ETSI TS 129 335 V19.0.0 (2025-10)

TECHNICAL SPECIFICATION

Digital cellular telecommunications system (Phase 2+) (GSM);
Universal Mobile Telecommunications System (UMTS);
LTE;
User Data Convergence (UDC);
User data repository access protocol over the Ud interface;
Stage 3
(3GPP TS 29.335 version 19.0.0 Release 19)

Reference
RTS/TSGC-0429335vj00

Keywords
GSM,LTE,UMTS

ETSI

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

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

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

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

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Legal Notice

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities. These shall be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between 3GPP and ETSI identities can be found at 3GPP to ETSI numbering cross-referencing.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document specifies the stage 3 of the Ud interface between the Front-Ends (FEs) and the User Data Repository (UDR) in the User Data Convergence (UDC architecture).

This 3GPP Technical Specification (TS) specifies the protocol and interactions between the FE and the UDR for Ud reference point, in particular:

- The details of the LDAP protocol that are to be considered
- The details of the SOAP envelope that provide support for subscriptions to notifications and notifications about data changes service (S/N operations).

The User Data Convergence Stage 2 description (architecture and information flows) is specified in 3GPP TS 23.335 [10].

The UDR data model used with LDAP (i.e. attributes, object classes and directory information tree) is implementation specific and is left outside the scope of 3GPP specifications. For multivendor interoperability between FEs and UDR specific integration projects are needed. Some examples of Ud data models are described in 3GPP TR 29.935 [19].

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TR 41.001: "GSM Release specifications".

[3] 3GPP TR 21 912 (V3.1.0): "Example 2, using fixed text".

[4] IETF RFC 4510: "Lightweight Directory Access Protocol (v3)".

[5] IETF RFC 5805: "Lightweight Directory Access Protocol (LDAP) Transactions".

[6] W3C Recommendation "Simple Object Access Protocol (SOAP) 1.2" (27 April 2007). http://www.w3.org/TR/.

[7] Void.

[8] IETF RFC 4511: "Lightweight Directory Access Protocol (LDAP): The Protocol".

[9] IETF RFC 4528: "Lightweight Directory Access Protocol (LDAP) Assertion Control".

[10] 3GPP TS 23.335: "User Data Convergence (UDC); Technical realization and information flows".

[11] IETF RFC 4512: "Lightweight Directory Access Protocol (LDAP): Directory Information Models".

[12]. 3GPP TS 32.182: "Telecommunication management; User Data Convergence (UDC); Common Baseline Information Model".

[13] Void

[14] Void

[15] 3GPP TS 33.210: "3G Security; Network Domain Security; IP network layer security".

[16] IETF RFC 4513: "Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms".

[17] National Institute of Standards and Technology, FIPS Pub 197: Advanced Encryption Standard (AES), 26 November 2001. .

[18] OASIS "Directory Services Markup Language v2.0" (30 April 2002) http://www.oasis-open.org/.

[19] 3GPP TR 29.935: "Study on UDC Data Model".

[20] OASIS: " Web Services Security: SOAP Message Security 1.1 OASIS Standard Specification, 1 February 2006", https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.

[21] OASIS: "Web Services Security: "UsernameToken Profile 1.1 OASIS Standard Specification, 1 February 2006", https://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-UsernameTokenProfile.pdf.

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

LDAP Session: starts with LDAP Bind Operation and ends with LDAP Unbind Operation or Notice of Disconnection.

Front End: a core network functional entity or service layer entity or provisioning entity that can access user data stored in a unique repository.

Front End Identifier: A name that uniquely identifies an FE within the set of all FEs accessing an UDR.

Front End Cluster: FEs handling the same application may be grouped in clusters to differentiate between them e.g. with regard to geographical location, feature support, vendor, or other characteristics. All FEs within a cluster are treated equally for required purposes (e.g. authorization, notifications, etc.).

Application type: The application handled by a FE (e.g. HLR) determines the application type of the FE. The application type is derived from the name indicated by a FE.

Front End Cluster Identifier: A name that identifies a cluster grouped with FEs supporting the same application.

User Data Repository: facility where user data can be accessed stored and managed in a common way.

3.2 Symbols

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

Ud reference point between a FE and the UDR

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

FE Front End
UDR User Data Repository

NDS Network Domain Security

4 Protocol Stack

4.1 General

Data access messages on the Ud interface shall make use of IETF RFC 4510 [4] and IETF RFC 4511 [8]. See clause 4.2 for details.

Subscription messages and Notification messages on the Ud interface shall make use of SOAP [6]. See clause 4.3 for details.

4.2 Protocol Stack for Ud Data Access Messages

Figure 4.2-1 shows the protocol layering used for UDR data access.

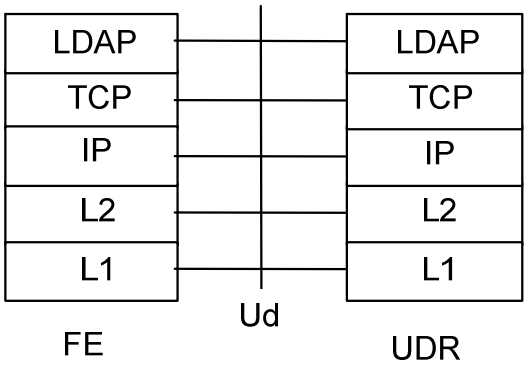

Figure 4.2-1 Protocol Layering for Data Access

4.3 Protocol Stack for Ud Subscriptions/Notifications

Figure 4.3-1 shows the protocol layering used for Ud Subscription/Notification.

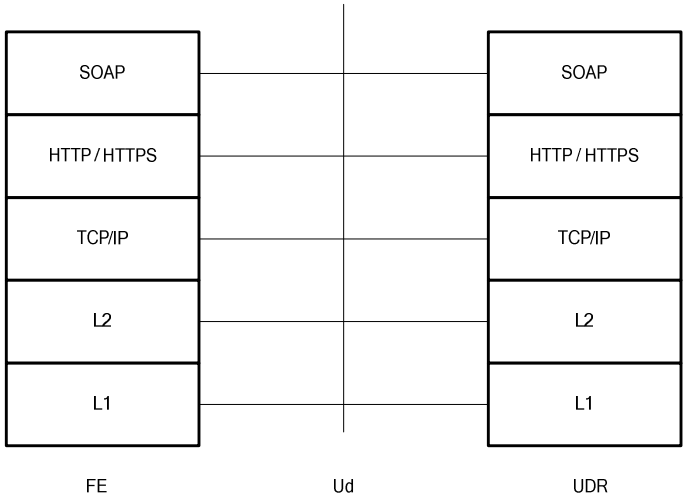

Figure 4.3-1 Protocol Layering for Subscription/Notification

5 General Messages

5.1 General

This clause describes common messages for UDC to establish sessions and administrate transactions. For an existing session, UDC messages are exchanged between the FE and the UDR. See figure B.1-1 in Annex B for general LDAP message flows.

5.2 Open Link for a LDAP Session

To initiate a LDAP session, a Front-End shall first establish a transport connection with the UDR. The transport connection shall be a TCP connection. The IP Layer may be secured according to clause 8. When IPsec is used, an IPsec connection may support several TCP connections, each supporting a LDAP session.

After establishment of the transport connection, the FE shall initiate a LDAP session by sending a LDAP BindRequest message. The establishment of the LDAP session shall comply with IETF RFC 4511 [8]. It shall be done before sending any other LDAP message. FE Identifier or FE Cluster Identifier shall be included in the BindRequest message.

The UDR shall support the "unauthenticated authentication mechanism of simple Bind" and the "name/password authentication mechanism of simple Bind" in the "simple authentication method" specified in IETF RFC 4513 [16].

The UDR derives the application type from the FE Identifier or the FE Cluster Identifier. If the FE provided the Front End Identifier the UDR may also derive the Front End Cluster Identifier.

NOTE: As security is handled at IP Layer (see clause 8), optional security mechanisms (TLS, SASL) described in IETF RFC 4513 [16] specification are not required for Ud.

5.3 Close Link for a LDAP Session

Termination of the LDAP session may be initiated by the FE by sending an UnbindRequest message or by the UDR by sending a Notice of Disconnection message. The termination of the LDAP session shall comply with IETF RFC 4511 [8]

5.4 Transactions

In order to allow FEs to relate a number of update operations, such as Create (see 6.3), Delete (see 6.4), and Update (see 6.5), and have them performed in one unit of interaction, the transaction concept in IETF RFC 5805 "Lightweight Directory Access Protocol (LDAP) Transactions" [5] shall be supported. See figure B.2-1 in Annex B for LDAP Transaction flow.

If used, they shall only be used for a single subscriber in order to decrease the complexity of transactions.

LDAP server shall terminate the transaction if the timer is expired.

5.5 SOAP Authentication

The UDR may support the SOAP WS-Security extension [20] and "UsernameToken" authentication as specified by the Oasis Web Services Security [21].

The WS-Security Username field may be set to a value that is correlated with the username of the LDAP BindRequest (see clause 5.2).

NOTE: As security is handled at the IP Layer (see clause 8), optional security mechanisms (i.e. signature, encryption and passwordDigest) as described in the Oasis Web Services Security specifications [20][21] are not required for the Ud inteface.

6 User Data Convergence Messages

6.1 General

The Query, Create, Delete and Update messages for UDC shall comply with IETF RFC 4511[8].

6.2 Query

Query request messages shall be coded as LDAP SearchRequest messages.

Query result messages shall be coded as LDAP SearchResultEntry, SearchResultReference, and SearchResultDone messages.

6.3 Create

Create request messages shall be coded as LDAP AddRequest messages or as LDAP ModifyRequest messages with the "operation" field set to "add", depending on the data to be created.

Create result messages shall be coded as LDAP AddResponse messages or as LDAP ModifyResponse messages, depending on the used LDAP request message.

6.4 Delete

Delete request messages shall be coded as LDAP DelRequest messages or as LDAP ModifyRequest messages with the "operation" field set to "delete", depending on the data to be deleted.

Delete result messages shall be coded as LDAP DelResponse messages or as LDAP ModifyResponse messages, depending on the used LDAP request message.

The LDAP assertion control as specified in IETF RFC 4528 [9] may be used for conditional delete operation if the FE knows that the UDR supports the corresponding control.

6.5 Update

Update request messages shall be coded as LDAP ModifyRequest messages with "operation" field set to "replace".

Update result messages shall be coded as LDAP ModifyResponse messages.

The LDAP assertion control as specified in IETF RFC 4528 [9] may be used for conditional update operation if the FE knows that the UDR supports the corresponding control.

6.6 Subscribe

Subscribe request messages shall make use of the HTTP Post method and contain a SOAP message envelope.

Subscribe response messages shall be coded as HTTP response message and shall contain a SOAP envelope.

Subscribe request and response messages shall contain a SOAP message envelope header with a header block containing the following elements:

- msgId
  This element uniquely identifies the Subscribe message request – response pair within a connection. The msgId is of type integer. The FE shall allocate the value and use it together with the connId (if any) to correlate a received subscribe response with a sent subscribe request.

- connId
  This optional element identifies the connection between FE and UDR. The connId is of type integer. If used, the

value is allocated by the FE and is used together with the msgId to correlate a subscribe response with a subscribe request.

Optionally, the Subscribe request SOAP message envelope may contain an authentication ws-security header block and a UsernameToken element containing the following elements:

- Username
- Password

An example of a SOAP envelope header for a Subscribe request message is given below:

```
<env:Header>
  <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
   <wsse:UsernameToken wsu:Id="sample"
         xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
    <wsse:Username>http://udfe.example.org:8080/UDRNotificationService
                </wsse:Username>
    <wsse:Password Type="wsse:PasswordText">password</wsse:Password>
                </wsse:UsernameToken>
  </wsse:Security>
  <hb:CorrelationHeader xmlns:hb="urn:headerblock" env:mustUnderstand="true">
    <hb:msgId>25409</hb:msgId>
    <hb:connId>2</hb:connId>
  </hb:CorrelationHeader>
</env:Header>
```

The UDR shall copy the SOAP Envelope CorrelationHeader received in the Subscribe request and send it unmodified in the Subscribe response.

Subscribe request messages shall contain a SOAP message envelope body formatted according to the XML schema defined in Annex A.

Subscribe response messages shall contain an empty SOAP message envelope body, unless SOAP Fault element is included to indicate detailed error information. The HTTP status code in the Subscribe response message is used to indicate success or failure.

A Subscribe request containing several subscriptions shall be managed in an atomic manner. In the event where at least one requestedData elements among several within a Subscribe request is rejected, the Subscribe response shall report a failure and none of the requestedData elements within this Subscribe request shall be accepted by the UDR. A SOAP Fault element may be included to indicate detailed error information relevant to the first error encountered.

6.7 Notify

Notify request messages shall make use of the HTTP Post method and contain a SOAP message envelope.

Notify response messages shall be coded as HTTP response message and shall contain a SOAP message envelope.

Notify request and response messages shall contain a SOAP message envelope header with a header block containing the following elements:

- serviceName
  This optional element identifies a service in the Front End. The value is copied from the Subscribe message or pre-configured in the UDR. The serviceName is of type string with up to 20 characters.
- msgId
  This element uniquely identifies the Notify message request – response pair within a connection. The msgId is of type integer. The UDR shall allocate the value and use it together with the connId (if any) to correlate a received notify response with a sent notify request.
- connId
  This optional element identifies the connection between the UDR and the FE. The connId is of type integer. If used, the value is allocated by the UDR and is used together with the msgId to correlate a notify response with a notify request.

An example of a SOAP envelope header for a Notify request message is given below:

```
<env:Header>
  <hb:CorrelationHeader xmlns:hb="urn:headerblock" env:mustUnderstand="true">
    <hb:serviceName>HLR-FE</hb:serviceName>
    <hb:msgId>787627350</hb:msgId>
    <hb:connId>1</hb:connId>
  </hb:CorrelationHeader>
</env:Header>
```

Application Front Ends shall copy the SOAP Envelope Header received in the Notify request and send it unmodified in the Notify response.

Notify request messages shall contain a SOAP message envelope body formatted according to the XML schema defined in Annex A.

Notify response messages shall contain an empty SOAP message envelope body, unless SOAP Fault element is included to indicate detailed error information. The HTTP status code in the Notify response message is used to indicate success or failure.

6.8 Abandon operation

When a FE wishes to abandon an uncompleted operation towards the UDR, it shall send a LDAP AbandonRequest message.

7 Information Elements

7.1 Information Element Types with LDAP

Information elements and their type to be used in the messages on a given Ud interface for the LDAP Search, Add, Delete and Modify operations are dependent of the application type of the Ud interface. They are described in a LDAP Directory Schema associated to this application type. The general content and structure of a LDAP Directory Schema is described in IETF RFC 4512 [11].

NOTE: The LDAP Directory Schema associated to an application type relates to the application data view mentioned in 3GPP TS 23.335 [10].

When Information elements that are used on a given Ud interface are addressed in 3GPP TS 32.182 [12], their description in the LDAP Directory Schema associated to the application type of this Ud interface shall comply with 3GPP TS 32.182 [12].

7.2 Information Elements for Subscriptions and Notifications

Information elements and their type to be used for subscriptions and notifications about data changes shall also follow the rules specified above in 7.1 and according to the formats/schemas defined in Annex A.

8 Security

Ud interface shall provide security with the methods specified in 3GPP TS 33.210 [15]

Application sensitive data (e.g. permanent authentication keys K/Ki) shall be stored encrypted in the UDR and transferred encrypted over Ud. All FEs (e.g. HLR/HSS/AuC) handling sensitive data, accessing the same UDR and handling the same users (e.g. belonging to the same cluster) shall use the same mechanism and key for decrypting these data. The Application FEs shall support at least AES [17] as a common algorithm, and may support additional algorithms. The key for encryption/decryption is not subscriber-specific.

Editor's note: details of AES (modes of operation, key lengths, data formats, etc.) are to be completed.

Editor's note: statement above must be reviewed and confirmed by SA3

Annex A (normative): SOAP Subscription and Notification

A.1 XML schema for Subscribe Request

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.3gpp.org/udc/subscription" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.3gpp.org/udc/subscription" elementFormDefault="qualified">
    <xs:element name="subscription">
    <xs:complexType>
        <xs:sequence>
           <xs:element name="frontEndID" type="xs:string"/>
           <xs:element name="serviceName" type="xs:string" minOccurs="0"/>
        <xs:element name="originalEntity" type="xs:string" minOccurs="0"/>
        <xs:element ref="requestedData" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="expiryTime" type="xs:dateTime" use="optional"/>
        <xs:attribute name="typeOfSubscription" type="typeOfSubscriptionType"/>
        <xs:attribute name="typeOfNotification" type="typeOfNotificationType"/>
    </xs:complexType>
    </xs:element>
    <xs:element name="requestedData">
    <xs:complexType>
        <xs:sequence>

        <xs:element name="notificationCondition" type="notificationConditionType" minOccurs="1"
maxOccurs="3"/>
        </xs:sequence>
        <xs:attribute name="objectClass" type="xs:string"/>
        <xs:attribute name="DN" type="xs:string"/>
    </xs:complexType>
    </xs:element>

    <xs:simpleType name="notificationConditionType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="add"/>
        <xs:enumeration value="modify"/>
        <xs:enumeration value="delete"/>
    </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="typeOfSubscriptionType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="subscribe"/>
        <xs:enumeration value="unsubscribe"/>
    </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="typeOfNotificationType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="notifyAnyFE"/>
        <xs:enumeration value="notifySubscribingFE"/>
    </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

The serviceName element is optional and, when present, identifies a service in the Front End. The serviceName is stored in the UDR and is used when the Notify message is sent. The serviceName is of type string with up to 20 characters.

A.2 XML schema for Subscribe Response

The SOAP message envelope body shall be empty.

A.3 XML schema for Notify Request

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema  xmlns="http://www.3gpp.org/udc/notification"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
        targetNamespace="http://www.3gpp.org/udc/notification"
        elementFormDefault="qualified">

    <xs:element name="notification">
    <xs:complexType>
        <xs:sequence>
        <xs:element ref="object" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    </xs:element>

    <xs:element name="object">
    <xs:complexType>
        <xs:sequence>
        <xs:element ref="attribute" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="objectClass" type="xs:string"/>
        <xs:attribute name="DN" type="xs:string"/>
        <xs:attribute name="operation" type="operationType" use="required"/>
    </xs:complexType>
    </xs:element>

    <xs:simpleType name="operationType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="add"/>
        <xs:enumeration value="modify"/>
        <xs:enumeration value="delete"/>
        <xs:enumeration value="none"/>
    </xs:restriction>
    </xs:simpleType>

    <xs:element name="attribute">
    <xs:complexType>
        <xs:choice>
        <xs:element name="currentValue" type="xs:string" minOccurs="0"
            maxOccurs="unbounded"/>
        <xs:sequence>
            <xs:element name="beforeValue" type="xs:string" minOccurs="0"
            maxOccurs="unbounded"/>
            <xs:element name="afterValue" type="xs:string" minOccurs="0"
            maxOccurs="unbounded"/>
        </xs:sequence>
        </xs:choice>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="modification" type="modificationType" use="required"/>
    </xs:complexType>
    </xs:element>

    <xs:simpleType name="modificationType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="add"/>
        <xs:enumeration value="replace"/>
        <xs:enumeration value="delete"/>
        <xs:enumeration value="none"/>
    </xs:restriction>
    </xs:simpleType>

</xs:schema>
```

A.4 XML schema for Notify Response

The SOAP message envelope body shall be empty.

Annex B (informative): LDAP Message flows and Transaction flows for UDC

B.1 General LDAP Message flow for UDC

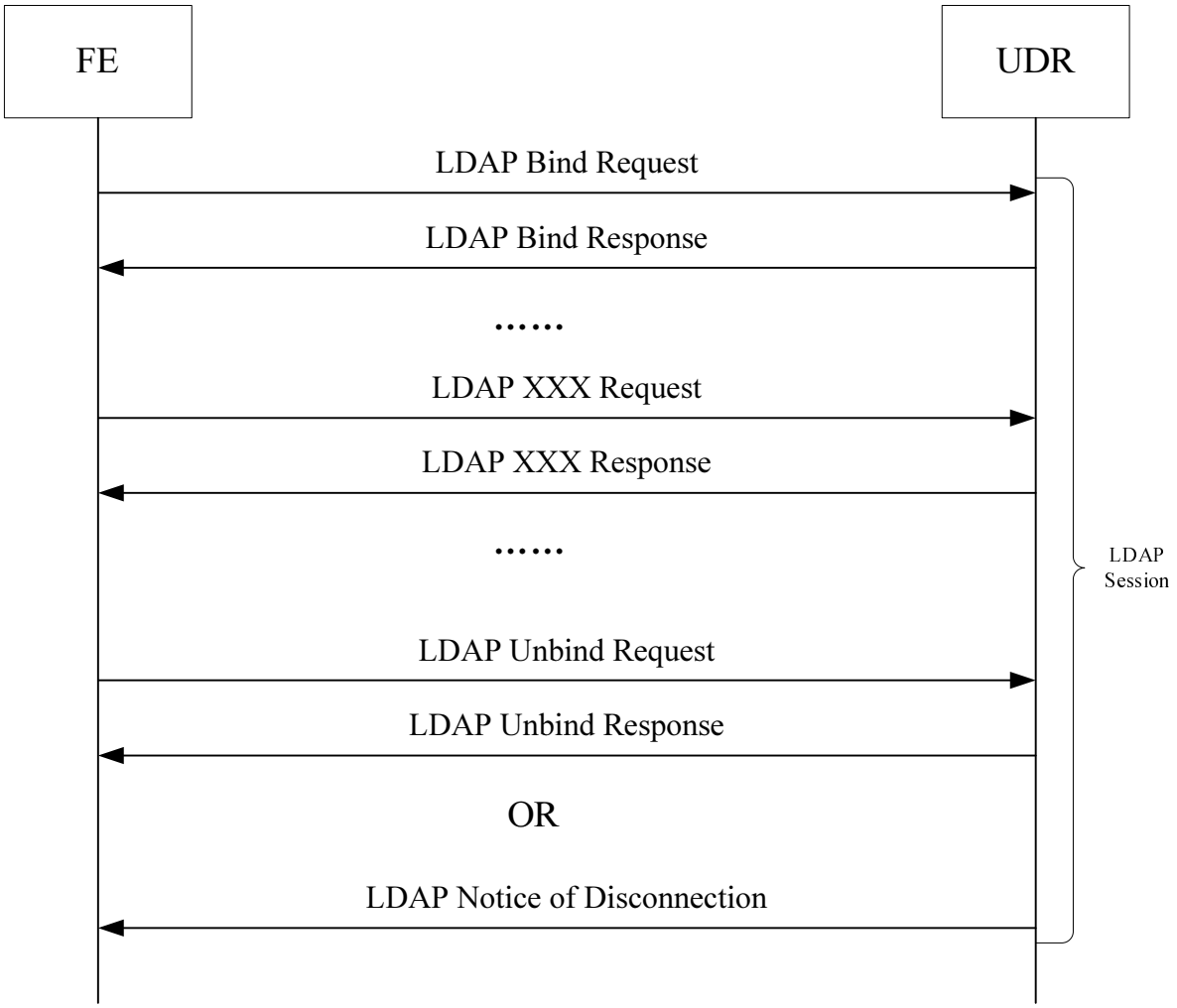

Figure B.1-1. General LDAP Message flow

Figure B.1-1 illustrates the general LDAP message flow associated to a LDAP session for UDC, in compliance with IETF RFC 4511[8].

B.2 LDAP Transaction flows for UDC

Figure B.2-1 illustrates the LDAP transaction flow for UDC. The procedures described in IETF RFC 5805 "Lightweight Directory Access Protocol (LDAP) Transactions" [5] are applied.

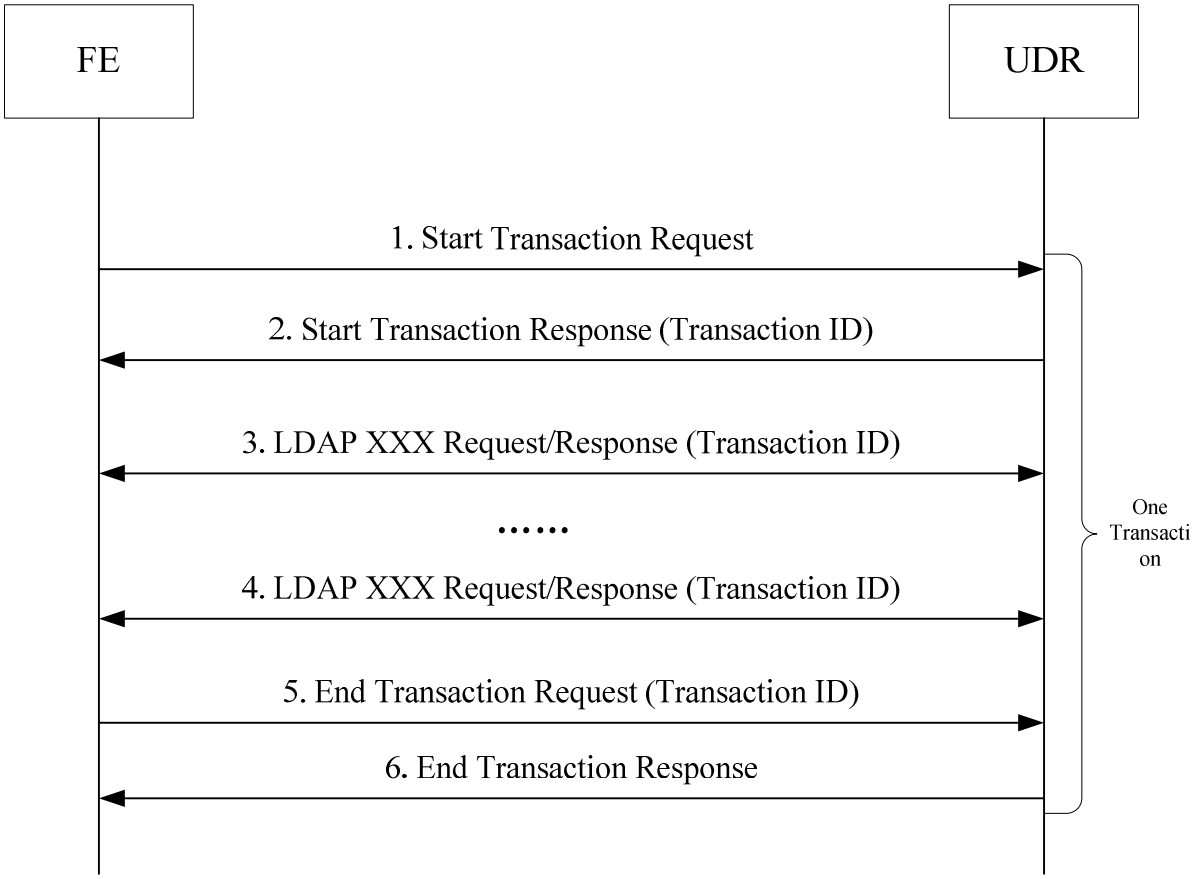

Figure B.2-1. LDAP Transaction flow

1. When the FE needs to bind more than one operation to the UDR in one FE session, the FE issues a Start Transaction Request to UDR to begin a transaction.

2. When receiving a Start Transaction Request, the UDR checks whether the transaction is allowed, e.g. check whether the number of the concurrent transaction exceeds the limit. If the transaction is allowed, the UDR returns a Start Transaction Response with an identity of the transaction assigned by the UDR to the FE and store the transaction identity temporarily until the transaction ends or expires. Otherwise, a Start Transaction Response with the failure code is returned.

3-4.If the result code indicates failure, the following steps are skipped and the procedure terminates. Otherwise, the FE issues the operations to the UDR with the transaction identity received in the Start Transaction Response message from the UDR. The operations with the same transaction identity are in the same transaction.

5. After all or some of the operations are performed, the FE sends an End Transaction Request with the transaction identity to the UDR to commit or rollback the transaction according to the application logic of the FE.

6. When commitment is indicated in the End Transaction Request, the UDR commits all the updates made. If all the updates are committed successfully, a success response is indicated in the End Transaction Response. Otherwise, all the updates are rolled back and the data remains as it was before the transaction and failure is indicated in the End Transaction Response. When rollback is indicated in the End Transaction Request, the UDR cancels all the updates made and return the End Transaction Response with the cancel result to the FE.

Annex C (informative): Messages Based on SOAP

C.1 General

Some Front Ends which do not have high real time performance requirements would prefer to interact with the UDR on the basis of a SOAP protocol for data access (CRUD) as well as for Subscriptions and Notifications (S/N). A way to accomplish this is by means of a protocol converter that converts SOAP based data access messages into LDAP based data access messages. SOAP based Subscription and Notification messages need not be converted. The architectural options are shown in figure C.1/1:

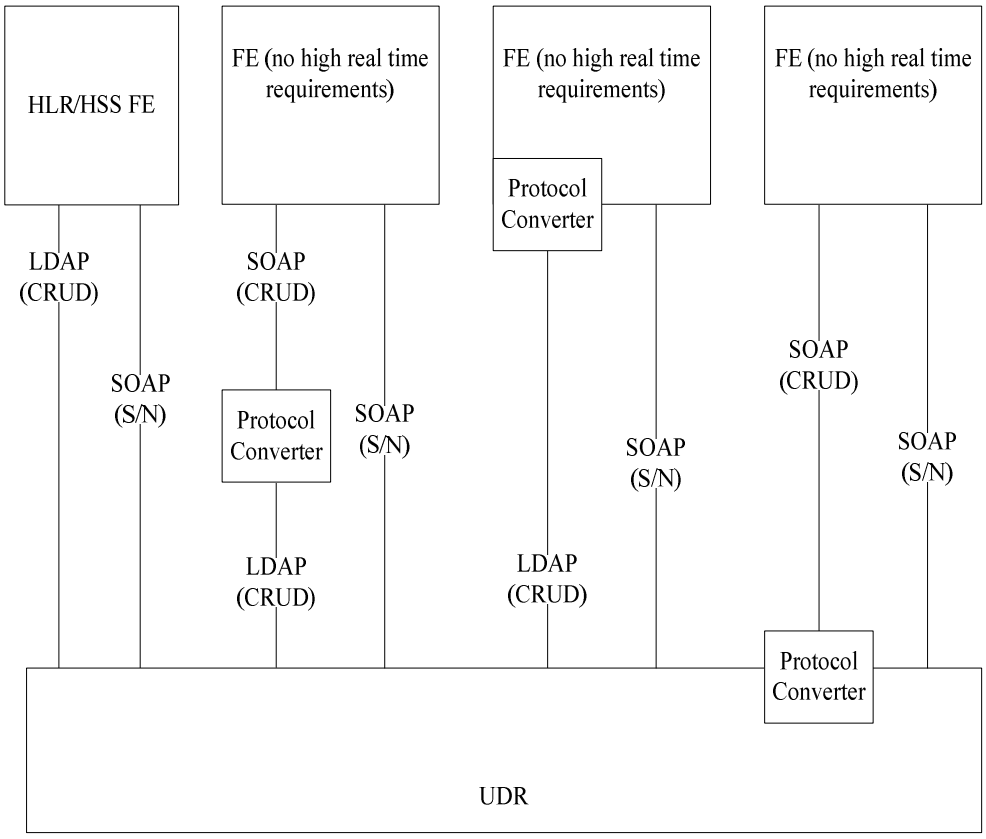

Figure C.1/1: Data access options based on SOAP

As shown in the Figure C.1/1, the Protocol Converter that converts SOAP based data access messages to LDAP based data access messages can be stand alone, be integrated into the FE or be integrated into the UDR. LDAP based data access (CRUD) messages and SOAP based Subscription and Notification (S/N) messages are specified in the normative part of this specification. A description of SOAP based data access messages which are exchanged between FEs without high realtime performance requirements and the Protocol Converter is given in this informative annex.

NOTE: An application that does not have high real time requirements implies that it does not demand neither high values of traffic throughput nor small request/response latency values nor high demanding processing needs.

C.2 Protocol Stack for Messages Based on SOAP

The protocol layering used for messages based on SOAP described in this informative annex is the same as shown in Figure 4.3-1 in the normative part of this specification.

C.3 SOAP Based Query

SOAP Based Query request messages make use of the HTTP Post method and contain a SOAP message envelope.

SOAP Based Query response messages are coded as HTTP response message and contain a SOAP envelope.

Query request messages contain a SOAP message envelope body formatted according to the XML schema defined in OASIS DSML [18] related to SearchRequest.

Query response messages contain a SOAP message envelope body formatted according to the XML schema defined in OASIS DSML [18] related to SearchResultEntry, SearchResultReference, and SearchResultDone. The HTTP status code in the Query response message is used to indicate success or failure.

C.4 SOAP Based Create

SOAP Based Create request messages make use of the HTTP Post method and contain a SOAP message envelope.

SOAP Based Create response messages are coded as HTTP response message and contain a SOAP envelope.

Create request messages contain a SOAP message envelope body formatted according to the XML schema defined in OASIS DSML [18] related to AddRequest or ModifyRequest with the "operation" field set to "add".

Create response messages contain a SOAP message envelope body formatted according to the XML schema defined in OASIS DSML [18] related to AddResponse or ModifyResponse depending on the used request message. The HTTP status code in the Create response message is used to indicate success or failure.

C.5 SOAP Based Delete

SOAP Based Delete request messages make use of the HTTP Post method and contain a SOAP message envelope.

SOAP Based Delete response messages are coded as HTTP response message and contain a SOAP envelope.

Delete request messages contain a SOAP message envelope body formatted according to the XML schema defined in OASIS DSML [20] related to DelRequest or ModifyRequest with the "operation" field set to "delete".

Delete response messages contain a SOAP message envelope body formatted according to the XML schema defined in OASIS DSML [20] related to DelResponse or ModifyResponse depending on the used request message. The HTTP status code in the Delete response message is used to indicate success or failure.

C.6 SOAP Based Update

SOAP Based Update request messages make use of the HTTP Post method and contain a SOAP message envelope.

SOAP Based Update response messages are coded as HTTP response message and contain a SOAP envelope.

Update request messages contain a SOAP message envelope body formatted according to the XML schema defined in OASIS DSML [18] related to ModifyRequest with the "operation" field set to "replace".

Update response messages contain a SOAP message envelope body formatted according to the XML schema defined in OASIS DSML [18] related to ModifyResponse. The HTTP status code in the Update response message is used to indicate success or failure.

C.7 SOAP Based Abandon

SOAP Based Abandon request messages make use of the HTTP Post method and contain a SOAP message envelope.Abandon request messages contain a SOAP message envelope body formatted according to the XML schema defined in OASIS DSML [18] related to AbandonRequest.

Annex D (informative): Change history

Date	TSG #	TSG Doc.	CR	Rev	Cat	Subject/Comment	New
2010-03	CT#47	CP-100052				3GPP TS presented for information and approval.	9.0.0
2010-06	CT#48	CP-100282	0001	1		Removal of not used references	9.1.0
			0002	1		Reference to draft removed	
			0003	2		Removal of security editor's note	
2010-09	CT#49	CP-100459	0004	4		Subscription and Notification Messages	9.2.0
2011-03	CT#51	CP-110055	0007	4		Messages Based on SOAP	10.0.0
2011-06	CT#52	CP-110267	0010	1		UDC Notifications and Service Name	10.1.0
			0012	1		UDC Notifications and original entity	
2012-09	CT#57	CP-120469	0013	1		UDC Data Model in TS 29.335	11.0.0
2012-12	CT#58	CP-120721	0017	-		SOAP Fault in Subscribe and Notify Response messages	11.1.0
2014-09	-	-	-	-		Update to Rel-12 version (MCC)	12.0.0
2015-12	-	-	-	-		Update to Rel-13 version (MCC)	13.0.0
2016-06	CT#72	CP-160235	0019	1		Authentication on the Ud SOAP Interface	13.1.0
2016-06	CT#72	CP-160235	0020	1		SOAP Subscription Request containing several subscriptions error handling	13.1.0
2016-06						References [20] and [21] corrected	13.1.1
2017-03	CT#75	-	-	-		Update to Rel-14 version (MCC)	14.0.0
2018-06	CT#80	-	-	-		Update to Rel-15 version (MCC)	15.0.0
2020-07	CT#88e	-	-	-		Update to Rel-16 version (MCC)	16.0.0
2021-12	CT#94e	CP-213104	0021	-	F	Reference update: removal of RFC 2616	17.0.0
2024-03	-	-	-	-		Update to Rel-18 version (MCC)	18.0.0
2025-10	-	-	-	-		Update to Rel-19 version (MCC)	**19.0.0**

History

Document history		
V19.0.0	October 2025	Publication