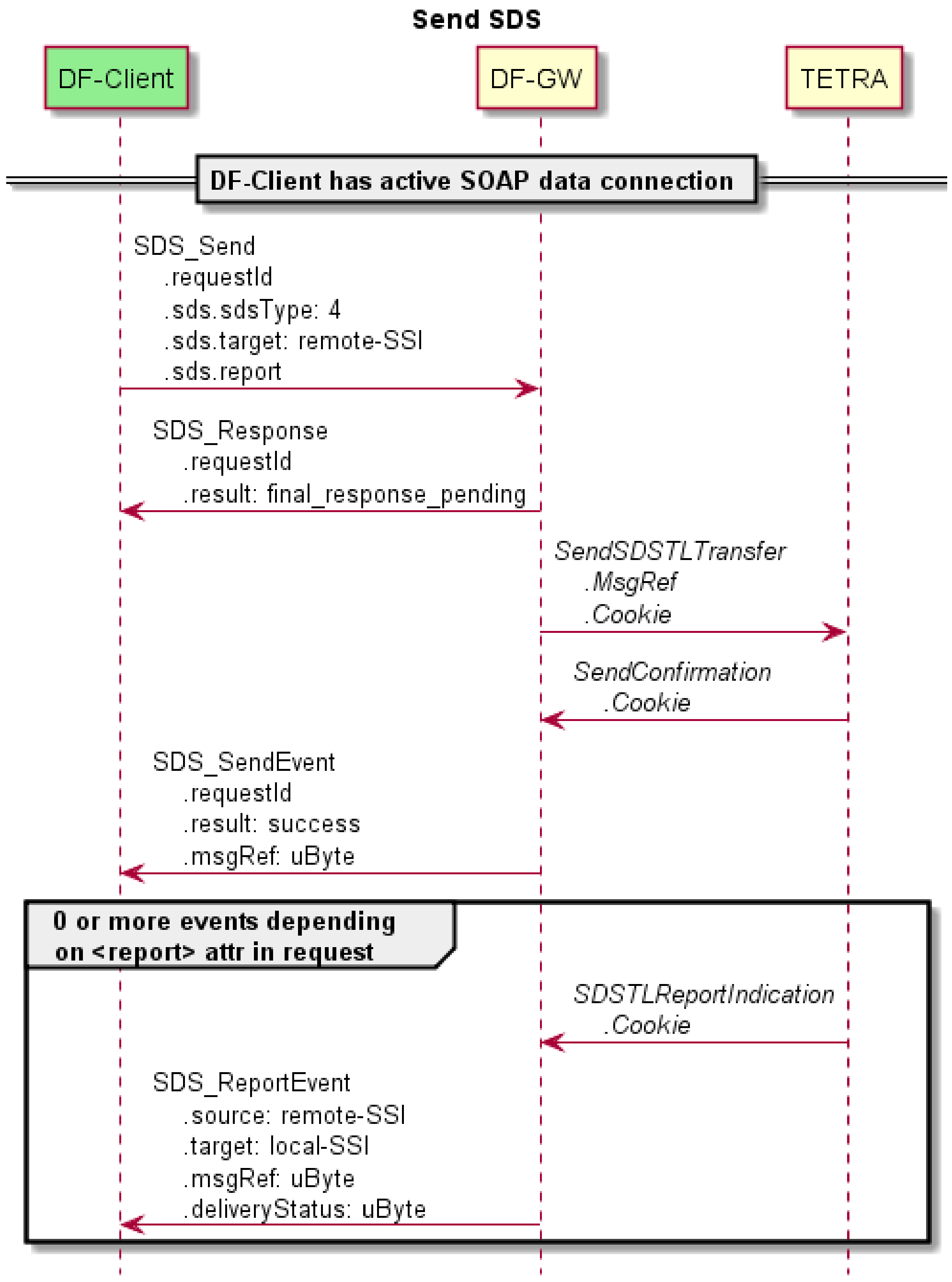


Figure 27 – MSC Send SDS

Messages SDS_Send, SDS_Response, SDS_SendEvent are correlated via the key „requestId“. Referencing for the related Events (SDS_SendEvent, SDS_ReportEvent) is given via the pair „remote-SSI“ and “msgRef”.

No SDS_ReportEvent is delivered if the SDS Receiver is a talkgroup (GSSI). Depending on the request parameter “report” when addressing an individual subscriber with the SDS, zero or several SDS_ReportEvents are provided.

5.11.1.1 Send Binary SDS

DR-Client may send arbitrary Protocol IDs (PIDs) and related user data supported by the TETRA network. For this purpose the client needs to binary encode all SDS data except PID, TL Header. Binary encoded SDS data have to be provided with the “hexdata” parameter.

If DF-Client requested encrypted transport the DR-GW encrypts the message before handing over to the TETRA system.

With this option DR-Client may send text messages (PIDs 2, 9, 130, 137 and 138) with other than the standard supported character encoding.

Anyhow DR-Client is responsible for providing proper attributes (e.g. length attributes) within hexdata.

5.11.1.2 Send Text Messages

As mentioned in the paragraph before, DR-Client may send text messages with any encoding by using binary SDS.

On the other hand, DR-Client can use the “data” parameter to send text messages with the standard encoding IEC 8859-15 Latin 9. In this use case encoding of the message is performed by the DR-Gateway. It is worth mentioning that it is in the responsibility of the DR-Client to use only characters supported by the specified character set.

In this scenario DR-Gateway will not perform any check of the message length but encodes the entire string as one single SDS.

The Protocol IDs supported here are at least 2, 9, 130 und 137.

5.11.1.3 Send Long Text Messages

Standard (PID 2 und 130) and flash (PID 9 und 137) text messages are sent as described in the previous chapter.

If DR-Client needs to send a message that exceeds the standard message’s length (140 characters), this needs special treatment. DR-Client needs to send one single SDS message provided in the “data” parameter with protocol ID 138 (instead of 2, 9, 130 or 137). DR-Gateway in this use case is responsible for splitting to several SDS (and encryption if requested by the client) towards TETRA system. Character encoding follows the same rules as for sending standard text messages.

If DR-Client has requested “delivered” or “consumed” report, DR-Gateway does not propagates this report indication as long as the TETRA system did not confirm all split SDS.

5.11.1.4 Typical error responses

Following error codes are only a subset of possible error messages and represent the most common situations during SDS sending.

Error	SDS_SendEvent

destination address unknown	responseCode = error
	sourceSystem =TCS-API
	result = 16777253

destination unreachable	responseCode = error
	sourceSystem =TCS-API
	result = 16777231

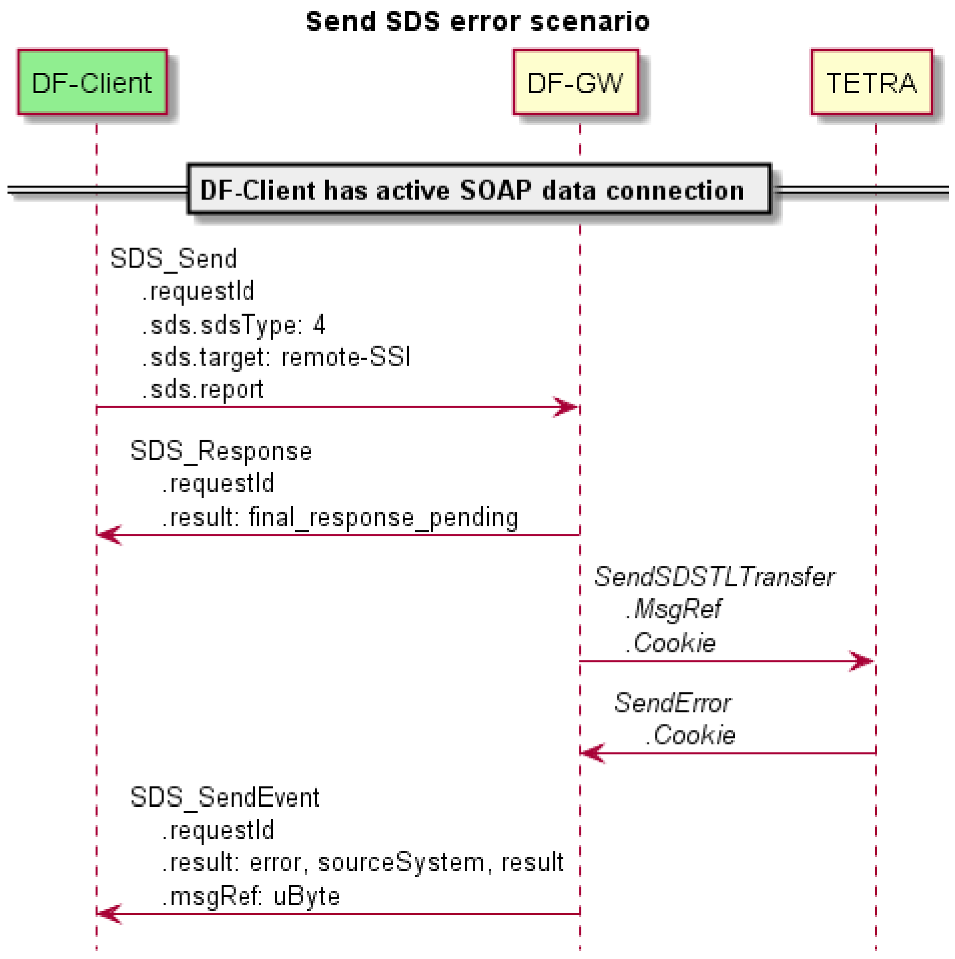


Figure 28 – MSC Send SDS error scenario

5.11.2 RECEIVE SDS

DR-Gateway applies “SDS_ReceiveEvent” to asynchronously notify DR-Client when SDS was received.

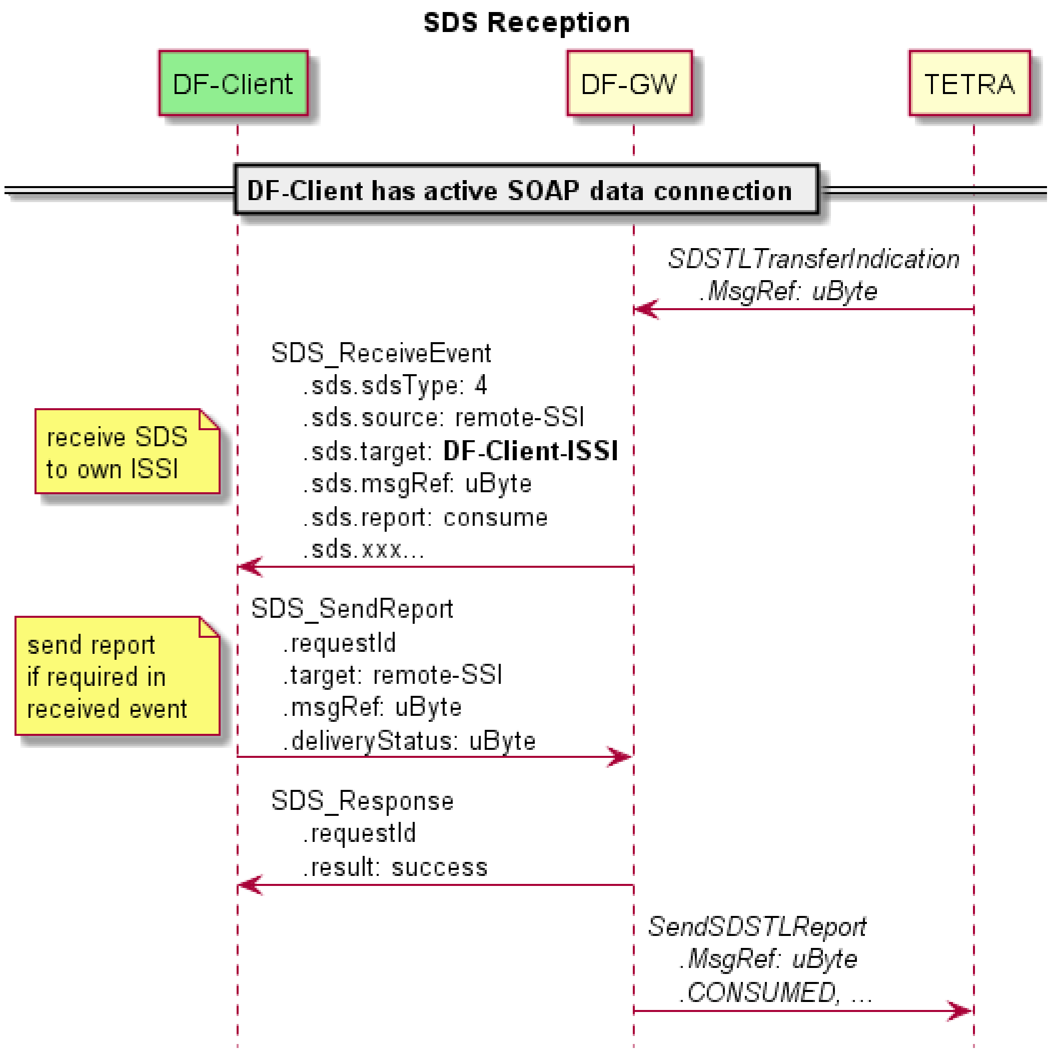


Figure 29 – MSC Receive SDS

The events SDS_ReceiveEvent and SDS_SendReport are related to each other with the pair formed of remote-SSI and msgRef.

DR-Client will receive SDS addressed to a talk group, it has registered for with “Group_SubscribeData”. SDS are indicated via SDS_ReceiveEvent as well, where no report is reported to the SDS originator.

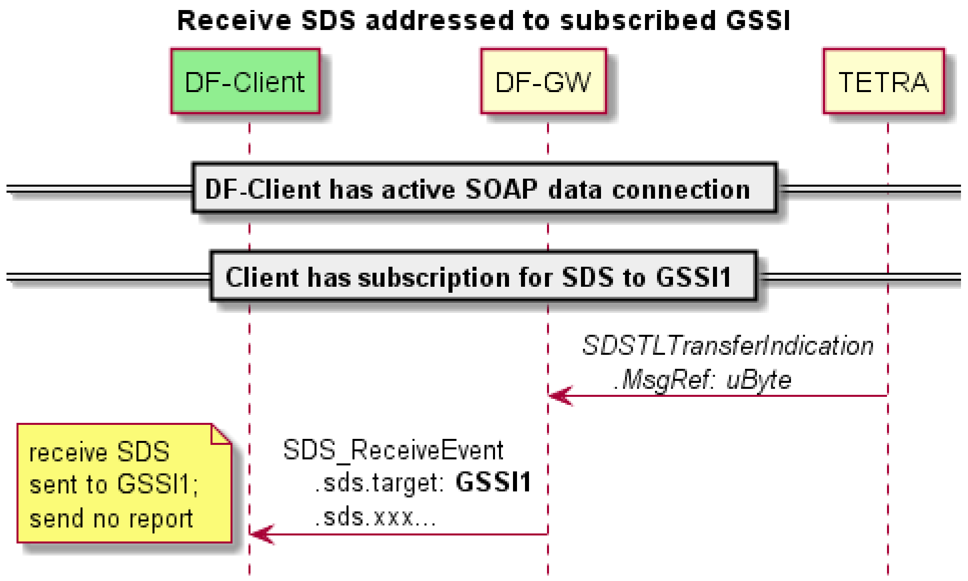

Receive SDS addressed to subscribed GSSI
DF-Client
DF-GW
TETRA
DF-Client has active SOAP data connection
Client has subscription for SDS to GSSI1
SDSTLTransferIndication
.MsgRef: uByte
SDS_ReceiveEvent
.sds.target: GSSI1
.sds.xxx...
receive SDS sent to GSSI1; send no report

Figure 30 – MSC Receive SDS addressed to subscribed GSSI

If SDS has been encrypted by the originator (independent of whether destination has been an ISSI or a GSSI), DR-GW takes care of decryption before forwarding to DR-Client. In this case DR-Client is notified, that an encrypted message has been received with the “Encryption” parameter. When decryption produces an error, the encrypted message is forwarded to DR-GW-Client with the error indication.

Parameter “hexdata” contains the SDS content in hexadecimal format. This parameter is propagated for any kind of SDS received.

Parameter “data” is populated **additionally** if a text message was received. Data encoding always follows the encoding of the xml body.

5.11.2.1 RECEIVE LONG TEXT MESSAGES

Standard (PID 2 und 130) and flash (PID 9 und 137) text messages are received as described in the previous chapter.

If DR-Gateway receives part of a split text message, the message is not forwarded immediately to DR-Client. Instead DR-Gateway collects all parts of the message and propagates one single text message with protocol ID 138.

5.12 RADIO/GROUP TRACKING

DR-Client can track Radio by applying Radio_Track (see chapter 10.15.4 “Request: Radio_Track”) and / or Group_Track (see chapter 10.10.5 “Request: Group_Track”) subscription respectively. Both of those subscriptions end up in Group_RadioMemberEvent (see chapter 10.11.7 “Event: Group_RadioMemberEvent”).

Caution: Based on observations performed in different scenarios and edge cases Radio

Tracking is the reliable method to discover the group a radio is active in.

Please note italic text as well as the actor “TETRA System” is for information only in all sequence charts within this chapter. It is upon the implementation of DR-GW to map TETRA System’s behavior to an appropriate DR-Gateway interface.

5.12.1 TRACKING VIA RADIO

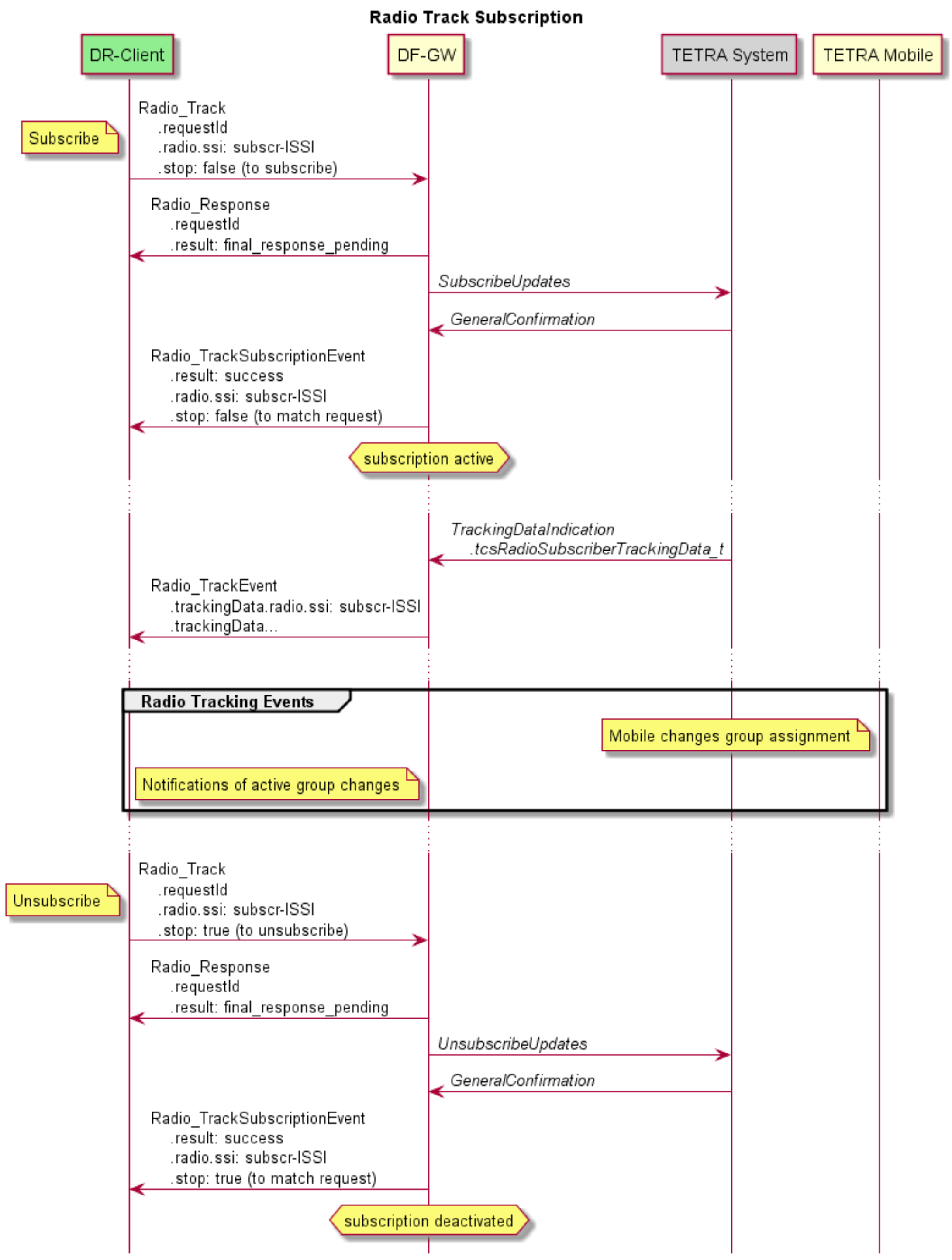

Radio Track Subscription
DR-Client
DF-GW
TETRA System
TETRA Mobile
Subscribe
Radio_Track .requestId .radio.ssi: subscr-ISSI .stop: false (to subscribe)
Radio_Response .requestId .result: final_response_pending
SubscribeUpdates
GeneralConfirmation
Radio_TrackSubscriptionEvent .result: success .radio.ssi: subscr-ISSI .stop: false (to match request)
subscription active
TrackingDataIndication .tcsRadioSubscriberTrackingData_t
Radio_TrackEvent .trackingData.radio.ssi: subscr-ISSI .trackingData...
Radio Tracking Events
Mobile changes group assignment
Notifications of active group changes
Unsubscribe
Radio_Track .requestId .radio.ssi: subscr-ISSI .stop: true (to unsubscribe)
Radio_Response .requestId .result: final_response_pending
UnsubscribeUpdates
GeneralConfirmation
Radio_TrackSubscriptionEvent .result: success .radio.ssi: subscr-ISSI .stop: true (to match request)
subscription deactivated

Figure 31 – Subscription for Radio Tracking

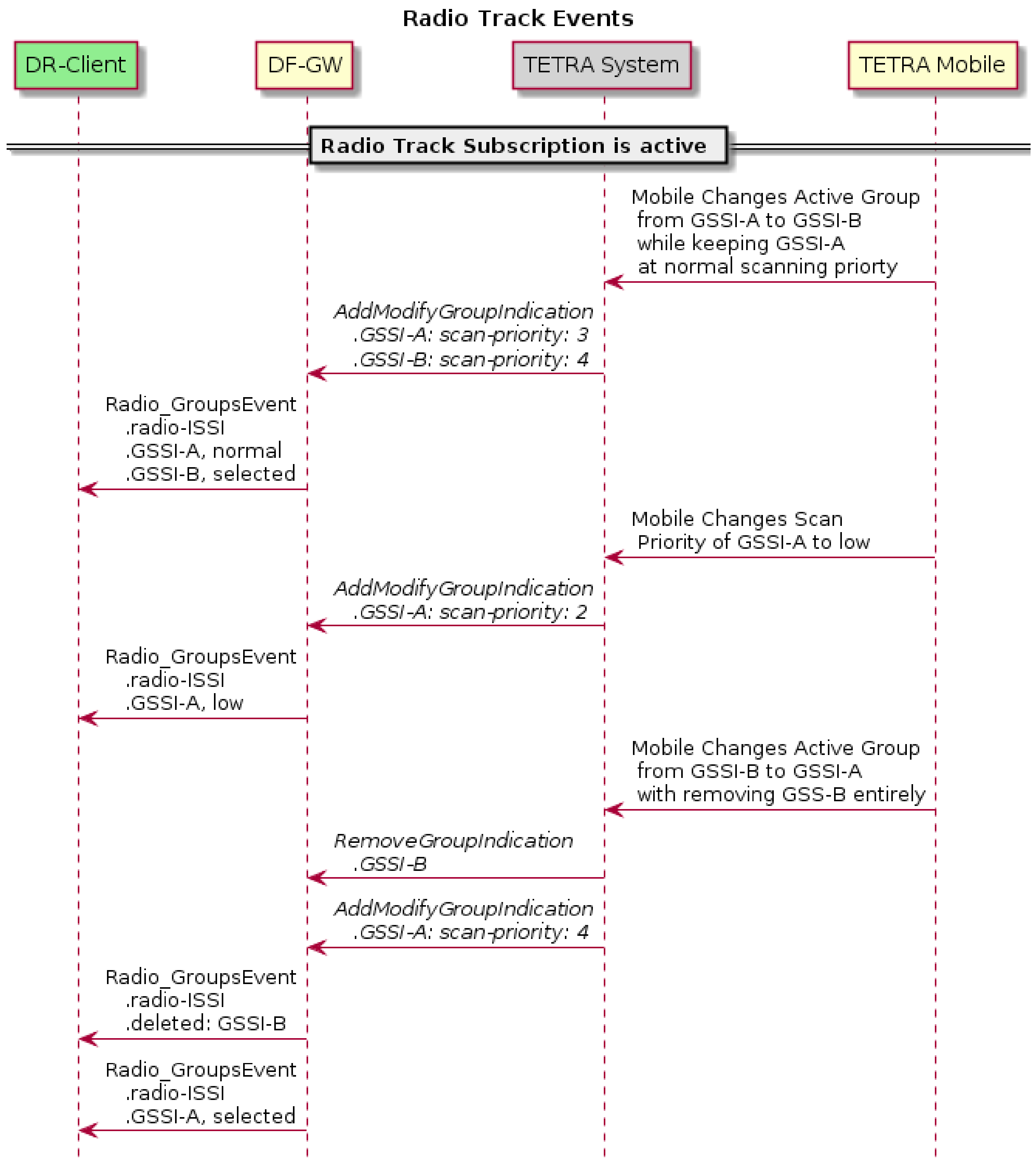

Radio Track Events
DR-Client
DF-GW
TETRA System
TETRA Mobile
Radio Track Subscription is active
Mobile Changes Active Group from GSSI-A to GSSI-B while keeping GSSI-A at normal scanning priorty
AddModifyGroupIndication .GSSI-A: scan-priority: 3 .GSSI-B: scan-priority: 4
Radio_GroupsEvent .radio-ISSI .GSSI-A, normal .GSSI-B, selected
Mobile Changes Scan Priority of GSSI-A to low
AddModifyGroupIndication .GSSI-A: scan-priority: 2
Radio_GroupsEvent .radio-ISSI .GSSI-A, low
Mobile Changes Active Group from GSSI-B to GSSI-A with removing GSS-B entirely
RemoveGroupIndication .GSSI-B
AddModifyGroupIndication .GSSI-A: scan-priority: 4
Radio_GroupsEvent .radio-ISSI .deleted: GSSI-B
Radio_GroupsEvent .radio-ISSI .GSSI-A, selected

Figure 32 – Radio Track Events

Caution: The order of the events (if there are multiple) is not specified.

It is the DR-Client’s task to track which is the active group of this mobile.

Based on implementation experience, TCS API fires initial AddModifyGroupIndication already with the subscription of radio tracking. That information is forwarded by the DR-Gateway as well. The behavior is not explicitly stated in TCS API and may change without any further notice.

5.12.2 Tracking via Group

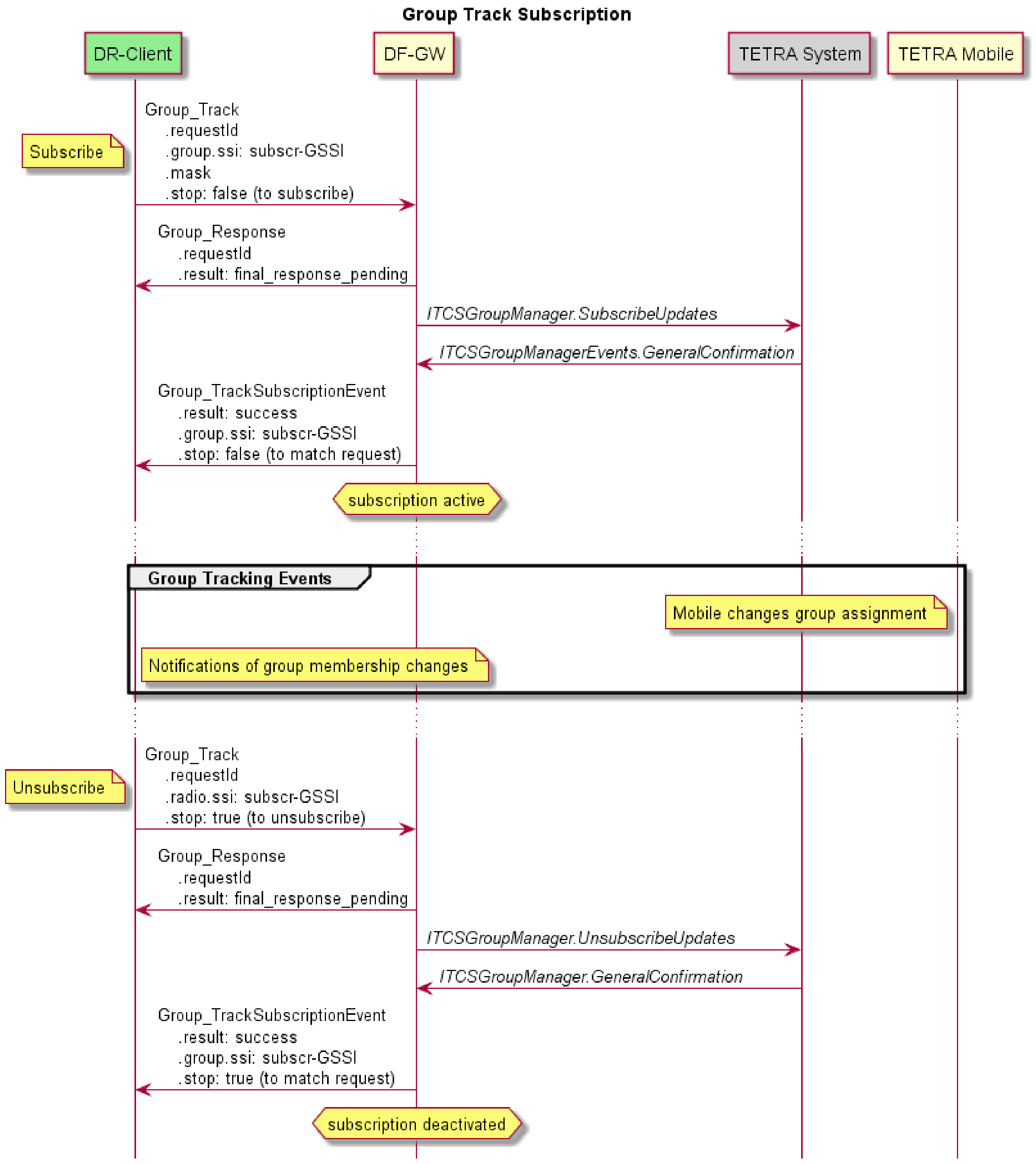

Group Track Subscription
DR-Client
DF-GW
TETRA System
TETRA Mobile
Subscribe
Group_Track
.requestId
.group.ssi: subscr-GSSI
.mask
.stop: false (to subscribe)
Group_Response
.requestId
.result: final_response_pending
ITCSGroupManager.SubscribeUpdates
ITCSGroupManagerEvents.GeneralConfirmation
Group_TrackSubscriptionEvent
.result: success
.group.ssi: subscr-GSSI
.stop: false (to match request)
subscription active
Group Tracking Events
Mobile changes group assignment
Notifications of group membership changes
Unsubscribe
Group_Track
.requestId
.radio.ssi: subscr-GSSI
.stop: true (to unsubscribe)
Group_Response
.requestId
.result: final_response_pending
ITCSGroupManager.UnsubscribeUpdates
ITCSGroupManager.GeneralConfirmation
Group_TrackSubscriptionEvent
.result: success
.group.ssi: subscr-GSSI
.stop: true (to match request)
subscription deactivated

Figure 33 – Subscription for Group Tracking

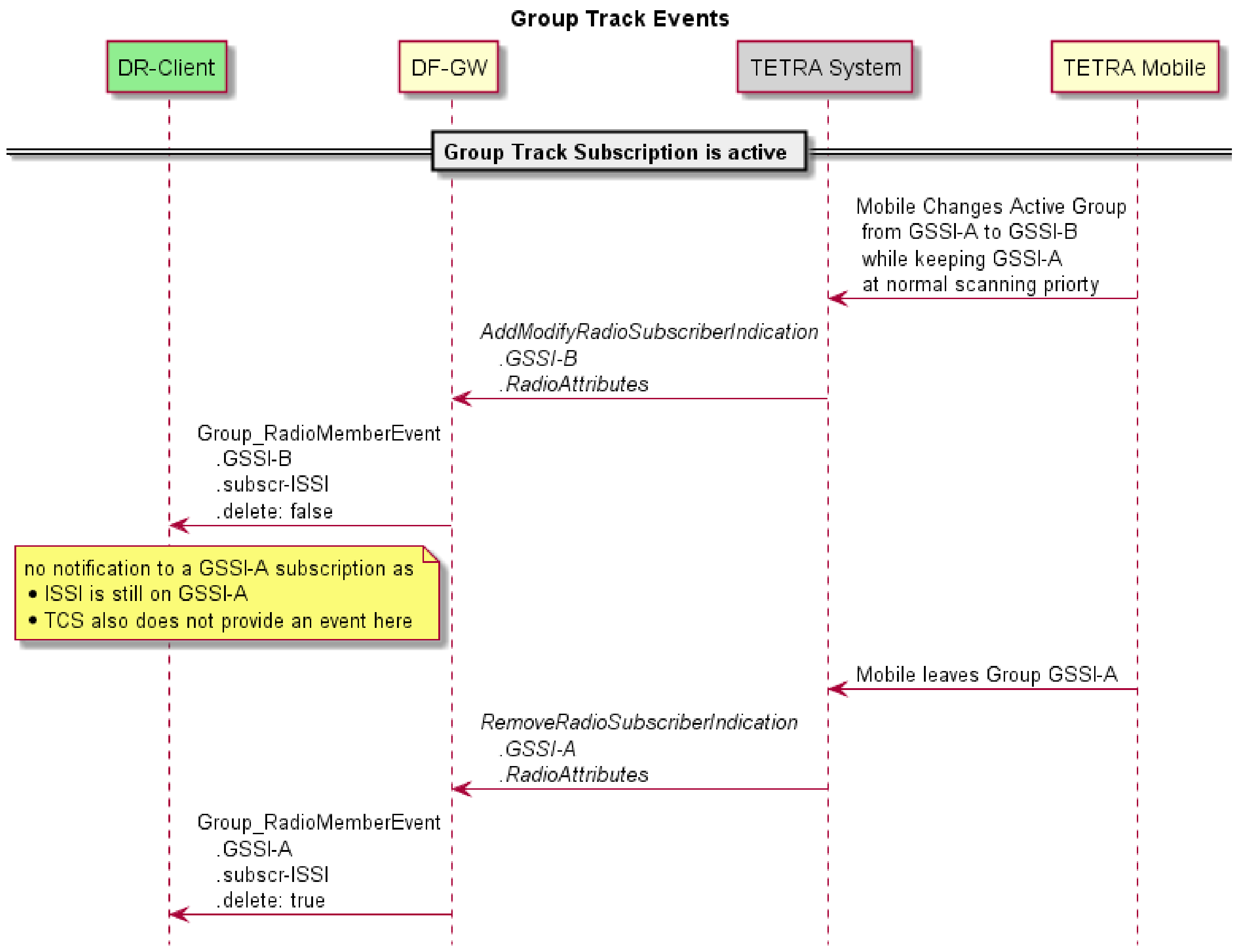

Group Track Events
DR-Client
DF-GW
TETRA System
TETRA Mobile
Group Track Subscription is active
Mobile Changes Active Group from GSSI-A to GSSI-B while keeping GSSI-A at normal scanning priorty
AddModifyRadioSubscriberIndication .GSSI-B .RadioAttributes
Group_RadioMemberEvent .GSSI-B .subscr-ISSI .delete: false
no notification to a GSSI-A subscription as • ISSI is still on GSSI-A • TCS also does not provide an event here
Mobile leaves Group GSSI-A
RemoveRadioSubscriberIndication .GSSI-A .RadioAttributes
Group_RadioMemberEvent .GSSI-A .subscr-ISSI .delete: true

Figure 34 – Group Track Events

5.12.3 Conclusion

As depicted in the chapters before, there are 2 use cases:

- Tracking via the Radio gives the option to discover on which group a particular Radio is active.
- Tracking via the Group gives the option to discover all Radios associated with a group independent of whether those radios are active on this group or not.

6 Security Aspects

The DR-GW-Interface as an interface based on IP will be run on infrastructure which is treated as “secure”. Therefore, neither Parkerian’s Hexad Model[2] nor the basic concept of CIA triad (confidentiality, integrity and availability) on application level has to be applied to provide required security aspects. The reason for security on the DR-GW-Interface’s application level, are needs of:

- Resource Management
- Audit Trail
- Accountability

While the first two items are mandatory and SHALL be implemented within the very first implementation, the latter is an optional feature which could be required by a specific customer. The basis for all these three features is authentication in a state of the art way where no credentials are propagated in a plain readable form over the network.

SIP protocol suite as being used by public telephone operators provides means for:

- Confidentiality (SIPS → SIP over TLS, SRTP with payload encryption)
- Authentication (SIP Authentication, SRTP with payload authentication)
- The SOAP protocol over either a HTTP or WebSocket transport provides means for:Confidentiality (HTTPS → HTTP over TLS)
- Server authentication (SSL certificates)
- Client authentication (digest authentication or SSL client certificates)

For the context of the DR-GW-Interface the authentication part of the control protocols SHALL be used which means:

- Each SIP dialogue established by the DR-GW-Client (REGISTER, INVITE) SHALL be authorized by the DR-GW by validating user name and password using MD5 digest authentication
- Each SIP dialogue established to the DR-GW-Client SHALL be accepted only when being originated from the DR-GW
- Each HTTP request originating from the DR-GW-Client SHALL be authorized by the DR-GW by validating user name and password using digest authentication [RFC 2617]. This includes the initial WebSockets requests.
- No confidentiality measures will be applied – no SIPS, no https, no SRTP

6.1 Encryption

It is recognized that there may be deployments with additional security requirements. For these deployments, many of the concerns regarding confidentiality and integrity can be addressed by using cryptography. Therefore, the following provisions are made:

- Optionally, when using SOAP over HTTP or WebSocket, the connection can be

[2] Parker, Donn B. (1998). Fighting Computer Crime. New York, NY: John Wiley & Sons. ISBN 0-471-16378-3. The work in which Donn B. Parker introduced this model.

encrypted with TLS (HTTPS).
- Optionally, SIPS can be employed (SIP over a connection encrypted with TLS).
- Optionally, SRTP can be employed.

To ascertain authenticity of the communication partners, SSL certificate validation can be used for the server. The client authenticates by using Digest Authentication. The validation process and associated issues like key management and distribution are beyond the scope of this document and should be specified separately on a case by case basis.

7 PROFILE STANDARD FOR THE USE OF SIP

7.1 SESSION INITIATION PROTOCOL

DR-GW and DR-GW-Client SHALL support SIP version 2 as specified in RFC 3261.

The SIP protocol is an application-layer protocol which has been developed and designed within the IETF and is defined by RFC 3261. With respect to TETRA Radio applications the SIP protocol SHALL be used by each Digitalradio-Gateway (DR-GW)-Client to establish, modify and terminate SIP sessions with a Digitalradio-Gateway (DR-GW).

Once a communication session between the DR-GW-Client and the DR-GW has been established using the SIP protocol, the two endpoints SHALL then employ the Real time Transport Protocol (RTP) (RFC 3550) communication. Once the RTP is active this communication SHALL be used for the transport of audio in RTP packets between the endpoints (as defined in RFC 3550).

The audio transport MAY be augmented by its associated control protocol (RTCP) (RFC 3550 [21]) to allow monitoring of voice packet delivery.

Each of the two parties engaged in a two way communication may use RTP packets for supervising the session. If a heartbeat timeout for supervision is applied the timeout value has to be set according to the ptime and the maxptime value of the negotiated SDP part of the SIP Messages.

7.2 TRANSPORT LAYER

According to RFC 3261 support of UDP for the transport protocol is mandatory. To avoid any problems caused by dynamical changed transport layer by:

- Misconfigured firewalls blocking any TCP traffic
- off-the-shelf SIP stacks are pretty well tested when the first message of a dialogue is the longest message (INVITE, REGISTER, SUBSCRIBE, NOTIFY). In contradiction to the use case of DR-Gateway where SIP INFO (as part of the INVITE dialogue) is expected to be one of the longest messages which is definitively an edge case maybe not best tested.

For this purpose it is **recommended** to:

- announce "transport=udp" parameter as part of the via and from URI (e.g. **"sip:alice@atlanta.com;transport=udp"**)
- consistently use UDP transport layer even if message size exceeds 1300 Byte

for all User Agent Clients. Future versions of this specification will add the transport parameter as a **mandatory** attribute. Both the DR-GW and the DR-Client are implementing the UAC role depending on the call scenario.

The last statement intentionally contradicts RFC 3261 page 141:

18.1.1 Sending Requests

If a request is within 200 bytes of the path MTU, or if it is larger than 1300 bytes and the path MTU is unknown, the request MUST be sent using an RFC 2914 [43] congestion controlled transport protocol, such as TCP….

On the other hand strictly follow RFC3261 requirement page 141:

18.1.1 Sending Requests

…However, implementations MUST be able to handle messages up to the maximum datagram packet size. For UDP, this size is 65,535 bytes, including IP and UDP headers….

Good indication that allows this approach is also given on RFC3261 page 191:

19.1.1 SIP and SIPS URI Components

…

The transport parameter determines the transport mechanism to be used for sending SIP messages, as specified in [4]. SIP can use any network transport protocol. Parameter names are defined for UDP (RFC 768 [14]), TCP (RFC 761 [15]), and SCTP (RFC 2960 [16])….

Where "… as specified in [4]…" directs to RFC 3263 (Locating SIP Servers) to page 5:

4.1 Selecting a Transport Protocol

First, the client selects a transport protocol.

If the URI specifies a transport protocol in the transport parameter, that transport protocol SHOULD be used.

Otherwise, if no transport protocol is specified, but the TARGET is a numeric IP address, the client SHOULD use UDP for a SIP URI, and TCP for a SIPS URI. Similarly, if no transport protocol is specified, and the TARGET is not numeric, but an explicit port is provided, the client SHOULD use UDP for a SIP URI, and TCP for a SIPS URI. This is because UDP is the only mandatory transport in RFC 2543 [6], and thus the only one guaranteed to be interoperable for a SIP URI. It was also specified as the default transport in RFC 2543 when no transport was present in the SIP URI. However, another transport, such as TCP, MAY be used if the guidelines of SIP mandate it for this particular request. That is the case, for example, for requests that exceed the path MTU.

…

Those sections give enough justification for the procedure specified here.

7.3 LOGICAL SIP ENTITIES

The DR-GW-Client with respect to TETRA Radio applications is acting as SIP User Agent. The DR-GW in the same context is acting as SIP Registrar, SIP Proxy Server and SIP User Agent. Both entities (DR-GW, DR-GW-Client) acting as SIP User Agent have to implement both roles the UAC (User Agent Client) as well as UAS (User Agent Server).

7.3.1 User Agents

User Agents in a TETRA Radio environment SHALL support the following services and procedures:

7.3.1.1 Registration

Registration Discovery in the one and only variant “configured registrar address” according to RFC 3261, section 10.2.6

Adding Bindings according to RFC 3261, section 10.2.1

Removing Bindings according to RFC 3261, section 10.2.2

Refreshing Bindings according to RFC 3261, section 10.2.4

Ordering Contacts according to RFC 3261, section 10.2.1.2

7.3.1.2 Call Control

7.3.1.2.1 Establishing a session

UAC procedures according to RFC 3261, section 13.2

UAS procedures according to RFC 3261, section 13.3

Based on Location server messages procedures according to RFC 3261, section 8.1.3.4

7.3.1.3 Terminating a session with BYE

UAC procedures according to RFC 3261, section 15.1.1

UAS procedures according to RFC 3261, section 15.1.2

7.3.1.4 Cancelling a session

UAC procedures according to RFC 3261, section 9.1

UAS procedures according to RFC 3261, section 9.2

7.3.2 Registrar

DR-GW acting as SIP Registrar in a TETRA Radio environment SHALL support the following services and procedures:

7.3.2.1 Registration

Maintaining Bindings according to RFC 3261, section 10.3

Ordering contacts according to RFC 3261, section 10.2.1.2

Unicast Registration according to RFC 3261, section 10.3

7.3.3 Proxy Server

DR-GW acting as SIP Proxy Server in a TETRA Radio environment SHALL support the following services and procedures:

7.3.3.1 Call Control

7.3.3.1.1 Establishment a session

Stateful procedures according to RFC 3261, sections 16 and 8.2

Based on Location server messages according to RFC 3261, sections 16 and 8.2

7.3.3.1.2 Terminating a session with BYE

Stateful procedures according to RFC 3261, sections 16 and 8.2

7.3.3.1.3 Cancelling a session

Stateful procedures according to RFC 3261, sections 16 and 8.2

7.4 Supported Requests

Method	DR-GW-Client		DR-GW-Server	
	Sending	Receiving	Sending	Receiving
INVITE	m	m	m	m
ACK	m	m	m	m
CANCEL	m	m	m	m
BYE	m	m	m	m
REGISTER	o	-	x	m
INFO	o	o	m	m
SUBSCRIBE	o	o	o	o
NOTIFY	o	o	o	o
UPDATE	o	o	o	m
OPTIONS	o	o	o	m
REFER	o	o	x	o
MESSAGE	o	o	o	o
PUBLISH	o	o	o	o
PRACK	o	m	o	m

“m”: mandatory; “o”: optional; “x”: prohibited; “-“: not applicable

The requirements of RFC 2976 (SIP INFO), RFC 3261 (SIP 2), RFC 3262 (Provisional Responses), RFC 3311 (SIP UPDATE) and RFC 3515 (SIP REFER) and RFC 4028 (SIP Session Timers) SHALL apply.

7.4.1 BYE

In case of group Communication the DR-GW SHALL terminate a SIP session with BYE only if error occurs.

7.4.2 INFO

SIP INFO is used for following purposes:

- arbitration of the floor in case of group communication and half duplex individual call
- indicating and control of TETRA Call which has been established and torn down
- trigger and notification of new TETRA end 2 end encryption key exchange

7.4.3 OPTIONS

A DR-Client may use out of dialog OPTION requests to query the general availability of the SIP path to and the DF-GW itself.

7.5 Supported Responses

Response	**DR-GW-Client**		**DR-GW-Server**	
	Sending	**Receiving**	**Sending**	**Receiving**
100 – Trying	o	m	o	m
180 – Ringing	o	m	o	m
181 – Call Is Being Forwarded	-	m	o	-
182– Queued	-	m	x	-
183 – Session Progress	o	m	o	m
200 – Ok	m	m	m	m
202 – Accepted	o	o	o	o
300 – Multiple Choices	x	-	x	-
301 – Moved Permanently	o	m	o	m
302 – Moved Temporarely	o	m	o	m
400 – Bad Request	-	m	m	-
401 – Unauthorized	x	m	o	-
404 – Not Found	o	m	o	m
405, 406	-	m	m	-

Response	DR-GW-Client		DR-GW-Server	
	Sending	Receiving	Sending	Receiving
407 – Proxy Authentication Required	x	m	o	-
408, 410,413, 414	-	m	m	-
415 – Unsupported Media Type	-	m	m	-
416 – Unsupported URI Scheme	-	m	m	-
420,421, 423		m	o	m
422 – Session Interval To Small	m	m	m	m
480 – Temporarily Unavailable	-	m	m	-
481 -485, 487, 489	-	m	o	-
486 – Busy Here	o	m	m	m
488 – Not Acceptable Here	m	m	m	m
491 – Request Pending	-	m	m	-
493 – Undecipherable	-	m	m	-
501 – Request Not Supported	m	m	m	m
502 -505, 513	-	m	o	-
603 – Decline	m	m	m	m
604, 606	o	m	o	m

“m”: mandatory; “o”: optional; “x”: prohibited; “-“: not applicable

7.6 Supported SIP Header Fields

Complete and sufficient description of mandatory and optional Header Fields.

For all the tables within this chapter entries are abbreviated:

o	optional
m	mandatory
m*	The header field SHOULD be sent, but clients/servers need to be prepared to receive messages without that header field.
c	Conditional; requirements on the header field depend on the context of the message.
*	The header field is required if the message body is not empty.
-	The header field is not applicable.

7.6.1 User Agent Request Headers

UA Request Header Field		**Requests**										
	ACK	**BYE**	**CAN**	**INF**	**INV**	**MES**	**NOT**	**OPT**	**REF**	**REG**	**SUB**	**UPD**
Allow	---	o	---	o	o	o	o	o	o	o	o	o
Allow-Events (RFC 3265)	o	o	---	---	o	---	o	o	---	o	o	---
Authorization	o	o	o	o	o	o	o	o	o	o	o	o
Call-ID	m	m	m	m	m	m	m	m	m	m	m	m
Contact	o	---	---	---	m	---	m	o	m	o	m	m
Content-Length	m	m	m	m	m	m	m	m	o	m	m	m
Content-Type	*	*	---	m	*	*	*	*	*	*	*	*
Cseq	m	m	m	m	m	m	m	m	m	m	m	m
Date	o	o	o	o	o	o	o	o	o	o	o	o
Event (RFC 3265)	---	---	---	---	---	---	m	---	---	---	m	---
Expires	---	---	---	---	o	o	---	---	o	o	o	---
From	m	m	m	m	m	m	m	m	m	m	m	m

UA Request Header Field		Requests										
	ACK	BYE	CAN	INF	INV	MES	NOT	OPT	REF	REG	SUB	UPD
In-Reply-to	---	---	---	---	o	o	---	---	---	---	---	---
Join (RFC 3911)	---	---	---	---	o	---	---	---	---	---	---	---
Max-Forwards	m	m	m	o	m	m	m	m	m	m	m	m
MIME-Version	o	o	---	o	o	---	o	o	o	o	o	o
Priority	---	---	---	---	m	o	---	---	---	---	o	---
Proxy-Authorization	o	o	---	o	o	o	o	o	o	o	o	o
Proxy-Require	---	o	---	o	o	o	o	o	o	o	o	o
Record-Route	o	o	o	o	o	---	o	o	o	---	o	o
Reason	---	o	o	---	---	---	---	o	---	---	--	---
Refer-To (RFC 3515)	---	---	---	---	---	---	---	---	o	---	---	---
Replaces (RFC 3891)	---	---	---	---	o	---	---	---	---	---	---	---
Reply-to	---	---	---	---	o	o	---	---	---	---	---	---
Require	---	c	---	o	c	c	o	c	c	c	o	c
Route	c	c	c	o	c	o	c	c	c	c	c	c
Subject	---	---	---	o	m	o	---	---	---	---	---	---
Subscription-State (RFC 3265)	---	---	---	---	---	---	m	---	---	---	---	---
Supported	---	o	o	o	m*	---	o	o	o	o	o	o
To	m	m	m	m	m	m	m	m	m	m	m	m
Via	m	m	m	m	m	m	m	m	m	m	m	m
DR-GW-Version	o	o	o	m	m	m	o	m	m	m	m	o

7.6.2 User Agent Response Headers

UA Response Header Field	Status Code	Requests											
		ACK	BYE	CAN	INF	INV	MES	NOT	OPT	REF	REG	SUB	UPD
Allow	2xx	---	o	---	o	m	o	o	m*	---	o	o	o
Allow	405	---	m	---	m	m	m	m	m*	m	m	m	m
Allow	All except 2xx, 405	---	o	---	o	o	o	o	o	o	o	o	o
Allow-Events (RFC 3265)	2xx	o	o	---	---	o	---	o	o	---	o	o	---
Allow-Events (RFC 3265)	489	---	---	---	---	---	---	m	---	---	---	m	---
Authentication-Info	2xx	---	o	---	o	o	o	o	o	o	o	o	o
Call-ID	All	m	m	m	m	m	m	m	m	m	m	m	m
Contact	1xx	---	---	---	---	o	---	o	---	---	---	o	o
Contact	2xx	---	---	---	---	m	---	o	o	m	o	m	m
Contact	3xx	---	o	---	---	o	o	m	o	---	o	m	o
Contact	485	---	o	---	---	o	o	o	o	o	o	o	o
Content-Length	All	m	m	m	o	m	m	m	m	o	m	m	m
Content-Type	All	*	*	---	*	*	*	*	*	*	*	*	*
Cseq	All	m	m	m	m	m	m	m	m	m	m	m	m
Date	All	o	o	o	o	o	o	o	o	o	o	o	o
Expires	2xx	---	---	---	---	o	---	---	---	---	o	---	---
From	All	m	m	m	m	m	m	m	m	m	m	m	m
MIME-Version	All	o	o	---	o	o	---	o	o	o	o	o	o

UA Response Header Field	Status Code	Requests											
		ACK	BYE	CAN	INF	INV	MES	NOT	OPT	REF	REG	SUB	UPD
Min-Expires	423	---	---	---	--	---	---	---	---	---	m	m	---
Proxy-Authenticate	407	---	m	---	m	m	m	m	m	m	m	m	m
Proxy-Authenticate	401	---	o	o	o	o	o	---	o	o	o	---	o
Reason	3xx, 4xx, 6xx	---	---	---	---	o	---	--	o			o	
Record-Route	2xx, 18x	o	o	o	o	o	---	---	o	o	---	---	o
Record-Route	401, 484	---	---	---	o	---	---	o	---	---	---	o	---
Reply-To	All	---	---	---	---	o	o	---	---	---	---	---	---
Require	All	---	c	---	o	c	c	o	c	c	c	o	c
Supported	2xx	---	o	o	---	m*	---	o	m*	o	o	o	o
To	All	m	m	m	m	m	m	m	m	m	m	m	m
Unsupported	420	---	m	---	o	m	o	o	m	o	m	o	m
Via	All	m	m	m	m	m	m	m	m	m	m	m	m
Warning	All	---	o	o	o	o	o	o	o	o	o	o	o
WWW-Authenticate	401	---	m	---	m	m	m	m	m	m	m	m	m
WWW-Authenticate	407	---	o	---	o	o	o	---	o	o	o	---	o
DR-GW-Version	2xx	---	o	o	o	m	m	o	o	m	o	o	o
DR-GW-Version	All except 2xx	---	o	o	o	o	o	o	o	o	o	o	o

7.7 SIP Header Usage

7.7.1 Content-Type

When the DR-GW protocol is transported via SIP, the SIP header Content-Type must explicitly describe it.

- if DR-GW only, the content type header is encoded as:

```
Content-Type: application/DR-GW+xml
Content-Length: XXX
```

The involved parties (DR-Client and/or DR-GW) have to announce possibility to accept DR-GW+xml in the according accept header. If not announced the remote peer has to assume application/sdp is supported only as defined in RFC 3261.

- in case a protocol message is to be transported together with SDP, multipart mime according RFC 2046 ("Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types") chapter 5.1.1 has to be used for content type encoding

```
Content-Type: multipart/mixed;boundary=<any characters used for boundary>
Content-Length: XXX

--<any characters used for boundary>
Content-Type: application/sdp
…SDP content…
--<any characters used for boundary>
Content-Type: application/DR-GW+xml
…DR-GW content…
```

The involved parties (DR-Client and/or DR-GW) have to announce possibility to accept multipart mime content in the according accept header. If not announced the remote peer has to assume application/sdp is supported only as defined in RFC 3261.

- in case of sdp only is propagated content type is encoded:

```
Content-Type: application/sdp
Content-Length: XXX
```

In the latter case if the DR-Client is using SDP content type only and indicating no support for multipart mime in the accept header of the INVITE (for the 200 OK final response) or in the REGISTER (for a DR-Client incoming individual call via INVITE) the DR-GW SHALL send SDP only message body. This behaviour ensures compatibility with RFC 3261 and shall enable the implementation of simple voice only clients with limited capability.

DR-GW SHALL support multipart mime as well as SDP only to allow maximum flexibility for the DR-Clients.

7.7.2 SIP FROM HEADER

The Display Name part of the SIP From header is used to indicate the assigned ISSI when SIP Call is established by the DR-Client.

When the DR-Client requests information about the assigned ISSI it SHALL formulate the from URI in the SIP INVITE request according following syntax:

```
From: ”ISSI=” <sip:someone@somewhere.org>
```

The DR-Gateway publishes the assigned ISSI in the SIP FROM Header field with in the according SIP Response:

```
From: ”ISSI=ISSI Indicator” <sip:someone@somewhere.org>
```

Even if a SIP FROM Header may not be changed within a dialogue there is the exception specified in chapter 20.20 of RFC 3261:

```
Two From header fields are equivalent if their URIs match, and their
parameters match. Extension parameters in one header field, not present in the
other are ignored for the purposes of comparison. This means that the display
name and presence or absence of angle brackets do not affect matching.
```

This feature is optional both for the DR-Client as well as for DR-Gateway.

7.7.3 SIP TO HEADER

The Display Name part of the SIP To header is used to indicate the assigned ISSI in the direction DR-Gateway to DR-Client in two scenarios:

- DR-Gateway establishes a SIP Call to the DR-Client (e.g. individual call)
- DR-Gateway positively replies to a REGISTER request.

Syntactical representation of the SIP To Header is the same as described for the SIP From Header in chapter 7.7.2.

This feature is optional both for the DR-Client as well as for DR-Gateway.

7.8 DEFINITION OF SIP HEADER FIELDS

7.8.1 DR-GW-VERSION

The DR-GW -Version header field SHALL appear in the requests and responses according to 7.6.1 respectively 7.6.2. Whenever the User Agent does not provide the version header field where it is mandatory, the behaviour may be undefined (receiving entity may reject the request or may continue processing).

The syntax of the header field follows the standard SIP parameter syntax as defined in RFC 3261 and SHALL have the following content:

```
DR-GW-Version = “DR_GW-Version” HCOLON version-value *(COMMA version-value)
version-value = field-value
field-value = type “.” number [“.” number]
```

```
type = “radio” / “phone”
number = DIGIT
```

The field-value SHALL contain the latest document version that reflects the implemented version of the corresponding interface specification as listed in the section ”4.3.2. SIP Interface”. Implementations MUST be able to process multiple header field rows with the same name in any combination of the single-value-per-line or comma-separated value forms. Respective actions may be specified where applicable.

Example:

```
DR-GW-Version: radio.01
DR-GW-Version: radio.1.1
```

NOTE:Version-params may be used for future extensions and are not yet described further in this document.

Defined SIP Version Values:

DR-GW Interface	SIP Header version value
Version 1.0	radio.01
Version 1.0.1	radio.01
Version 1.1.0	radio.1.1

When a UAC sends a request containing a version information the UAS cannot deal with, the UAS shall reject the request by responding with a “406 Not Acceptable”. In this case the response shall contain the DR-GW-Version header the UAS supports.

7.8.2 DR-GW-Reason Header

The DR-GW Reason Header may appear in any 3xx, 4xx, 5xx 6xx response answering a SIP INVITE. Additionally the DR-GW Reason Header may be sent in any request except for a SIP INFO request.

RFC 3326 will be extended in the way (see RFC 3326 “2. The Reason Header Field”):

```
protocol =  "SIP" / "Q.850" / “DR-GW” / “TCS-API” / “TETRA”
```

Definition of Error Values for protocol = “TETRA”:

Cause	Text
0	Not defined or unknown

Cause	**Text**
1	User requested disconnect
2	Called party busy
3	Called party not reachable
4	Called party does not support encryption
5	Congestion in infrastructure
6	Not allowed traffic case
7	Incompatible traffic case
8	Requested service not available
9	Pre-emptive use of resource
10	Invalid call identifier
11	Call rejected by the called party
12	No idle cc entity
13	Expiry of timer
14	SwMI requested disconnection
15	Acknowledged service not completed
16	Unknown tetra identity
17	SS specific disconnection
18	Unknown external subscriber identity
19	Call restoration of the other user failed
20	Called party requires encryption
21	Concurrent setup not supported
22	Called party is under the same DM Gate of the calling party

(Ref: ETSI EN 300 392-2, chapt. 14.8.18)

Examples at the end should look like:

```
Reason: “TETRA” ;cause=3 ;text="Called party not reachable"
```

Error Values for protocol = "TCS-API" are defined in the corresponding TCS-API documentation.

Error Values for protocol = "DF-GW" are defined by the corresponding vendor. To not have reason values used by multiple vendors, each vendor has an assigned reason values range as specified in the next table. The range from 0 to 999 is reserved for future use by this profile standard.

Cause Range	Vendor
1000 to 9999	Frequentis AG
10000 to 19999	Hagedorn Informationssysteme GmbH / T-Systems GmbH
20000 to 29999	Vendor 3
...	...

7.9 MESSAGE BODY

7.9.1 SDP MESSAGE BODY

The SDP Message body is encoded according to RFC4566. Deviations, interpretations and extensions to that standard are stated within these chapters.

7.9.1.1 SELECTION LEVEL

In order to separate technical and operational connection demand the selection level is propagated either via the DR-GW-Interface xml Body or via a SDP attribute as described herein. In this particular case it is possible to establish silence RTP streams, even in case of audio Monitoring only. The purpose is to „technically monitor" the other party which is still alive (as an option) in order to immediately release resources in case of crashed peer.

Attributes ("a=")	type:<call type>	Radio-Idle Radio-Rxonly Radio-TxRx or Radio (default value) Private Call
	duplex <duplex type>	full half
	encryption <encryption type>	true false
	OPTA <opta string>	string

7.9.1.2 CODEC PRIMITIVES

In general the DR-GW-Interface supports 3 different Codec types. The G.711, ACELP and the FSTE-OSTE Codec.

G.711 is well defined in the RFC4566. The utilization of ACELP in the RTP Payload and the respective SDP parameters are defined in the document “Real-Time Transport Protocol (RTP) Payload Format for the TETRA Audio Codec”. It has to be mentioned that the RTP packets contain exactly one ACELP packet – the packetization on the UDP layer matches with the one on the codec layer. This fact is well described in the document above.

No static RTP payload type has been assigned to ACELP TETRA Audio Codec, which means that it has to use a dynamic payload type number according chapter 6 in RFC4566. For RTP/AVP and RTP/SAVP, the range of dynamic numbers is between 96 and 127. The number that will be used in a specific RTP session will be negotiated via SDP.

When encrypted payload is in use the entire double frame of FSTE or OSTE is transported within one RTP packet. In such case this RTP packet contains 60ms of voice.

When no audio is available to be propagated by indicating “no PTT” signaling, efficient use of most valuable QoS bandwidth shall take place by propagating comfort noise packets instead of audio packets. These comfort noise packets shall be encoded with payload type 13 according to RFC 3389 “Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)”. Each peer (both DF-Client and DF-GW) shall continue periodic propagation of RTP packets to enable his peer on performing session supervision by RTP heartbeat monitoring. By replacing the regular RTP audio packets with comfort noise packets bandwidth is saved with increased transmission interval as well as reduced packet size.

Obviously no real comfort noise shall be generated at the receiving party of any “comfort noise” packet – silence shall played instead.

If one of the peers makes use of comfort noise, payload type 13 has to be announced in the same

```
m=audio …
```

line as the voice codec (payload type 8 or dynamic payload type) in the sdp part.

To indicate an advisory for RTP heartbeat timeout the maxptime parameter SHALL be part of the sdp part in the SIP messages. If one DF-Client does not provide the maxptime parameter the DF-GW MAY NOT rely on regular RTP packet interval that could be used for RTP heartbeat monitoring.

7.9.1.3 PTIME PARAMETER

The ptime parameter allows negotiation of the packet size. Depending on the codec, following packet sizes SHALL be supported by the DR-GW:

Codec	ptime	20 ms	30 ms	60 ms

Codec	ptime	20 ms	30 ms	60 ms
G.711		X	X	X
ACELP			X	X
FSTE / OSTE				X
Any other codecs			X	X

7.9.1.4 MAXPTIME PARAMETER

The maxptime parameter SHALL be put into each sdp message body a DF-GW transmits and SHOULD be put (highly recommended) into each sdp message body generated by a DF-Client. The maxptime according to RFC 4566 specifies the maximum interval between two RTP packets. The maximum interval should be between 200 and 2000 ms as a recommendation.

7.9.1.5 MONITORING SESSION WITH ACELP ENCODING

Monitoring Session with ACELP encoding – Example:

```
v=0
o=yourClient 0 2 IN IP4 172.31.60.191
s=subject
t=0 0
m=audio 8196 RTP/AVP 99 13
c=IN IP4 172.31.60.191
a=rtpmap:99 TETRA/8000
a=recvonly
a=type:Radio-Rxonly
```

Note: There a numeric payload type assigned dynamically because no static numeric payload type is assigned for ACELP codec. Though this example uses payload type "99" for audio/TETRA, it is highly discouraged to think of this number as statically assigned to this format.

7.9.1.6 MONITORING SESSION WITH G.711 A-LAW ENCODING

The specialty of this SDP is the SIP session negotiation in full duplex while the selection mode is Rx only. This means the client expects RTP packets even if it does not make use of it for audio propagation. An application for such sdp is negotiation to provide heartbeat supervision based on RTP which allows for a very quick and highly reliable detection of failures.

Monitoring Session with G.711 A-Law encoding – Example:

```
v=0
```

```
o=myclient 0 0 IN IP4 192.168.12.97
s=conversation
c=IN IP4 192.168.12.97
t=0 0
m=audio 10020 RTP/AVP 8 13
a=rtpmap:8 pcma/8000
a=sendrecv
a=type:Radio-Rxonly
```

7.9.1.7 Selected Talkgroup with ACELP encoding

Selected Talkgroup with ACELP encoding – Examble:

```
v=0
o=hisClient 0 2 IN IP4 10.180.22.93
s=subject
t=0 0
m=audio 8196 RTP/AVP 99 13
c=IN IP4 10.180.22.93
a=rtpmap:99 TETRA/8000
a=sendrecv
a=type:Radio-TxRx
```

Note: There a numeric payload type assigned dynamically because no static numeric payload type is assigned for ACELP codec. Though this example uses payload type "99" for audio/TETRA, it is highly discouraged to think of this number as statically assigned to this format.

7.9.2 Floor Control Message Body

The following example illustrates the structure of FLOOR CONTROL MESSAGE BODY:

```
<?xml version="1.0" encoding="utf-8"?>
<Call_PTTEvent>
  <action>txGranted</action>
  <txGrant>granted</txGrant>
  <talkingParty>
    <subscriber>
      <tsi>
        <mnc>248</mnc>
        <mcc>162</mcc>
        <ssi>70000001</ssi>
      </tsi>
    </subscriber>
    <opta>OPTA123789</opta>
  </talkingParty>
  <attributes>
    <demandPriority>normal</demandPriority>
  </attributes>
  <txrepeat>30</txrepeat>
  <workstationId>Place 231</workstationId>
</Call_PTTEvent>
```

7.9.3 KEY EXCHANGE TRIGGERING MESSAGE BODY

Any kind of key management is performed based on in-dialogue SIP INFO messages with xml structures in the SIP message body. The particular xml schema with exemplary xml content will be specified in a subsequent revision of this document.

8 PROFILE STANDARD FOR THE USE OF SOAP

DR-GW and DR-GW-Client SHALL support SOAP in version 1.2.

SOAP provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML.

A few provisions are made to ensure interoperability. Following the recommendations of the [BasicProfile-v2.0], the “literal” encoding must be used. The use of other encodings, including the SOAP encoding, is prohibited[3]. The SOAP-binding style “document” must be supported. Other binding styles are optional. It is recommended to use the “Document/literal wrapped” style, to avoid ambiguity.

Features regarding an optimized transmission form of XML messages, for example MTOM, XOP and RRSHB, may not be used.

Alternative representations of XML, e.g. [MC-NBFX], may not be used[4].

The SOAP interface contains a REQUEST/RESPONSE and an EVENT model. The transport for SOAP messages in both cases is WebSocket. With WebSockets, it is possible to have a duplex connection initiated by an authenticated client.

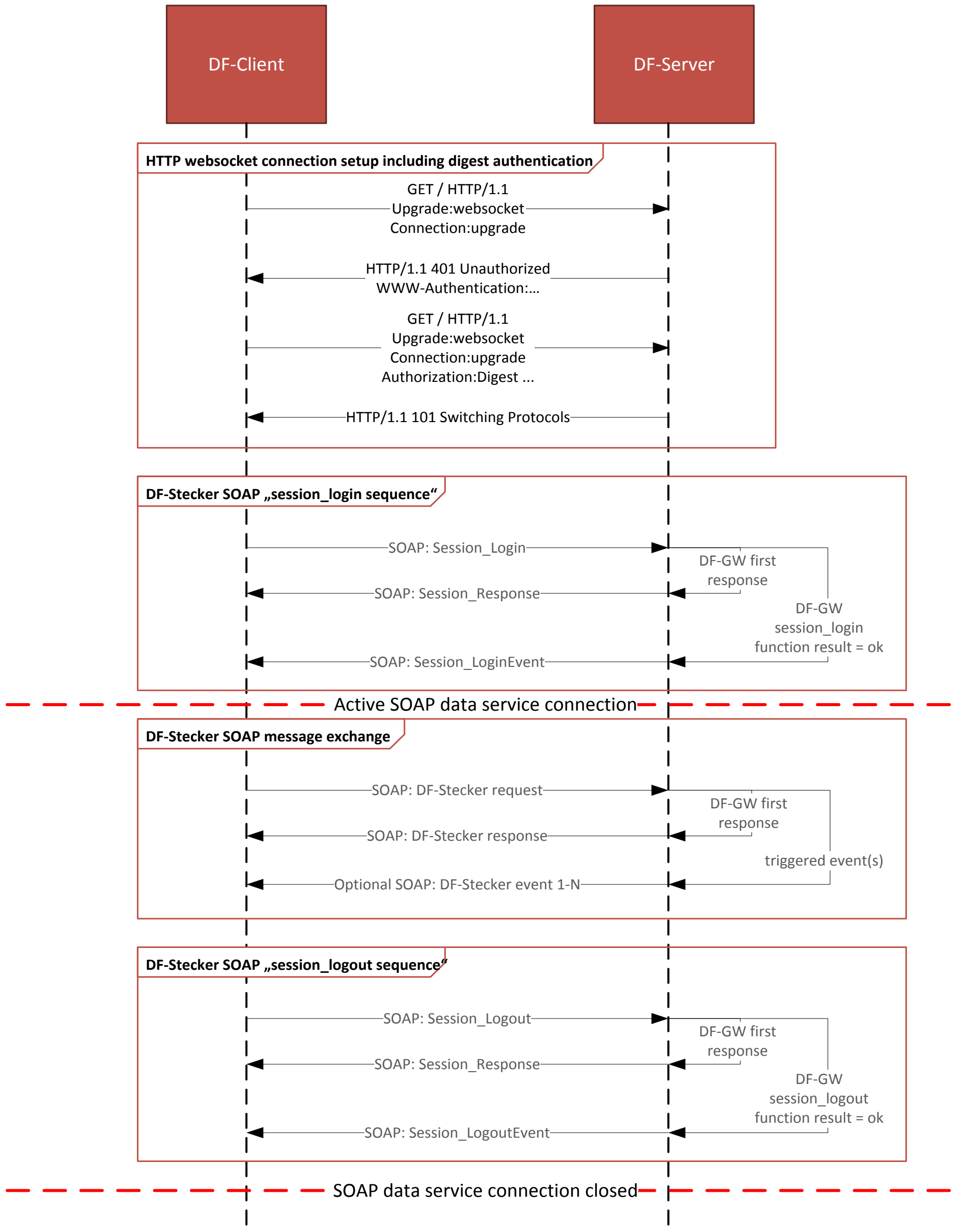
Websockets are used as transport protocol to be friendly for application layer gateways, firewalls and NAT routers.

As the standard HTTP socket has to be used as a base before the upgrade, HTTP authentication shall be used as a standard authentication mechanism.

The following figure shows a typical life-cycle of such a SOAP connection:

[3] http://docs.oasis-open.org/ws-brsp/BasicProfile/v2.0/cs01/BasicProfile-v2.0-cs01.html#_Toc392060487

[4] http://msdn.microsoft.com/en-us/library/cc219210.aspx



DF-Client
DF-Server
HTTP websocket connection setup including digest authentication
GET / HTTP/1.1
Upgrade:websocket
Connection:upgrade
HTTP/1.1 401 Unauthorized
WWW-Authentication:...
GET / HTTP/1.1
Upgrade:websocket
Connection:upgrade
Authorization:Digest ...
HTTP/1.1 101 Switching Protocols
DF-Stecker SOAP „session_login sequence“
SOAP: Session_Login
DF-GW first response
SOAP: Session_Response
DF-GW session_login function result = ok
SOAP: Session_LoginEvent
Active SOAP data service connection
DF-Stecker SOAP message exchange
SOAP: DF-Stecker request
DF-GW first response
SOAP: DF-Stecker response
triggered event(s)
Optional SOAP: DF-Stecker event 1-N
DF-Stecker SOAP „session_logout sequence“
SOAP: Session_Logout
DF-GW first response
SOAP: Session_Response
DF-GW session_logout function result = ok
SOAP: Session_LogoutEvent
SOAP data service connection closed

Figure 35 – SOAP Session Life-Cycle

Complete definition and description of XML data, used in this interface, is defined in the Chapter 10: *Interface Definition*.

9 REQUEST, RESPONSE AND EVENT SYSTEM

9.1 GENERAL MESSAGE SEQUENCE

In general, each REQUEST is answered by a RESPONSE. The behavior is specified as well in the wsdl files. This synchronous response does not contain the final information. Instead the final information is sent asynchronously as an EVENT. When the DR-GW does provide additional information (e.g. because the successful request subscribed for subscriber tracking) this information is sent via INDICATIONs.

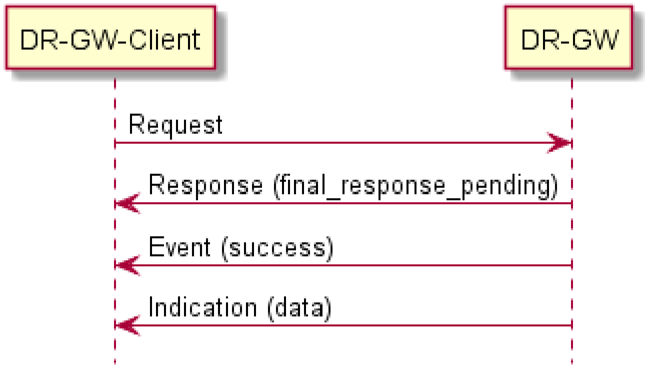


Figure 36 – Successful request / response

It is a matter and the responsibility of the DR-GW implementation on how to map this generic message flow at the DR-GW Interface to the according TCS-API. Nevertheless the interface description itself does not make any specification on the meaning of “final”. Due to the nature of the TETRA system via TCS-API as backend “final” does not ensure end to end acknowledgement. DR-GW implementation should do its best to deliver the most reliable return value as soon as it can.

9.2 REQUEST/RESPONSE/EVENT/INDICATION ORDER

Due to multi-threaded nature of DF-GW and connected TCS, it is possible to receive asynchronous EVENT before the called method synchronously returns the RESPONSE. This should be taken into consideration when implementing DF-Clients.

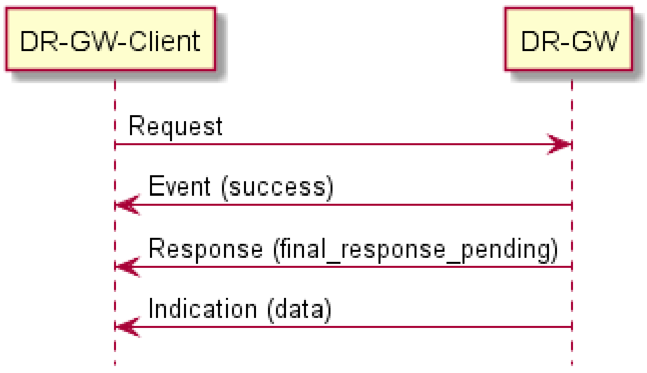


Figure 37 –Request / response with Event before Response

As the INDICATION contains data provided spontaneously they may occur in any order related to event and response.

INDICATIONs as such are caused by business rules considering REQUESTs. There may be no ID or reference linking the INDICATION to the REQUEST.

9.3 Error Scenarios

Errors may occur during synchronous processing of the REQUEST or asynchronously.

When error occurs already at processing of the REQUEST no further event is fired by the DR-GW.

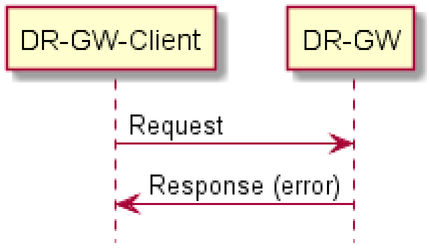


Figure 38 –Erroneous request

Even in scenario where RESPONSE indicates success the asynchronous EVENT indicates an error. This may lead to the obvious paradox situation where the EVENT indicates an error but the RESPONSE submit after indicates success due to the asynchronous / multithreaded characteristics of the system.

In such scenarios the EVENT with the ERROR takes precedence over the RESPONSE.

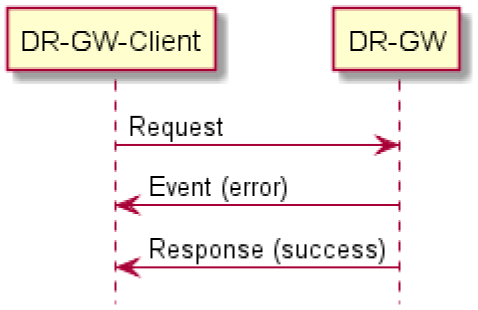


Figure 39 –Asynchronous fired error

INDICATIONs may appear even in case the REQUEST fails. One obvious example would be a failed un-subscription of subscriber tracking where the INDICATIONs still arrive but technically do not look different than the ones triggered due to a REQUEST.

10 INTERFACE DEFINITION

10.1 INTERFACE DIAGRAM

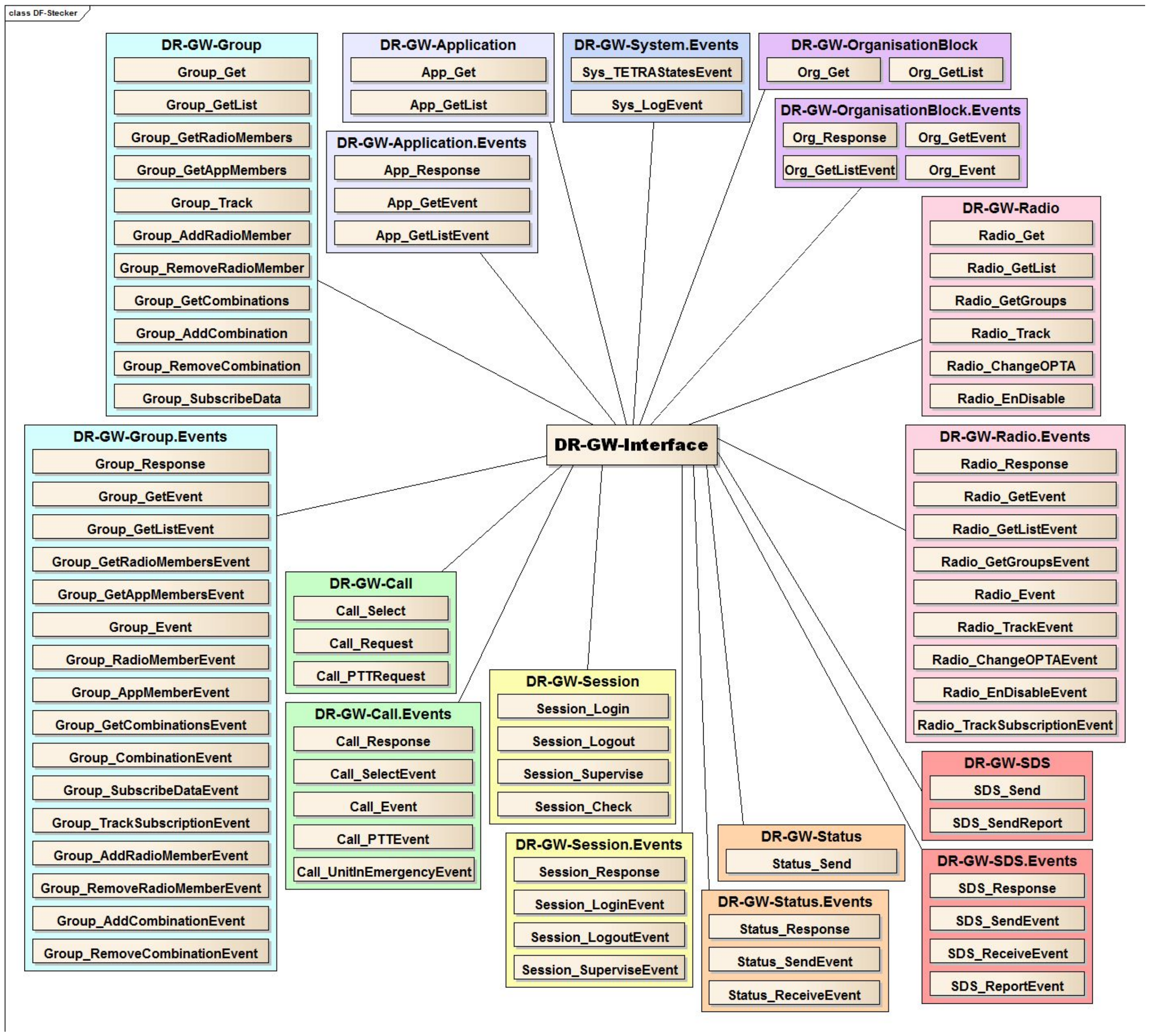


Figure 40 – DR-GW Graph

The Interface Diagram represents all method calls the DR-GW interface for Tetra call handling (in SIP) and Tetra messaging/control (in Soap) . See for details also the schema definitions in the appendix "DR-GW-Schemas".

10.2 Request Interface: DR-GW-Session

10.2.1 Request: Session_Login

This method is used to perform login procedure.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
clientId	xs:string	1..1	DR-GW-Client identifier
supervise	ctS:typeSuperviseTimeout	0..1	Supervise timeout in seconds (20, 30, 60)
version	xs:string	0..1	Client version; the version string shall be encoded following the definition in chapter 7.8.1 without the heading (e.g. “radio.1.1”)

10.2.2 Request: Session_Logout

This method is used to perform logout procedure.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier

10.2.3 REQUEST: SESSION_SUPERVISE

This method is used to perform supervised action.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier

10.2.4 REQUEST: SESSION_CHECK

This method is used to perform a connection check from the DR-Client up to the SOAP access point of the DR-GW.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
clientid	xs:string	1..1	Client identifier

Output parameters:

The output is formed as a list (xml sequence) of sequence containing the service description and the according result (the same value as the generic result value). The service list contains entries like:

- Group communication
- Individual communication
- SDS and Status propagation and reception
- ...

And is a per DR-GW Client specific list of service availability

10.3 RESPONSE AND EVENT INTERFACE: DR-GW-SESSION.EVENTS

10.3.1 RESPONSE: SESSION_RESPONSE

This is a general response for DR-GW-Session.Events interface.

Output parameters:

Argument	Type	Occurs	Description

requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information

10.3.2 Event: Session_LoginEvent

This event is generated as a result of DR-GW-Client Session_Login.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	0..1	Request identifier
result	xs:typeResult	1..1	Result information
ISSI	xs.string	0..1	Assigned ISSI

10.3.3 EVENT: SESSION_LOGOUTEVENT

This event is generated as a result of DR-GW-Client Session_Logout.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	0..1	Request identifier
result	xs:typeResult	1..1	Result information
reason	xs:unsignedLong	0..1	Reason of logout event

10.3.4 EVENT: SESSION_SUPERVISEEVENT

This event is generated as a result of DR-GW-Client Session_Supervise.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	0..1	Request identifier
result	xs:typeResult	1..1	Result information

10.4 REQUEST INTERFACE: DR-GW-SDS

10.4.1 REQUEST: SDS_SEND

This method is used to send SDS.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
Sds	ctS:typeSDS	1..1	All SDS attributes

The DR-GW Client can select either to send SDS as text message (string data) or to send a text message as hex Data (hexBinary hexdata). Even if the xsd Schema definition would allow to send both (the reason is to use one common type for both sending as described here as well as receiving – see “10.5.3 Event: SDS_ReceiveEvent” - SDS from client point of view) propagating both information leads to undefined behaviour.

In case the client decides to use text encoding character set conversion is the matter of the DF-GW and is implementation specific.

If the client decides to use hexdata encoding the client is responsible to generate the entire “User Data” Information element according “ETSI EN 300 392-2” chapter 29.4.3.13 including but not limited to:

- Text Coding Schema
- Timestamp
- Character packing
- and character encoding

In case of messages exceeding the maximum message length it is the client’s responsibility to split the message in case hexdata is used. In case text encoding is selected by the client, the DF-GW has to split messages exceeding the maximum message size.

10.4.2 REQUEST: SDS_ SENDREPORT

This method is used to send SDS report on received SDS.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
target	ct:typeAddress	1..1	target subscriber
msgRef	xs:unsignedByte	1..1	Message reference
deliveryStatus	xs:unsignedByte	1..1	Delivery status

10.5 RESPONSE AND EVENT INTERFACE: DR-GW-SDS.EVENTS

10.5.1 RESPONSE: SDS_RESPONSE

This is a general response for DR-GW-SDS.Events interface.

Output parameters:

Argument	Type	Occurs	Description

requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information

10.5.2 Event: SDS_SendEvent

This event is generated as a result of DR-GW-Client SDS_Send.

Output parameters:

Argument	**Type**	**Occurs**	**Description**
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
msgRef	xs:unsignedByte	0..1	Message reference

10.5.3 EVENT: SDS_RECEIVEEVENT

This event is generated upon receiving SDS.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	0..1	Request identifier
result	xs:typeResult	0..1	Result information
sds	ctS:typeSDS	1..1	All SDS attributes

The DR-GW provides both contents the text message (string data) and hex Data (hexBinary hexdata). When a SDS was split and can be reassembled by the DF-GW, the DF-GW delivers the split portion(s) of the message in pure hexdata representation. With the last portion of the message in hexdata representation the re-assembled text message is delivered too if applicable.

In the hexdata representation no decoding and no interpretation is performed by the DF-GW – in that case this is the task of the DF-GW client.

10.5.4 EVENT: SDS_REPORTEVENT

This event is generated upon receiving SDS report.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
source	ct:typeAddress	1..1	Sending subscriber to be able to match the event to the according request for the DF-Client
target	ct:typeAddress	1..1	target subscriber to be able to match the event to the according request for the DF-Client
msgRef	xs:unsignedByte	1..1	Message reference
deliveryStatus	xs:unsignedByte	1..1	Delivery status

10.6 Request Interface: DR-GW-Status

10.6.1 Request: Status_Send

This method is used to send status.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
status	ctS:typeStatus	1..1	All Status attributes

10.7 Response and Event Interface: DR-GW-Status.Events

10.7.1 Response: Status_Response

This is a general response for DR-GW-Status.Events interface.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information

10.7.2 Event: Status_SendEvent

This event is generated as a result of DR-GW-Client Status_Send.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information

10.7.3 Event: Status_ReceiveEvent

This event is generated upon receiving status.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
status	ctS:typeStatus	1..1	All Status attributes

10.8 Request interface: DR-GW-OrganisationBlock

10.8.1 Request: Org_Get

This method is used to get organization block attributes.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
orgblockId	ctO:typeOrganisationBlockId	1..1	Organization block id

10.8.2 Request: Org_GetList

This method is used to get organization block subtree attributes.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
orgblockId	ctO:typeOrganisationBlockId	0..1	Organization block id

10.9 Response and Event Interface: DR-GW-OrganisationBlock.Events

10.9.1 Response: Org_Response

This is a general response for DR-GW-OrganisationBlock. Events interface.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information

10.9.2 Event: Org_GetEvent

This event is generated as a result of DR-GW-Client Org_Get.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
orgblock	ctO:typeOrganisationBlock	1..1	Organization block attributes

10.9.3 Event: Org_GetListEvent

This event is generated as a result of DR-GW-Client Org_GetList.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
orgblock	ctO:typeOrganisationBlock	0..N	Organization block attributes

listEnd	xs:boolean	1..1	End of list indicator

10.9.4 Event: Org_Event

This event is generated upon creation, modification or deletion of organization block.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
orgblock	ctO:typeOrganisationBlock	1..1	Organization block attributes
delete	xs:boolean	0..1	Deletion indicator

10.10 Request Interface: DR-GW-Group

10.10.1 Request: Group_Get

This method is used to get group attributes.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
group	ct:typeSubscriberAddress	1..1	group address

10.10.2 Request: Group_GetList

This method is used to get groups and their attributes belonging to a given organization block.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
orgblockId	ctO:typeOrganisationBlockId	0..1	Organization block id

10.10.3 Request: Group_GetRadioMembers

This method is used to get radio members of group.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
group	ct:typeSubscriberAddress	1..1	group address

10.10.4 Request: Group_GetAppMembers

This method is used to get application members of the group.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
group	ct:typeSubscriberAddress	1..1	group address

10.10.5 Request: Group_Track

This method is used to start or stop tracking of the group.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
group	ct:typeSubscriberAddress	1..1	group address
mask	ctG:typeGroupTrackingMask	0..1	Tracking mask
stop	xs:boolean	1..1	Stop indicator

GROUP_TRACK allows individual subscriptions for different information about a group. The information for which a DR-GW-client is subscribing is handed over via the argument MASK. A description about the different bit values and their meaning for this command can be found in EADS TETRA System Release 5.5, TCS API Description, 8.2.2.22 SubscribeUpdates.

To subscribe for the information which radios are added or removed to a group the bit value TCS_GROUP_SUBSCRIPTION_MASK_VALUES_T_RS_ADD_REMOVE_C has to be used in MASK. The corresponding event for this subscription is Group_RadioMemberEvent.

To subscribe for the information if a group is combined with other group(s) or removed from such a combination the bit value

TCS_GROUP_SUBSCRIPTION_MASK_VALUES_T_RS_ADD_REMOVE_C has to be used. The corresponding event for this subscription is Group_CombinationEvent.

10.10.6 Request: Group_AddRadioMember

This method is used to add radio member to the group.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
radio	ct:typeSubscriberAddress	1..1	Member address
group	ct:typeSubscriberAddress	1..1	group address
membership	ctG:typeMembershipType	0..1	Membership type

10.10.7 Request: Group_RemoveRadioMember

This method is used to remove a radio member from the group.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
radio	ct:typeSubscriberAddress	1..1	Member address
group	ct:typeSubscriberAddress	1..1	group address

10.10.8 REQUEST: GROUP_GETCOMBINATIONS

This method is used to get groups belonging to the same combined group.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
group	ct:typeSubscriberAddress	1..1	group address

10.10.9 REQUEST: GROUP_ADDCOMBINATION

This method is used to add a group to the combined group.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
group	ct:typeSubscriberAddress	1..1	group address
baseGroup	ct:typeSubscriberAddress	1..1	Base group address
Force	xs:boolean	0..1	Force indicator

10.10.10 REQUEST: GROUP_REMOVECOMBINATION

This method is used to remove a group from the combined group.

Input parameters:

Argument	Type	Occurs	Description

requestId	xs:unsignedLong	1..1	Request identifier
group	ct:typeSubscriberAddress	1..1	group address
baseGroup	ct:typeSubscriberAddress	1..1	Base group address

10.10.11 Request: Group_SubscribeData

This method is used to start or stop receiving of SDS and Status for the group.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
group	ctG:typeGroupSubscribeData	1..N	Group subscribe data attributes

10.11 Response and Event Interface: DR-GW-Group. Events

10.11.1 Response: Group_Response

This is a general response for the DR-GW-Group.Events interface.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information

10.11.2 Event: Group_GetEvent

This event is generated as a result of DR-GW-Client Group_Get.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier

result	xs:typeResult	1..1	Result information
group	ctG:typeGroup	1..1	Group attributes

10.11.3 Event: Group_GetListEvent

This event is generated as a result of DR-GW-Client Group_GetList.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
group	ctG:typeGroup	0..N	Group attributes
listEnd	xs:boolean	1..1	End of list indicator

10.11.4 Event: Group_GetRadioMembersEvent

This event is generated as a result of DR-GW-Client Group_GetRadioMembers.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
group	ct:typeSubscriberAddress	1..1	the group to which the request is related to
radio	ct:typeSubscriberAddress	0..N	Radio address
listEnd	xs:boolean	1..1	End of list indicator

10.11.5 Event: Group_GetAppMembersEvent

This event is generated as a result of DR-GW-Client Group_GetAppMembers.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
app	ct:typeSubscriberAddress	0..N	Application address
listEnd	xs:boolean	1..1	End of list indicator

10.11.6 EVENT: GROUP_EVENT

This event is generated upon creation, modification or deletion of group.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
group	ctG:typeGroup	1..1	Group attributes
delete	xs:boolean	1..1	Deletion indicator

10.11.7 EVENT: GROUP_RADIOMEMBEREVENT

This event is generated upon addition or deletion of a radio member from the group. Subscription of Group_Track is the precondition this event is fired by the DR-GW. Use case description for this scenario is provided in chapter 5.12 ”Radio/Group Tracking”. For the Group_Track a general presence of a mobile is sufficient.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
group	ct:typeSubscriberAddress	1..1	Group address
radio	ct:typeSubscriberAddress	1..1	Radio address

delete	xs:boolean	1..1	Deletion indicator

10.11.8 Event: Group_AppMemberEvent

This event is generated upon addition or deletion of the application member from the group.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:unsignedLong	0..1	Result code as in TCS API description
group	ctG:typeGroup	1..1	Group attributes
app	ct:typeSubscriberAddress	1..1	Application address
delete	xs:boolean	1..1	Deletion indicator

10.11.9 Event: Group_GetCombinationsEvent

This event is generated as a result of DR-GW-Client Group_GetCombinations

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
group	ct:typeSubscriberAddress	1..1	Group address
baseGroup	ct:typeSubscriberAddress	0..1	Base group address
constitGroup	ct:typeSubscriberAddress	1..7	Constituent groups addresses

10.11.10 Event: Group_CombinationEvent

This event is generated upon addition or deletion of a group from the combined group.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
group	ct:typeSubscriberAddress	1..1	Group address
baseGroup	ct:typeSubscriberAddress	1..1	Base group address
constitGroup	ct:typeSubscriberAddress	0..7	Constituent groups addresses

10.11.11 EVENT: GROUP_GROUPSUBSCRIBEDATAEVENT

This event is used to transport subscription information about talkgroup(s) to a DR-GW-Client. It can also be used to signal subscription errors or changes originating in the DR-system.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
group	ct:typeSubscriberAddress	1..N	Group address

10.11.12 EVENT: GROUP_TRACKSUBSCRIPTIONEVENT

This event is generated upon change subscription status of group tracking. The subscription was originally initiated with Group_Track

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
group	ct:typeSubscriberAddress	1..1	Group address
mask	ctG:typeGroupTrackingMask	0..1	Tracking mask
stop	xs:boolean	1..1	Result information

10.11.13 EVENT: GROUP_ADDRADIOMEMBEREVENT

This event is generated as the asynchronous response by the DR-GW for the previously sent request Group_AddRadioMember

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
radio	ct:typeSubscriberAddress	1..1	Member address
group	ct:typeSubscriberAddress	1..1	group address

10.11.14 EVENT: GROUP_REMOVERADIOMEMBEREVENT

This event is generated as the asynchronous response by the DR-GW for the previously sent request Group_RemoveRadioMember

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
radio	ct:typeSubscriberAddress	1..1	Member address
group	ct:typeSubscriberAddress	1..1	group address

10.11.15 EVENT: GROUP_ADDCOMBINATIONEVENT

This event is generated as the asynchronous response by the DR-GW for the previously sent request Group_AddCombination.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
group	ct:typeSubscriberAddress	1..1	group address
baseGroup	ct:typeSubscriberAddress	1..1	Base group address

10.11.16 EVENT: GROUP_REMOVECOMBINATIONEVENT

This event is generated as the asynchronous response by the DR-GW for the previously sent request Group_RemoveCombination.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
group	ct:typeSubscriberAddress	1..1	group address
baseGroup	ct:typeSubscriberAddress	1..1	Base group address

10.12 Request Interface: DR-GW-Application

10.12.1 Request: App_Get

This method is used to get application attributes.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
app	ct:typeSubscriberAddress	1..1	Application address

10.12.2 Request: App_GetList

This method is used to get applications and their attributes belonging to a given organization block.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
orgblockId	ctO:typeOrganisationBlockId	0..1	Organization block id

10.13 Response and Event Interface: DR-GW-Application.Events

10.13.1 Response: App_Response

This is a general response for DR-GW-Application. Events interface.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information

10.13.2 EVENT: APP_GETEVENT

This event is generated as a result of DR-GW-Client App_Get.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
app	ctA:typeApplicatio n	1..1	Application attributes

10.13.3 EVENT: APP_GETLISTEVENT

This event is generated as a result of DR-GW-Client App_GetList.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
app	ctA:typeApplicatio n	1..1	Application attributes
listEnd	xs:boolean	1..1	End of list indicator

10.14 EVENT INTERFACE: DR-GW-SYSTEM.EVENTS

10.14.1 EVENT: SYS_TETRASTATESEVENT

This event is generated upon change of a DR-GW system state.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
tcsState	ctS:typeSystemElementState	0..1	TCS state
dxtState	ctS:typeSystemElementState	0..1	DxT state
cddconnectionState	ctS:typeSystemElementState	0..1	CDD connection state
cddserverState	ctS:typeSystemElementState	0..1	CDD server state

10.14.2 EVENT: SYS_LOGEVENT

This event is generated upon DR-GW sending additional textual information.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
value	xs:hexBinary	0..1	Value
text	xs:normalizedString	0..1	Text

10.15 REQUEST INTERFACE: DR-GW-RADIO

10.15.1 REQUEST: RADIO_GET

This method is used to get radio attributes.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
radio	ct:typeSubscriberAddress	1..1	Radio address

10.15.2 REQUEST: RADIO_GETLIST

This method is used to get radios and their attributes belonging to a given organization block.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
orgblockId	ctO:typeOrganisationBlockId	0..1	Organization block id

10.15.3 REQUEST: RADIO_GETGROUPS

This method is used to get groups, to which a given radio belongs.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
radio	ctR:typeRadio	1..1	Radio attributes

10.15.4 REQUEST: RADIO_TRACK

This method is used to start or stop tracking of radio.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
radio	ct:typeSubscriberAddress	1..1	Radio address
stop	xs:boolean	1..1	Stop indicator

10.15.5 REQUEST: RADIO_CHANGEOPTA

This method is used to perform the change of the OPTA associated with a Mobile.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
radio	ct:typeSubscriberAddress	1..1	Radio address
alias	xs:normalizedString	1..1	the new OPTA

10.15.6 REQUEST: RADIO_ENDISABLE

This method is used to enable or disable a mobile temporarily.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
radio	ct:typeSubscriberAddress	1..1	Radio address
enabled	xs:boolean	1..1	True to enable, False to disable

10.16 Response and Event Interface: DR-GW-Radio.Events

10.16.1 Response: Radio_Response

This is a general response for DR-GW-Radio.Events interface.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information

10.16.2 Event: Radio_GetEvent

This event is generated as a result of DR-GW-Client Radio_Get.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
radio	ctR:typeRadio	1..1	Radio attributes

10.16.3 EVENT: RADIO_GETLISTEVENT

This event is generated as a result of DR-GW-Client Radio_GetList.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
radio	ctR:typeRadio	0..N	Radio attributes
listEnd	xs:boolean	1..1	End of list indicator

10.16.4 EVENT: RADIO_GETGROUPSEVENT

This event is generated as a result of DR-GW-Client Radio_GetGroups.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
radio	ct:typeSubscriberAddress	1..1	The radio's ISSI
group	ctR:typeRadioGroupSelection	0..N	Groups addresses with the according selection level
listEnd	xs:boolean	1..1	End of list indicator

10.16.5 EVENT: RADIO_GROUPSEVENT

This event is generated as a consequence of tracking a radio.

Output parameters (in case a modification has happened):

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information

radio	ct:typeSubscriberAddress	1..1	The radio's ISSI
group	ctR:typeRadioGroupSelection	0..N	Groups with the according selection level
listEnd	xs:boolean	1..1	End of list indicator

Output parameters (in case a group was removed):

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
radio	ct:typeSubscriberAddress	1..1	The radio's ISSI
deletedGroup	ctR:typeRadioGroupSelection	1..1	Group which was removed
listEnd	xs:boolean	1..1	End of list indicator

10.16.6 Event: Radio_Event

This event is generated upon creation, modification or deletion of radio.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
radio	ctR:typeRadio	0..N	Radio attributes
delete	xs:boolean	1..1	End of list indicator

10.16.7 Event: Radio_TrackEvent

This event is generated upon change of radio tracking attributes.

Output parameters:

Argument	Type	Occurs	Description

requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
trackingData	ctR:typeRadioTrackingData	1..1	Tracking information for a radio

10.16.8 Event: Radio_TrackSubscriptionEvent

This event is generated upon change subscription status of radio tracking. The subscription was originally initiated with Radio_Track

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
radio	ct:typeSubscriberAddress	1..1	Radio address
stop	xs:boolean	1..1	Result information

10.16.9 Request: Radio_ChangeOptaEvent

This event is generated as a result of DR-GW-Client Radio_ChangeOPTA.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
radio	ct:typeSubscriberAddress	1..1	Radio address
opta	xs:normalizedString	1..1	the new OPTA

10.16.10 Request: Radio_EnDisableEvent

This event is generated as a result of DR-GW-Client Radio_EnDisabled. It will be sent first if the radio could be enabled or disabled in the TETRA system itself and second if the radio could be enabled or disabled over the air.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
radio	ct:typeSubscriberAddress	1..1	Radio address
reason	xs:unsignedByte	0..1	Reason for disabling radio, optional
enabled	xs:boolean	1..1	True if enable, False if disabled
overTheAir	xs:boolean	0..1	True if the radio was EnDisabled over the air. False if the state was only changed in the TETRA system

10.17 Request Interface: DR-GW-Call

10.17.1 Request: Call_Select

This method reserves speech line for the targets of selection operation. When client uses this request via the SOAP interface the only selection levels allowed are:

- event
- no

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
sel	ctC:typeSelection	1..1	Selection attributes

10.17.2 Request: Call_Request

This method is used to accomplish all call related operations.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
tetraCallId	xs:unsignedLong	0..1	TETRA Call Id
action	ctC:typeActionRequest	1..1	Action type
attributes	ctC:typeCallAttributes	0..1	Call attributes
callingParty	ct:typeAddress	0..1	Calling party address
calledParty	ct:typeAddress	0..1	Called party address
workstationId	ctC:typeWorkstationId	0..1	Workstation Id

10.17.3 Request: Call_PTTRequest

This method is used for "DemandTx" and "CeaseTx" actions.

Input parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
tetraCallId	xs:unsignedLong	0..1	TETRA call Id
action	ctC:typePTTRequest	1..1	Action type (demandTx, ceaseTx)
attributes	ctC:typeCallAttributes	0..1	Call attributes
talkingParty	ct:typeAddress	0..1	Talking party address
workstationId	ctC:typeWorkstationId	0..1	Workstation Id

10.18 Response and event Interface: DR-GW-Call.Events

10.18.1 Response: Call_Response

This is a general response for DR-GW-Call.Events interface.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information

10.18.2 Event: Call_SelectEvent

This event is generated as a result of DR-GW-Client Call_Select.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
sel	ctC:typeSelection	1..1	Selection attributes

10.18.3 EVENT: CALL_EVENT

This event is generated upon all call related operations.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
result	xs:typeResult	1..1	Result information
tetraCallId	xs:unsignedLong	0..1	TETRA call Id
action	ctC:typeActionEvent	1..1	Action type
attributes	ctC:typeCallAttributes	0..1	Call attributes
callingParty	ct:typeAddress	0..1	Calling party address
calledParty	ct:typeAddress	0..1	Called party address
keymngstate	ctC:typeKeyManagement	0..1	State of key management

10.18.4 EVENT: CALL_PTTEVENT

This event is generated upon "txGranted" and "txCeased" events.

Output parameters:

Argument	Type	Occurs	Description
requestId	xs:unsignedLong	1..1	Request identifier
Result	xs:typeResult	1..1	Result information
tetraCallId	xs:unsignedLong	0..1	TETRA call Id
Action	ctC:typePTTEvent	1..1	Action type (txGranted, txCeased)
txGrant	ctC:typeTxGrant	0..1	TxGrant flag
txInterrupt	xs:boolean	0..1	TxInterrupt flag
talkingParty	ct:typeAddress	0..1	Talking party address
Attributes	ctC:typeCallAttributes	0..1	Call attributes
Txrepeat	xs:unsignedLong	0..1	PTT repeat timer suggested by server
workstationId	ctC:typeWorkstationId	0..1	Workstation Id

10.18.5 EVENT: CALL_UNITINEMERGENCYEVENT

This event is used to inform DR-Clients about changes of a DR-mobiles emergency status.

Output parameters:

Argument	Type	Occurs	Description
radio	ct:typeSubscriberAddress	1..1	Radio address
group	ct:typeSubscriberAddress	1..1	Received on group address
emergencyInfo	xs:typeEmergencyInfo	1..1	The emergency information from the radio
tsstamp	xs:dateTime	1..1	Time stamp