ETSI TS 129 240 V16.0.0 (2020-08)

Universal Mobile Telecommunications System (UMTS);
LTE;
3GPP Generic User Profile (GUP);
Stage 3;
Network
(3GPP TS 29.240 version 16.0.0 Release 16)

Reference
RTS/TSGC-0429240vg00

Keywords
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 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is 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 Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, 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.

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 under http://webapp.etsi.org/key/queryform.asp.

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 defines the stage 3 network protocol description to the 3GPP Generic User Profile (GUP), which includes the elements necessary to realise the stage 2 requirements in 3GPP TS 23.240 [1].

The fact of having several domains within the 3GPP mobile system (e.g. Circuit-Switched, Packet-Switched, IP Multimedia Subsystem) and access technologies (e.g. GERAN, UTRAN and WLAN) introduces a wide distribution of data associated with the user. Further, the new functions both in terminals and networks mean that the data related to users, services and user equipment will be increased greatly. This causes difficulties for users, subscribers, network operators and value added service providers to create, access and manage the user-related data located in different entities.

The objective of specifying the 3GPP Generic User Profile is to provide a conceptual description to enable harmonised usage of the user-related information located in different entities. Technically the 3GPP Generic User Profile provides an architecture, data description and interface with mechanisms to handle the data.

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 TS 23.240: "3GPP Generic User Profile - Architecture; Stage 2".

[2] "Web Services Description Language (WSDL) 1.1," Christensen, Erik, Curbera, Francisco, Meredith, Greg, Weerawarana, Sanjiva, eds. World Wide Web Consortium W3C Note (15 March 2001). http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[3] Thompson, H.S., Beech, D., Maloney, M., Mendleshon, N., eds. (May 2002). "XML Schema Part 1: Structures," Recommendation, World Wide Web Consortium.http://www.w3.org/TR/xmlschema-1/

[4] Biron, P.V., Malhotra, A., eds. (May 2002). "XML Schema Part 2: Datatypes," Recommendation, World Wide Web Consortium http://www.w3.org/TR/xmlschema-2/.

[5] "Simple Object Access Protocol (SOAP) 1.1," Box, Don, Ehnebuske, David , Kakivaya, Gopal, Layman, Andrew, Mendelsohn, Noah, Nielsen, Henrik Frystyk, Thatte, Satish, Winer, Dave, eds. World Wide Web Consortium W3C Note (08 May 2000). http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[6] RFC 2616 (June 1999): "Hypertext Transfer Protocol – HTTP/1.1".

[7] 3GPP TS 29.228: "IP Multimedia (IM) Subsystem Cx and Dx interfaces; Signalling flows and message contents".

[8] 3GPP TS 23.008: "Organization of subscriber data".

[9] 3GPP TS 23.228: "IP Multimedia Subsystems (IMS); Stage 2".

[10] 3GPP TS 29.328: "IP Multimedia (IM) Subsystem Sh interface; Signalling flows and message contents".

[11] 3GPP TS 33.102: "3G Security; Security architecture".

[12] 3GPP TS 33.203: "3G security; Access security for IP-based services".

[13] "Liberty ID-WSF Data Services Template Specification", Liberty Alliance Project. http://www.projectliberty.org/specs/draft-liberty-idwsf-dst-v2.0-01.pdf (draft)

[14] "Liberty ID-WSF SOAP Binding Specification", Liberty Alliance Project. http://www.projectliberty.org/specs/liberty-idwsf-soap-binding-v1.1.pdf

[15] "Liberty ID-WSF Security Mechanisms Specification", Liberty Alliance Project. http://www.projectliberty.org/specs/liberty-idwsf-security-mechanisms-v1.0.pdf

[16] IETF RFC 2246: "The TLS Protocol".

[17] "Liberty ID-WSF Discovery Service Specification", Liberty Alliance Project. http://www.projectliberty.org/specs/liberty-idwsf-disco-svc-v1.0.pdf

[18] IETF RFC 2396: "Uniform Resource Identifiers (URI): Generic Syntax".

[19] IETF RFC 3261: "SIP: Session Initiation Protocol".

[20] IETF RFC 2486:" The Network Access Identifier".

[21] 3GPP TS 23.003: "Numbering, addressing and identification".

[22] IETF RFC 2821:"Simple Mail Transfer Protocol".

[23] "Liberty ID-WSF Interaction Service Specification", Liberty Alliance Project. http://www.projectliberty.org/specs/draft-liberty-idwsf-interaction-svc-1.0-errata-v1.0.pdf12

[24] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (January, 2004). "Web Services Security: SOAP Message Security," OASIS Standard V1.0 [OASIS 200401], Organization for the Advancement of Structured Information Standards http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document the following definitions apply:

3GPP Generic User Profile (GUP): The 3GPP Generic User Profile is the collection of user related data which affects the way in which an individual user experiences services and which may be accessed in a standardised manner as described in this specification.

GUP Component: A GUP component is logically an individual part of the Generic User Profile.

Data Element: the indivisible unit of Generic User Profile information.

Data Element Group: A pre-defined set of Data Elements and/or other Data Element Groups closely related to each other. One or more Data Element Groups can constitute the GUP Component.

Data Description Method: A method describing how to define the data contained in the Generic User Profile

3.2 Symbols

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

Rg Reference Point between Applications and the GUP Server.
Rp Reference Point between the GUP Server and GUP Data Repositories, and between Applications and GUP Data Repositories.

3.3 Abbreviations

For the purposes of the present document the following abbreviations apply:

GUP	3GPP Generic User Profile
SOAP	Simple Object Access Protocol
RAF	Repository Access Function

4 Main concept

This specification defines the binding of the GUP interfaces and procedures to SOAP protocol (defined in "Simple Object Access Protocol (SOAP) 1.1" [5]). Each interface is defined in terms of the messages sent and received. The payload of each message is XML, defined using an XML schema language. The framework, procedures, SOAP binding and security solutions of GUP are based on the Liberty Alliance Project work.

Throughout the rest of this specification the SOAP based binding of the 3GPP GUP is described

5 Definition methodology

The definition of the interfaces can be divided into the following sections:

Definition of the operations (WSDL/XML)

Common functions like security, authentication and authorisation (WSDL/XML)

Repository Access Function specific data contents for the operations (XML Schema)

5.1 Protocol layers

The protocol architecture of the Rg reference point is depicted in the figure 1. The essential contents of this specification describe the functionality, semantics and the WSDL/XML definitions of the interfaces. Additionally the special characteristics of the SOAP and http usage are defined. It is worth noting that part of the data is passed in the SOAP headers but the most GUP specific data is placed in the SOAP message body.

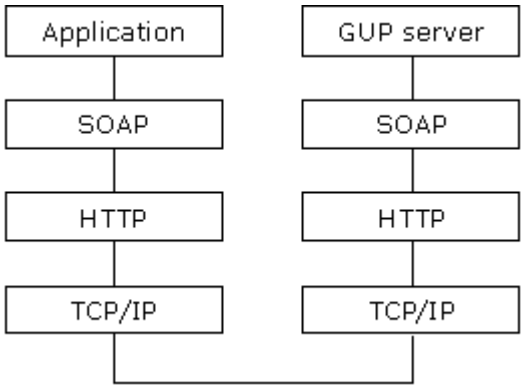

Figure 7.2.1 GUP Rg reference point protocol architecture

The protocol architecture of the Rp reference point is depicted in the figure 2.

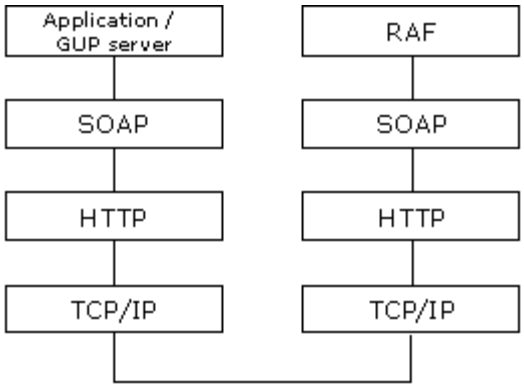

Figure 7.7.2 GUP Rp reference point protocol architecture

The interface between Repository Access Function (RAF) and GUP Data Repository is not defined by the present specification because it is not required to be standardised in 3GPP TS 23.240: "3GPP Generic User Profile - Architecture; Stage 2 [1] The protocol contains following layers:

- Application layer
  - Application level interface specification. All the operations and data are described by XML elements and attributes in an XML Schema and WSDL. The WSDL is defined by W3C in Web Services Description Language (WSDL) 1.1 [2].
  - The standard XML Schema is defined by W3C in "XML Schema Part 1: Structures", Recommendation [3] and "XML Schema Part 2: Datatypes," Recommendation [4].
- SOAP (Session layer)
  - SOAP is an XML based messaging protocol that provides support for remote procedure calls by messaging. The SOAP protocol is defined by W3C in "Simple Object Access Protocol (SOAP) 1.1" [5].
  - A few specific header types are defined for GUP e.g. for message IDs and time stamps.
- HTTP (Transport layer)
  - HTTP defines how messages are transmitted and formatted. HTTP is a stateless protocol, i.e. each command is executed independently. HTTP is defined by RFC 2616 "Hypertext Transfer Protocol – HTTP/1.1" [6].
- TCP/IP (Network layer)
  - TCP/IP handles network communications between network nodes. GUP does not define any special requirements for this layer.

6 General guidelines

The GUP architecture has identified three kinds of entities: (1) client applications, (2) GUP servers and (3) data repositories [point to stage 2]. For the communication between these entities, two interfaces have been defined, namely Rp and Rg.

Communication between GUP entities is performed via the exchange of messages expressed as XML documents. XML documents should include the XML declaration with the version and encoding attributes. The XML documents shall be well-formed and valid. The W3C XML Schema [3,4] is used in GUP to define the structure of valid XML documents.

The implementation of the Rp and Rg interfaces follows the Liberty Alliance Data Service Template specification [Liberty Alliance].

From a Liberty Alliance point of view, GUP servers and data repositories will play the role of Liberty Alliance data services.

6.1 Reusing Liberty Alliance DST

Liberty Alliance Data Service Template [13] specification proposes a framework for web services that offer access to data in general. In the context of GUP, the data services should be restricted to GUP user profile data.

The Data Service Template defines: (1) some abstract definitions about messages that are sent and received by the web service and (2) some guidelines regarding the structure of the data offered through the service.

More concretely, the Liberty Alliance Data Service Template [13] specification offers a set of incomplete XML schemas with placeholders (for data types) that need to be filled based on the nature of the data offered by the data service.

In the context of GUP, we will:

- fill the place holders with some GUP specific data types, and
- add some new messages that are not offered by the Data Service Template specification

The details of this instantiation of the Liberty Alliance Data Service Template for each GUP component are summarized in corresponding check list tables presented in Annex A together with the actual definition of the GUP component.

The guidelines and naming conventions recommended by Liberty alliance when using XML and XML Schemas are directly applicable to GUP.

6.2 Guidelines for XML usage

As described in Liberty ID-WSF Data Services Template Specification [13], the schemas of the different data services, and GUP in particular, should follow a set of guidelines that are included in this specification with the purpose of completeness.

6.3 GUP Specific Naming and Namespaces

The namespace URI for GUP specific XML documents is a 3GPP specific namespace identifier 'http://3gpp' followed by a namespace specific string starting with 'gup' followed by 'ns' and a sub-namespace specific for a certain namespace. (The 'ns' is used to grouping instances of the namespace type of URIs together. Other types of usage of URIs may be defined later.) The sub-namespaces are defined in GUP specifications and/or implementations.

Thus the syntax of the URI for all GUP specific namespaces is: 'http://3gpp/gup/ns/<sub-namespace>'.

The sub-namespace for the GUP Component specific Profile Components consists of the common 'comp' part followed by the component name.

The following namespaces are defined for GUP:

- GUP Profile: 'http://3gpp/gup/ns/profile'
- Common Attributes: 'http://3gpp/gup/ns/common/<name>'
- GUP procedures: 'http://3gpp/gup/ns/proc/<name>'
- GUP Components: 'http://3gpp/gup/ns/comp/<component name>'.

For example an HSS related component URN could be 'http://3gpp/gup/ns/comp/IMSSubscription'.

Editor's note: This section should go into 23.003 when the specification is getting to a stable condition.

6.4 GUP Profile schema

If the Liberty Alliance data service template specifies the interfaces that can be used to access this data, it does not specify the exact nature of the data. In the context of GUP, we need to explicitly define what the subscriber profile data consists of.

3GPP GUP defines a global schema for the XML content of the user profile.

This schema is unique and the same for every user. The schema is defined using W3C XML schemas.

The GUP user profile consists of profile components.

There are many manners to generate the schema for Generic User Profile. But irrespectively of the manner (e.g. one single XML schema vs many schema nested within each other), the schema can always be transformed into a single "canonical" XML schema. A user profile will be a valid instance of this "canonical" schema.
way to define the global schema
will be to define a set of sub-schemas, each with its own namespace. The schema defines a set a single rooted XML documents, each of them being a valid instance of one component defined by the schema. This fact enables easier schema management as applications should not have to worry about the whole schema, but only the parts they are interested in; when one schema component gets modified, only applications using this component should care about the change. Guidelines for the construction of the profile schema

When designing the schema we want to achieve:

- Modularity
- Extensibility
- Readability
- Easy support for versioning
- Isolation (only applications concerned by the schema component should be affected)

The recommended way to design the GUP schema is to split the schema into schema components, each component being defined as a separate XML schema document, with its own namespace.

Components are assembled together as optional content (minOccurs=0, maxOccurs=1) of an `<all>` construct.

Common attributes as defined in Liberty ID-WSF Data Services Template Specification [13] shall be used for GUP.

6.4.1 Example

To illustrate this, it is shown how the the top-level component of the GUP schema could be defined as the "concatenation" of four different sub-schemas.

The component is defined by an XML schema document. The component namespace is defined by the targetNamespace of the schema.

The sub-schemas used in the definition are referenced by (1) their namespace declaration in the `<xsd:schema>` element and (2) by importing their corresponding schema documents (`<xsd:import>`).

Finally, the top-level component is defined by a content-model that concatenates the four sub-schemas under an <xsd:all> contruct. We now illustrate how we can build components out of other components. Let it be assumed that there are four components (c1 … c4), represented by four distinct namespaces (nsc1 … nsc4) , described in four different schema files; they can be put together as follows:

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://3gpp/gup/profile"
            elementFormDefault="unqualified"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:nsc1="http://3gpp/gup/profile/c1"
            xmlns:nsc2="http://3gpp/gup/profile/c2"
            xmlns:nsc3="http://3gpp/gup/profile/c3"
            xmlns:nsc4="http://3gpp/gup/profile/c4">

   <xsd:import namespace="http://3gpp/gup/profile/c1"
               schemaLocation="gup-profile-c1.xsd"/>
   <xsd:import namespace="http://3gpp/gup/profile/c2"
               schemaLocation="gup-profile-c2.xsd"/>
   <xsd:import namespace="http://3gpp/gup/profile/c3"
               schemaLocation="gup-profile-c3.xsd"/>
   <xsd:import namespace="http://3gpp/gup/profile/c4"
               schemaLocation="gup-profile-c4.xsd"/>

   <xsd:element name="Top">
      <xsd:complexType>
         <xsd:all>
            <xsd:element ref="nsc1:c1" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="nsc2:c2" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="nsc3:c3" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="nsc4:c4" minOccurs="0" maxOccurs="1"/>
         </xsd:all>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Note the benefits that we achieve using this way of doing things:

- Modularity: the GUP schema does not consist of a monolithic standalone document, but rather a collection of small schemas that can be designed and maintained independently.
- Extensibility: at every level, it is possible to extend the schema by simply adding one child to the <xsd:all>.
- Readability
- Support for versioning

6.5 Data Referencing Language

For referencing components, the GUP Component Language (GCL) shall be used.

The GCL is a subset of the XPath 1.0 language [XPath reference], as defined below.

Editor's note: we only use the name "GCL" as a way to avoid confusion. In the final version of this document, the name will be skipped or replaced.

Editor's note: we need to define the LA name to identify the GCL for the discovery service. Probably something like: xpath-gup-gcl.

6.5.1 Language Syntax for GCL

The following subset of XPath shall be supported by GCL:

- only the child:: and attribute:: axis of XPath are supported
- predicates are supported

- expressions inside predicates only apply to attribute nodes or element nodes with no children
- expressions inside predicates shall be of the form "node <op> value" where <op> ranges over the usual string and arithmetic operators.
- boolean negation is supported
- ordinal predicates are supported (e.g. [1], [last()], [position()<=2])

NOTE: AND and OR are not part of the syntax because they can be emulated by already existing language constructs: A[exp1 AND exp2] corresponds to A[exp1][exp2]; A[exp1 OR exp2] corresponds to A[exp1] | A[exp2].

6.5.2 Semantics

The semantics of the GCL language can be defined as follows:

1. we evaluate the GCL expression on the document using the XPath semantics leading to a set of nodes
2. for each node in the set, we also include its descendants and its ancestors

This defines a new document, sub-document of the original document.

6.6 GUP metadata

The 3GPP Rp and Rg interfaces define the management of data and metadata for user profile components. For the sake of uniformity, Rp and Rg do not define special procedures for metadata management. Rather, we distinguish between user profile components (or data components) and metadata components.

We define two kinds of metadata components: access control metadata and mapping metadata. The definition of other metadata components (e.g. billing, etc.) is beyond the scope of this specification.

In the following definition, ResourceIDGroup and SelectType are defined by Liberty Alliance.

7 GUP bindings

Liberty messages are designed so that they can be mapped onto various transport or transfer protocols, however Liberty defines and recommends a specific binding to the SOAP protocol, in order to convey such messages.
Following a similar approach, the following section describes how to map GUP messages to the SOAP protocol.
Another bindings are not precluded, but are not described in this specification.

This specification defines the binding of the GUP interfaces and procedures to SOAP protocol (defined in "Simple Object Access Protocol (SOAP) 1.1" [5]). Identically to Liberty Alliance specifications, and as described in Liberty ID-WSF SOAP Binding Specification [14]. GUP procedures address specific aspects of message exchange (such as to which system entity the message is to be sent, message correlation, the mechanics of message exchange, or security context), mainly at the SOAP header part of the message

In order to address the specific aspects of GUP procedures message exchange functionality this specification reuses a number of SOAP header blocks defined as part of the Liberty ID-WSF SOAP Binding Specification described in [14].

Another bindings are not precluded, but are not described in this specification.

7.1 General principles

SOAP provides a mechanism for exchanging structured and typed information between peers using XML. It is a generic protocol which can also be used to carry remote procedure calls. Each SOAP message has an element "Envelope" and its immediate child elements "Header" and "Body". SOAP carries the GUP procedure elements in its body part in compliance with the SOAP standard "Simple Object Access Protocol (SOAP) 1.1" [5]. The GUP Procedure elements are placed immediately below the Body element. If there are several requests or responses, the GUP Procedure elements are carried one after another.

GUP SOAP messages are specified to run over standard http, see "Hypertext Transfer Protocol – HTTP/1.1" [6] as specified in "Simple Object Access Protocol (SOAP) 1.1" [5] but implementations may also support other transport mechanisms. If any SOAP level error is reported, no application data are returned. The used SOAP binding and error reporting mechanisms are defined in Liberty ID-WSF SOAP Binding Specification [14].

Liberty ID-WSF Data Services Template Specification [13] currently defines some generic XML attributes and data types. In general, it is recommended to make use of such information already in place, rather than defining GUP-specific elements to provide the same functionality

Editor's note: we should mention that the "Document" style is preferred for SOAP messages.

3GPP GUP SOAP messages will be declared according to the SOAP "document style" (I.e., 3GPP GUP messages will be placed directly into the body portion of the SOAP envelope, either as-is or encoded). This preference (versus the rpc option) provides a number of benefits as:

- Full use of XML can be done, enabling description and validation of high-level business documents
- A rigid business relationship between Web Services consumers and providers is not required.
- Asynchronous processing is improved
- Object exchange becomes more flexible.

Editor's note: we should provide the corresponding WSDL files in the appendix.

Editor's note: we should mention how security is taken care of (work with SA3).

7.2 GUP headers

Liberty Alliance defines some headers needed when passing around messages. Since Liberty Alliance only defines a SOAP binding, these headers are defined in terms of the SOAP protocol. But this is NOT mandatory, as mentioned in [soap-binding],

"Although this binding [SOAP binding] is the only one given in this specification, other protocols could be used to convey ID-* messages, with appropriateness depending on the protocol selected and the target operational context. "

In the context of GUP, a set of abstract headers are defined. They are needed by the messages exchanged between the various parties, against the Rp and Rg interfaces. The information contained in these headers will be described as XML data even though the binding may decide to map the information using a different syntax (e.g. ASCII, ASN.1, etc.). In the following sections, namespace "S" corresponds to the SOAP namespace.

Normative bindings for the GUP headers to SOAP are provided in the (Annex E).

7.2.1 Correlation header

7.2.1.1 Description

The **correlation** header block provides a means for correlating messages. Message correlation is achieved by using the `messageID` attribute to identify individual messages. Additionally, a message may refer to another message by setting its `refToMessageID` attribute to the value of the `messageID` of the message of interest.

7.2.1.2 Content

The content of the correlation header can be defined using the following type.

```
<xs:complexType name="correlationType">
 <xs:attribute name="messageID" type="IDType" use="required"/>
 <xs:attribute name="refToMessageID" type="IDType" use="optional"/>
 <xs:attribute name="timestamp" type="xs:dateTime" use="required"/>
 <xs:attribute name="id" type="xs:ID" use="optional"/>
 <xs:attribute name="mustUnderstand" type="xs:boolean" use="optional"/>
 <xs:attribute name="actor" type="xs:anyURI" use="optional"/>
</xs:complexType>
```

7.2.2 Provider header

7.2.2.1 Description

This header block provides a means for a sender to claim that it 545
is represented by a given providerID value. The sender may also claim that it is a member of a given affiliation. Such claims are generally verifiable by receivers by looking up these values in the sender's metadata.

7.2.2.2 Content

```
<xs:complexType name="ProviderType">
  <xs:attribute name="providerID" type="xs:anyURI" use="required"/>
  <xs:attribute name="affiliationID" type="xs:anyURI" use="optional"/>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
  <xs:attribute name="mustUnderstand" type="xs:boolean" use="optional"/>
  <xs:attribute name="actor" type="xs:anyURI" use="optional"/>
</xs:complexType>
```

7.2.3 ProcessingContext header

7.2.3.1 Description

This header block may be employed by a sender to signal to a receiver that the latter should add a specific additional facet to the overall *processing context* in which any action(s) are invoked as a result of processing any message also conveyed in the overall message.

7.2.3.2 Content

```
<xs:complexType name="ProcessingContextType">
  <xs:simpleContent>
  <xs:extension base="xs:anyURI">
  <xs:attribute name="id" type="xs:ID" use="optional"/>
  <xs:attribute name="mustUnderstand" type="xs:boolean" use="optional"/>
  <xs:attribute name="actor" type="xs:anyURI" use="optional"/>
  </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

7.2.4 Consent header

7.2.4.1 Description

The Consent header block element MAY be employed by either a requester or a receiver. For example, the Principal may be using a Liberty-enabled client or proxy (common in the wireless world), and in that sort of environment the mobile operator may cause the Principal's terminal (AKA: cell phone) to prompt the principal for consent for some interaction.

7.2.4.2 Content

```
<xs:complexType name="consentType">
  <xs:attribute name="uri" type="xs:anyURI" use="required"/>
  <xs:attribute name="timestamp" type="xs:dateTime" use="optional"/>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
  <xs:attribute name="mustUnderstand" type="xs:Boolean" use="optional"/>
  <xs:attribute name="actor" type="xs:anyURI" use="optional"/>
</xs:complexType>
```

7.2.5 UsageDirective header

7.2.5.1 Description

Participants in the ID-WSF framework may need to indicate the privacy policy associated with a message. To facilitate this, senders, acting as either a client or a server, may add one or more UsageDirective header blocks to the message being sent. A UsageDirective header appearing in a request message expresses *intended usage*. A UsageDirective header appearing in a response expresses *how* the receiver of the response is to use the response data.

7.2.5.2 Content

```
<complexType name="UsageDirectiveType">
  <sequence>
    <xs:any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="ref" type="reference" use="required"/>
  <attribute name="id" type="id" use="optional"/>
  <attribute name="mustUnderstand" type="xs:boolean" use="optional"/>
  <attribute name="actor" type="xs:anyURI" use="optional"/>
</complexType>
```

7.2.6 ServiceInstanceUpdate header

7.2.6.1 Description

It may be necessary for an entity receiving a message to indicate that messages from the sender should be directed to a different endpoint, or that they wish a different credential to be used than was originally specified by the entity for access to the requested resource.

The ServiceInstanceUpdate header allows a message receiver to indicate that a new endpoint, new credentials, or new security mechanisms should be employed by the sender of the message.

The use of this header block allows the sender of the message to convey updates to security tokens, essentially providing a token renewal mechanism. This is not discussed further in this specification.

7.2.6.2 Content

```
<xs:complexType name="ServiceInstanceUpdateType">
  <xs:sequence>
    <xs:element name="SecurityMechID" type="xs:anyURI" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="Credential" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:any namespace="##any" processContents="lax"/>
        </xs:sequence>
        <xs:attribute name="notOnOrAfter" type="xs:dateTime" use="optional"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="Endpoint" type="xs:anyURI" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
  <xs:attribute name="mustUnderstand" type="xs:boolean" use="optional"/>
  <xs:attribute name="actor" type="xs:anyURI" use="optional"/>
</xs:complexType>
```

7.2.7 Timeout header

7.2.7.1 Description

A requesting entity may wish to indicate that they would like a request to be processed within some specified amount of time. Such an entity would indicate their wish via the Timeout header block.

7.2.7.2 Content

```
<xs:complexType name="TimeoutType">
 <xs:attribute name="maxProcessingTime" type="xs:integer" use="required"/>
 <xs:attribute name="id" type="xs:ID" use="optional"/>
 <xs:attribute name="mustUnderstand" type="xs:boolean" use="optional"/>
 <xs:attribute ref="actor" type="xs:anyURI" use="optional"/>
</xs:complexType>
```

7.2.8 CredentialsContext header

7.2.8.1 Description

The receiver of a GUP message might indicate that credentials supplied in the request did not meet its policy in

allowing access to the requested resource. The `<CredentialsContext>` header block allows such policies to be expressed to a GUP requester.

7.2.8.2 Content

```
<xs:complexType name="CredentialsContextType">
  <xs:sequence>
    <xs:element ref="lib:RequestAuthnContext" minOccurs="0"/>
    <xs:element name="SecurityMechID" type="xs:anyURI" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
  <xs:attribute ref="S:mustUnderstand" use="optional"/>
  <xs:attribute ref="S:actor" use="optional"/>
</xs:complexType>
```

7.2.9 wsse:Security header

7.2.9.1 Description

Relevent security information is communicated in the Security header. The details for the specific bindings are provided in the Annexes.

7.2.9.2 Content

The details of the content for the specific binding are provided in the Annexes.

7.2.10 is:UserInteraction header

In some scenarios it might be necessary for GUP Servers and/or RAFs to interact with the owner of GUP component data exposed in order to e.g. ask for explicit end-user consent for the release of the requested information. To this end, GUP messages may include a `<UserInteraction>` header, which controls the possibilities that GUP Servers and/or RAFs have to engage in interactions with end-users when serving a request from a GUP Requestor.

7.2.10.1 Description

7.2.10.2 Content

7.3 GUP Data types

GCL: need to check how the namespaces are handled by XPath. We may need to provide some namespace declarations in the GCL to resolve namespaces.

7.3.1 GUP Resource identifiers

According to Liberty Discovery Service specification [17], the Liberty ResourceId element takes the format of a URI, and it could therefore host different types of identifiers.

In general, the GUP ResourceId shall be used as the Liberty ResourceId and it shall take the form:

"GUP:"<datavalue>"/"<identifier>

Where:

datavalue = The data value of the identifier as specified in the <identifier> part.

identifier = The identifier that is being used in the <datavalue> part of the subscriber identity.

A definitive list of Identifiers are defined in table X below:

Table 9.3.1: Composition of the Subscriber Identity

Identifier	Comment	Example (informative)
IMSI	Subscriber's IMSI as defined in 3GPP TS 23.003 [21].	GUP:234150999999999/IMSI
MSISDN	Subscriber's MSISDN as defined in 3GPP TS 23.003 [21]. The number shall be specified in international format (excluding the "+" symbol).	GUP:447748000000/MSISDN
IMS Private Identity	Subscriber's IMS Private Identity as defined in 3GPP TS 23.003 [21].	GUP:user.name@abc.examplemno.com/IMPI
IMS Public Identity	Subscriber's IMS Public Identity as defined in 3GPP TS 23.003 [21].	GUP:user.name@xyz.examplemno.com/IMPU
E-mail Address	An e-mail address of the subscriber, in the format specified in IETF RFC 2821 [19].	GUP:user.name@mail.examplemno.com/MAILTO
Generic Data Reference	A format which is locally defined by an HPLMN.	GUP:av23asd46fjh230dm/GEN

Editors note: There might be situations where these encoding schemes conflict with syntax of some identifiers (e.g. SIP addresses). This is FFS.

Editors note: The content of this section may be moved to the 3GPP TS 23.003 once this specification is under change control.

In particular, the identifier named "Generic Data Reference" could host an identifier that is meaningless for the application (e.g., a binary piece of data or a pseudonym), but understood by the GUP server. This mechanism would allow that applications are able to reference a specific user, without revealing the user ´s real identity to the application (such as his/her MSISDN etc). Such identifier could be also obfuscated for even better privacy protection, as stated in Liberty DST [13].

7.3.2 GUP SelectType

The `SelectType` is the type that must be defined by any service willing to be instantiated as a data service template (see Annex A). For GUP, we need to distinguish two cases: simple queries (asking for a portion of one given user profile) and list queries (asking of the list of profiles available from data repository). The SelectType consists of a choice between the two cases.

Case 1 (regular queries): the type permits to access portions of the user profile. It consist of a GCL and an optional component name. `GCLType` and `ComponentType` are defined as strings.

Case 2 (list queries): the type represent the search filtering criteria, represented as a string.

```
<xs:complexType name="SelectType">
  <xs:choice>
    <xs:sequence>
      <xs:element name="Component" type="ComponentType" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="GCL" type="GCLType" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
    <xs:element name="SearchFilter" type="SearchFilterType" minOccurs="1"
maxOccurs="1"/>
  </xs:choice>
</xs:complexType>
```

```
<xs:simpleType name="GCLType">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

```
<xs:simpleType name="ComponentType">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

```
<xs:simpleType name="SearchFilterType">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

7.3.3 Access control metadata type

The access control metadata component is defined by type `authorizationType`. It consists of a resource (defined using the `ResourceIDGroup`), the component defined by a GCL expression embedded in a `SelectType`, a list of requestors, an action (read or write) and a condition.

The details of the condition language to be used for access control is left for further study. Therefore, it represented as `<xs:any>` in the `authorizationType`.

```
<xsd:element name="authorization" type="authorizationType"/>

<xsd:complexType name="authorizationType">
  <xsd:sequence>
    <xsd:group ref="ResourceIDGroup" minOccurs="0" maxOccurs="1"/>
     <xsd:element name="SelectType" type="SelectType" maxOccurs="1"
minOccurs="0"/>
     <xsd:element name="requestors" maxOccurs="1" minOccurs="1">
     <xsd:complexType>
         <xsd:sequence>
```

```
            <xsd:element name="requestor" minOccurs="1"
maxOccurs="unbounded">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string"/>
              </xsd:simpleType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="condition">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:any minOccurs="0" maxOccurs="1"
processContents="skip"/>
            </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="action">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value = "read"/>
            <xsd:enumeration value = "write"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element> </xsd:sequence>
</xsd:complexType>
```

Editor's note: for the condition, we should try to align with already existing standards such as XACML.

An example of an authorization component is given below:

```
<authorization>
  <ResourceID>john.smith@3ggp.org</ResourceID>
  <SelectType>
    <GCL>/gup/identity</GCL>
  </SelectType>
  <requestors>
    <requestor>*@3gpp.org</requestor>
  </requestors>
  <condition>
    <rule id="1">true</rule>
  </condition>
  <action>
  </action>
</authorization>
```

7.3.4 Mapping metadata type

A mapping metadata component maps one or more user profile components expressed using the GCL into the RAF that manages this data.

The GCL expression is defined using the `<SelectType>` of Liberty Alliance that we redefine for GUP.

The RAF is defined using the <RAFType>.

```
<xsd:simpleType name="RAFType">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>

<xsd:complexType name="MappingType">
  <xsd:sequence>
    <xsd:group ref="ResourceIDGroup" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="SelectType" type="SelectType" minOccurs="0"
maxOccurs="1"/>
    <!-- <xsd:element ref="SelectType" minOccurs="0" maxOccurs="1"/> -
->
    <xsd:element name="RAF" type="RAFType"
                 minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>


<element name="Mapping" type="MappingType"/>
```

An example of a mapping metadata component is given below:

```
<Mapping>
  <ResourceID>john.smith@3ggp.org</ResourceID>
  <SelectType>
  <GCL>/gup/identity</GCL>
  </SelectType>
  <RAF>ids.3gpp.org</RAF>
</Mapping>
```

7.4 GUP error codes

The following result codes are defined for GUP:

ActionNotAuthorized	ActionNotSupported	AllReturned	ChangeHistoryNotSupported
ChangedSinceReturnsAll	DataTooLong	ExistsAlready	ExtensionNotSupported
Failed	InvalidData	InvalidResourceID	InvalidSelect
MissingNewDataElement	MissingResourceIDElement	MissingSelect	ModifiedSince
NoMoreElements	NoMultipleAllowed	NoMultipleResources	OK
TimeOut	UnexpectedError	RequestorNotAuthorized	InvalidRequestorData

Editor's note: we should mention the ones that are not part of LA.

RequestorNotAuthorized, InvalidRequestorData, InvokeID codes.

For the next section, we should use the table to identify which result codes can be used or not. Use ~~RequestorNotAuthorized~~ to show that the result code is not applicable.

7.5 GUP Message Types

7.5.1 Create messages

7.5.1.1 Overview

The Create message is used by the application to add a new profile component. There may be more than one Create elements in one message.The CreateResponse message provides the result of the procedure. Create request type.

7.5.1.2 Create request type

LA does not offer any message type for creation. We need to provide our own type.

```
<xs:complexType name="CreateType">
  <xs:sequence>
    <xs:group ref="ResourceIDGroup" minOccurs="0"/>
    <xs:element name="CreateItem" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Select" type="SelectType" minOccurs="0"/>
          <xs:element name="NewData" minOccurs="0">
            <xs:complexType>
              <xs:any minOccurs="0" maxOccurs="unbounded" />
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID"/>
      </xs:complexType>
    </xs:element>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID"/>
  <xs:attribute name="itemID" type="IDType"/>
</xs:complexType>
```

7.5.1.3 Create response type

```
<xs:complexType name="CreateResponseType">
  <xs:sequence>
    <xs:element ref="Status"/>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID"/>
  <xs:attribute name="itemIDRef" type="IDReferenceType"/>
  <xs:attribute name="timeStamp" type="xs:dateTime"/>
</xs:complexType>
```

7.5.1.4 Create example

Based on the above types, we can create <Create> and <CreateResponse> elements as follows:

```
<xs:element name="Create" type="CreateType"/>
<xs:element name="CreateResponse" type="CreateResponseType"/>
```

We provide a simple example to illustrate how these elements will be used.

```
<gup:Create itemID="234" xmlns:gup="http://3gpp/gup">
   <ResourceID id="a123"/>
   <CreateItem id="c1">
      <Select>
         <GCL>/gup/MyAddressBook</GCL>
      </Select>
      <NewData>
         <gup:gup/>
      </NewData>
   </CreateItem>
   <CreateItem id="c12">
      <Select>
         <GCL>/gup/MyAddressBook</GCL>
      </Select>
      <NewData>
         <meta:authorization xmlns:meta="http://3gpp/gup/meta">
            <meta:ResourceID></meta:ResourceID>
            <meta:SelectType>
               <meta:GCL></meta:GCL>
            </meta:SelectType>
            <meta:requestors>
               <meta:requestor/>
            </meta:requestors>
            <meta:condition></meta:condition>
            <meta:action></meta:action>
         </meta:authorization>
      </NewData>
   </CreateItem>
</gup:Create>
```

```
  <gup:CreateResponse itemIDRef="234" xmlns="http://3gpp/gup">
    <Status code="OK"/>
  </gup:CreateResponse>
```

7.5.2 Delete messages

7.5.2.1 Overview

7.5.2.2 Delete Request type

LA does not offer any message type for deletion. We need to provide our own type.

```
<xs:complexType name="DeleteType">
    <xs:sequence>
      <xs:group ref="ResourceIDGroup" minOccurs="0"/>
      <xs:element name="DeleteItem" maxOccurs="unbounded">
  <xs:complexType>
  <xs:sequence>
    <xs:element name="Select" type="SelectType" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID"/>
  </xs:complexType>
      </xs:element>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID"/>
    <xs:attribute name="itemID" type="IDType"/>
</xs:complexType>
```

7.5.2.3 Delete Response type

```
<xs:element name="DeleteResponse" type="DeleteResponseType"/>
  <xs:complexType name="DeleteResponseType">
    <xs:sequence>
      <xs:element ref="Status"/>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID"/>
    <xs:attribute name="itemIDRef" type="IDReferenceType"/>
  </xs:complexType>
</xs:schema>
```

7.5.2.4 Delete example

```
<Delete itemID="234">
  <ResourceID id="a1123"/>
  <DeleteItem id="a12">
    <Select>
      <GCL>/gup/MyAddressBook</GCL>
    </Select>
  </DeleteItem>
</Delete>
```

```
<DeleteResponse itemIDRef="234">
  <Status code="OK"/>
</DeleteResponse>
```

7.5.3 Modify messages

7.5.3.1 Overview

7.5.3.2 Modify request type

This type is already defined by LA Data Service Template as `ModifyType` in liberty-idwsf-dst-v2.0-01.xsd.

7.5.3.3 Modify response type

This type is already defined by LA Data Service Template as `ModifyResponseType` in liberty-idwsf-dst-v2.0-01.xsd.

7.5.3.4 Modify example

```
<Modify>
  <ResourceID>sahuguet@lucent.com</ResourceID>
  <Modification id="modif1">
    <Select>
      <GCL>/gup/location</GCL>
    </Select>
    <NewData>
      <MyGup>
        <location>
         <latitude>12</latitude>
          <longitude>12</longitude>
          <precision>-1</precision>
        </location>
      </MyGup>
    </NewData>
  </Modification>
  <Modification id="modif2">
    <Select>
      <GCL>/gup/Identity/phone</GCL>
    </Select>
    <NewData>
      <gup>
        <identity>
          <Phone>908 582 6491</Phone>
        </identity>
      </MyGup>
    </NewData>
  </Modification>
</Modify>
```

A positive response for the first modification request.

```
<ModifyResponse itemIDRef="modif1">
  <Status code="OK"/>
</ModifyResponse>
```

A negative response for the second modification request.

```
<ModifyResponse itemIDRef="modif2">
  <Status code="NOT_OK"/>
</ModifyResponse>
```

Editor's note: This section is to be written in a similar way than clause 8.3.1.

7.5.4 Query messages

7.5.4.1 Overview

7.5.4.2 Query request type

This type is already defined by LA Data Service Template as `QueryType` in liberty-idwsf-dst-v2.0-01.xsd.

7.5.4.3 Query response type

This type is already defined by LA Data Service Template as `QueryResponseType` in liberty-idwsf-dst-v2.0-01.xsd.

7.5.4.4 Query example

```
<Query>
     <ResourceID>john.smith@3gpp.org</ResourceID>
     <QueryItem itemID="a12">
       <Select>
         <GCL>/gup/MyIdentity( FirstName # LastName )</GCL>
       </Select>
     </QueryItem>
     <QueryItem itemID="b23">
       <Select>
         <GCL>/gup/MyStatus</GCL>
       </Select>
     </QueryItem>
   </Query>
```

The answer is:

```
<QueryResponse>
     <Status code="OK"></Status>
     <Data itemIDRef="a12">
       <gup>
         <Mydentity>
           <FirstName>John</FirstName>
           <LastName>Smith</LastName>
         </MyIdentity>
       <gup>
     </Data>
     <Data itemIDRef="b23">
       <gup>
         <MyStatus>
           busy
         </MyStatus>
```

```
        </gup>
      </Data>
  </gup:QueryResponse>
```

7.5.5 Subscribe messages

7.5.5.1 Overview

7.5.5.2 Subscribe request type

This type is already defined by LA Data Service Template as `SubscribeType` in liberty-idwsf-dst-v2.0-01.xsd.

7.5.5.3 Subscribe example

```
<Subscribe>
  <ResourceID id="john.smith"/>
  <Subscription invokeID="1234" starts="2004-08-07T12:00:00" expires="2004-08-
09T12:00:00">
    <Select>
      <GCL>/gup/MyLocation</GCL>
    </Select>
    <Format>ChangedElements</Format>
    <NotifyTo>gupServer@3gpp.org</NotifyTo>
    <NotifyEndedTo>gupServer@3gpp.org</NotifyEndedTo>
    <Type>type of subscription defined by the service</Type>
    <Trigger>type of trigger defined by the service</Trigger>
  </Subscription>
</Subscribe>
```

7.5.6 Unsubscribe message

7.5.6.1 Overview

7.5.6.2 UnSubscribe request type

This type is already defined by LA Data Service Template as `SubscribeType` in liberty-idwsf-dst-v2.0-01.xsd.

7.5.6.3 UnSubscribe example

```
<gup:Subscribe xmlns="http://3gpp/gup">
  <ResourceID id="john.smith"/>
  <Subscription subscriptionID="1234">
    <NotifyTo>gupServer@3gpp.org</NotifyTo>
  </Subscription>
</gup:Subscribe>
```

Editor's note: This section is to be written in a similar way than clause 8.3.1

7.5.7 Notify message

7.5.7.1 Overview

7.5.7.2 Notify response type

This type is already defined by LA Data Service Template as `SubscribeResponseType` in liberty-idwsf-dst-v2.0-01.xsd.

7.5.7.3 Notify example

```
<SubscribeResponse>
  <Status code="OK"/>
  <Notification subscriptionID="1234">
    <Data>
      <gup>
        <MyLocation>
          not available
        </MyLocation>
      </gup>
    </Data>
  </Notification>
</SubscribeResponse>
```

7.5.8 List message

7.5.8.1 Overview

7.5.8.2 List request type

This type is already defined by LA Data Service Template as `QueryType` in liberty-idwsf-dst-v2.0-01.xsd.

7.5.8.3 List response type

This type is already defined by LA Data Service Template as `QueryResponseType` in liberty-idwsf-dst-v2.0-01.xsd.

7.5.8.4 List example

```
<!-- Asking for the list of subscriptions of resource with id = a777. -->
<gup:Query xmlns="http://3gpp/gup">
  <ResourceID id="a777"/>
  <QueryItem itemID="a12">
  <Select>
    <SearchFilter>filtering criteria</SearchFilter>
  </Select>
  </QueryItem>

</gup:QuerySubscriptions>
```

```
<!-- Getting the list of subscriptions -->
<gup:QueryResponse xmlns="http://3gpp/gup">
   <Status code="OK"/>
  <Data itemIDRef="a12">
    <List>
      …
    </List>
  </Data>
</gup:QueryResponse>
```

Editor's note: the details of the result of a List query need to defined. What is the syntax, what is the information returned, etc.?

8 Rp interface

8.1 General Principles

The Rp interface corresponds to the interaction between the GUP server and the GUP data repositories.

8.2 Procedures

The various procedures of the Rp interface are described in terms of the messages defined in chapter 9.

8.2.1 Create Component procedure

8.2.1.1 General description

The Create Component procedure is used by the application to add a new profile component in the contacted repository.

The procedure consists of a request and response defined by the `Create` and `CreateResponse` messages.

8.2.1.2 Detailed Behaviour

The RAF entity checks the request element contents as described in subclause 9.1. In particular the RAF entity checks whether the component can be created for the given subscriber based on the existing policies. There is no requirement for the user profile with the given subscriber identity to exist beforehand. However the subscriber identity (as well as other given data) shall be valid according to the operator's policies and subscriber addressing rules. If the component exists already, the error "InvalidResourceID" is returned.

Next the RAF entity checks the correctness of the provided XML data against the XML Schema related to the given ComponentType. If the data are faulty or missing, the result code "InvalidData" or "MissingNewDataElement" is returned respectively.

Finally, before creating the component instance, the RAF entity checks that the requestor data allow the operation based on the authorisation information attached to the component type and/or to the specific subscriber data. If any given part of the data cannot be created due to lack of authorisation, the Create Component procedure shall fail with the result code "ActionNotAuthorized" or "RequestorNotAuthorized". The latter result code is returned if the failure is caused by the given requestor information. The Create element processing shall also fail in other error cases, but the other Create elements may be handled normally.

If the request message contains several Create elements, but the receiver implementation supports only one in a single message, the result code "NoMultipleResources" is returned. If a single Create element contains several CreateItems, but the receiver implementation supports only one, the result code "NoMultipleAllowed" is returned.

CreateResponse is sent with the status information. If not otherwise stated above, the contents of the Status are as described in subclause 9.1.10.

8.2.1.3 List of result codes

The following result codes may appear in CreateResponse:

- ActionNotAuthorized;
- ActionNotSupported;
- ExtensionNotSupported;
- Failed;
- InvalidData;
- InvalidRequestorData;

- InvalidResourceID;
- MissingNewDataElement;
- MissingResourceIDElement;
- NoMultipleAllowed;
- NoMultipleResources;
- OK;
- RequestorNotAuthorized;
- TimeOut;
- UnexpectedError;

8.2.2 Delete Component procedure

8.2.2.1 General description

The Delete Component procedure is used by the application to delete one or more profile components.

The procedure consists of a request and response defined by the `Delete` and `DeleteResponse` messages.

8.2.2.2 Detailed Behaviour

If the referenced component exists and the deletion is authorized, the component is deleted.

The RAF entity checks the request element contents as described in subclause 9.1. In particular the RAF entity checks whether the component exists. If the component does not exist, the error "InvalidResourceID" is returned.

Finally, before deleting the component instance, the RAF entity checks that the requestor data allow the operation based on the authorisation information attached to the component type and/or to the specific subscriber data. If not authorized, the whole Delete element processing shall fail with the result code "ActionNotAuthorized" or "RequestorNotAuthorized". The latter result code is returned if the failure is caused by the given requestor information. The Delete element processing shall also fail in other error cases, but the other Delete elements may be handled normally.

If the request message contains several Delete elements, but the receiver implementation supports only one in a single message, the result code "NoMultipleResources" is returned. If a single Delete element contains several DeleteItems, but the receiver implementation supports only one, the result code "NoMultipleAllowed" is returned.

DeleteResponse is sent with the status information. If not otherwise stated above, the contents of the Status are as described in subclause 9.1.10.

8.2.2.3 List of result codes

The following result codes may appear in DeleteResponse:

- ActionNotAuthorized
- ActionNotSupported
- ExtensionNotSupported
- Failed
- InvalidRequestorData
- InvalidResourceID
- MissingResourceIDElement

- NoMultipleAllowed
- NoMultipleResources
- OK
- RequestorNotAuthorized
- TimeOut
- UnexpectedError

8.2.3 Modify Data procedure

8.2.3.1 General description

The Modify Data procedure is used by the application to modify profile components. One message may contain several modification request elements and one such element may contain several changes to the profile data of the specific subscriber.

The procedure consists of a request and response defined by the `Modify` and `ModifyResponse` messages.

8.2.3.2 Detailed Behavior

This procedure is defined according to the Liberty ID-WSF Data Services Template Specification [13] and the functionality is exactly as defined by that specification for all those aspects that are clearly determined there. This document is just intending to profile such specification. Note however that the use of the DS is mandatory in GUP just for certain specific scenarios, being optional in another.

The RAF entity checks the request element contents as described in subclause 9.1. If no errors are found and the modification is authorized, the contents of NewData are stored in the place identified by the ResourceID and Select parameters.

The RAF entity verifies that the given data matches with the data structure identified by the Select element. As a generic rule the upper level data structure need not be created beforehand when providing data for the leaf elements. However there may be profile component specific rules about which data are mandated to be provided.

If the request message contains several Modify elements, but the receiver implementation supports only one in a single message, the result code "NoMultipleResources" is returned. If a single Modify element contains several Modification items, but the receiver implementation supports only one, the result code "NoMultipleAllowed" is returned.

If any error is found in the request data, no modifications shall be made to any data as requested by this modification element. However the possible other modify elements in the message shall be processed normally.

ModifyResponse is sent with the status information. If not otherwise stated above, the contents of the contents of the Status are as described in subclause 9.1.10.

8.2.3.3 List of result codes

The following result codes may appear in responses:

ActionNotAuthorized	ActionNotSupported	~~AllReturned~~	ChangeHistoryNotSupported
~~ChangedSinceReturnsAll~~	~~DataTooLong~~	~~ExistsAlready~~	ExtensionNotSupported
Failed	InvalidData	InvalidResourceID	InvalidSelect
MissingNewDataElement	MissingResourceIDElement	MissingSelect	ModifiedSince
~~NoMoreElements~~	NoMultipleAllowed	NoMultipleResources	OK

TimeOut	UnexpectedError	RequestorNotAuthorized	InvalidRequestorData

8.2.4 Read Data procedure

8.2.4.1 General description

The Read Data procedure is used by the application to read profile data. One message may contain several Query elements.

The procedure consists of a request and response defined by the `Query` and `QueryResponse` messages.

8.2.4.2 Detailed Behaviour

This procedure is defined according to the Liberty ID-WSF Data Services Template Specification [13] and the functionality is exactly as defined by that specification for all those aspects that are clearly determined there. This document is just intending to profile such specification. Note however that the use of the DS is mandatory in GUP just for certain specific scenarios, being optional in another.

The RAF entity checks the request element contents as described in subclause 9.1. If the ResourceIDGroup and Select have appropriate values and the query is authorized, the requested data are returned in the response element.

If the request message contains several Query elements, but the receiver implementation supports only one in a single message, the result code "NoMultipleResources" is returned. If a single Query element contains several Query Items, but the receiver implementation supports only one, the result code "NoMultipleAllowed" is returned.

If any error is found in the request data when processing a QueryItem, the data already processed for the other QueryItem's may be returned, but the current QueryItem shall fail without any other results than the status element which shows the reason for the error and indicates the faulty QueryItem. In this case the remaining QueryItems in the Query element shall not be processed. However the previous and next Query elements (if they exist) shall be processed normally.

QueryResponse elements are sent with the data and status information. If not otherwise stated above, the contents of the Status are4 as described in subclause 9.1.10.

8.2.4.3 List of result codes

The following result codes may appear in responses:

ActionNotAuthorized	ActionNotSupported	AllReturned	ChangeHistoryNotSupported
ChangedSinceReturnsAll	~~DataTooLong~~	~~ExistsAlready~~	ExtensionNotSupported
Failed	~~InvalidData~~	InvalidResourceID	InvalidSelect
MissingNewDataElement	MissingResourceIDElement	MissingSelect	~~ModifiedSince~~
~~NoMoreElements~~	NoMultipleAllowed	NoMultipleResources	OK
TimeOut	UnexpectedError	RequestorNotAuthorized	InvalidRequestorData

8.2.5 Subscribe To Data procedure

8.2.5.1 General description

The Subscribe To Data procedure is used by the application to request notifications about changes in the profile component data.

The procedure consists of a request and response defined by the `Subscribe` message.

8.2.5.2 Detailed Behavior

This procedure is defined according to the Liberty ID-WSF Data Services Template Specification [13] and the functionality is exactly as defined by that specification for all those aspects that are clearly determined there. This document is just intending to profile such specification. Note however that the use of the DS is mandatory in GUP just for certain specific scenarios, being optional in another.

The RAF entity checks the request element contents as described in subclause 9.1. If there is no authority to retrieve any data defined by the Select elements, the status shall indicate "Failed" with the second level status code "ActionNotAuthorized" or "RequestorNotAuthorized".

If the FilterData indicates immediate notification, the Notify Data procedure carries the current values for the data defined by the Select element(s). If a time interval is specified in FilterData, the Notify Data procedure is invoked only after that time period has passed after the Subscribe element was received. In this case the Notify element shall contain the current values of all the changed data structures defined by the Select element(s).

The RAF entity creates and returns an InvokeID that shall be unique within the RAF. Once an InvokeID has been deleted because it is no longer required, another InvokeID with the same value can be created, without failing the uniqueness test. The InvokeID is used by the Notify Data procedure to link the notification to the Subscribe element.

A SubscribeResponse element is sent with the status information. If not otherwise stated above, the contents of the Status are as described in subclause 9.1.10.

8.2.5.3 List of result codes

The following result codes may appear in responses:

ActionNotAuthorized	ActionNotSupported	~~AllReturned~~	~~ChangeHistoryNotSupported~~
~~ChangedSinceReturnsAll~~	~~DataTooLong~~	~~ExistsAlready~~	ExtensionNotSupported
Failed	~~InvalidData~~	InvalidResourceID	InvalidSelect
~~MissingNewDataElement~~	MissingResourceIDElement	MissingSelect	~~ModifiedSince~~
~~NoMoreElements~~	NoMultipleAllowed	NoMultipleResources	OK
TimeOut	UnexpectedError	RequestorNotAuthorized	~~InvalidRequestorData~~

- InvalidFilterData where is defined. Not part of LA.

8.2.6 Unsubscribe To Data procedure

8.2.6.1 General description

The Unsubscribe To Data procedure is used by the application to cancel one or more existing subscriptions to notification of data change.

The procedure consists of a request and response defined by the `Subscribe` message.

8.2.6.2 Detailed Behavior

This procedure is defined according to the Liberty ID-WSF Data Services Template Specification [13] and the functionality is exactly as defined by that specification for all those aspects that are clearly determined there. This document is just intending to profile such specification. Note however that the use of the DS is mandatory in GUP just for certain specific scenarios, being optional in another.

The RAF entity checks that whether there is an active subscription for the given InvokeID. If a subscription is active, it will be cancelled. In other cases the error InvalidInvokeID or MissingInvokeID is returned.

An Unsubscribe response element is sent with the status information. If not otherwise stated above, the contents of the Status are as described in subclause 9.1.10.

8.2.6.3 List of result codes

The following result codes may appear in responses:

ActionNotAuthorized	~~ActionNotSupported~~	~~AllReturned~~	~~ChangeHistoryNotSupported~~
~~ChangedSinceReturnsAll~~	~~DataTooLong~~	~~ExistsAlready~~	ExtensionNotSupported
~~Failed~~	~~InvalidData~~	InvalidResourceID	InvalidSelect
~~MissingNewDataElement~~	~~MissingResourceIDElement~~	~~MissingSelect~~	~~ModifiedSince~~
~~NoMoreElements~~	NoMultipleAllowed	NoMultipleResources	OK
TimeOut	UnexpectedError	~~RequestorNotAuthorized~~	InvalidRequestorData

- InvalidInvokeID
- MissingInvokeID
- Not defined by LA.

8.2.7 Notify Data procedure

8.2.7.1 General description

The Notify Data procedure is invoked by the RAF when the data which was identified in Subscribe To Data Changes Notification procedure changes, or when the invoked Subscribe To Data procedure requested immediate sending of the current values of the referenced data. The procedure identifies the changed data and provides the new values.

The procedure consists of a response defined by the `SubscribeResponse` message.

8.2.7.2 Detailed Behaviour

This procedure is defined according to the Liberty ID-WSF Data Services Template Specification [13] and the functionality is exactly as defined by that specification for all those aspects that are clearly determined there. This document is just intending to profile such specification. Note however that the use of the DS is mandatory in GUP just for certain specific scenarios, being optional in another.

The Notify Data procedure is executed when the data for which there is a subscription to change notification have changed in the GUP repository. The application can identify its subscription by the InvokeID element. The element named Data contains the changed data.

When the application has processed this request, it shall immediately return a response message to the RAF.

A NotifyResponse element is sent with the status information. If not otherwise stated above, the contents of the Status are as described in subclause 9.1.10.

Editor's note: To be completed with the handling of error conditions

8.2.7.3 List of result codes

The following result codes may appear in responses:

~~ActionNotAuthorized~~	ActionNotSupported	~~AllReturned~~	~~ChangeHistoryNotSupported~~
~~ChangedSinceReturnsAll~~	~~DataTooLong~~	~~ExistsAlready~~	ExtensionNotSupported
Failed	~~InvalidData~~	~~InvalidResourceID~~	~~InvalidSelect~~
~~MissingNewDataElement~~	~~MissingResourceIDElement~~	~~MissingSelect~~	~~ModifiedSince~~
~~NoMoreElements~~	~~NoMultipleAllowed~~	~~NoMultipleResources~~	OK
TimeOut	UnexpectedError	~~RequestorNotAuthorized~~	~~InvalidRequestorData~~

- InvalidInvokeID
- MissingInvokeID
- Not defined by LA

9 Rg interface

9.1 General principles

The Rg interface corresponds to the interaction between the GUP client application and the GUP server.

9.2 Procedures

The various procedures of the Rg interface are described in terms of the messages defined in chapter 9.

9.2.1 Create Component procedure

9.2.1.1 General description

The Create Component procedure is used by the application to add a new profile component in the contacted repository.

The procedure consists of a request and response defined by the `Create` and `CreateResponse` messages.

9.2.1.2 Detailed Behaviour

The GUP server entity checks the request element contents as described in subclause 9.1. In particular the RAF entity checks whether the component can be created for the given subscriber based on the existing policies. There is no requirement for the user profile with the given subscriber identity to exist beforehand. However the subscriber identity (as well as other given data) shall be valid according to the operator's policies and subscriber addressing rules. If the component exists already, the error "InvalidResourceID" is returned.

Next the RAF entity checks the correctness of the provided XML data against the XML Schema related to the given ComponentType. If the data are faulty or missing, the result code "InvalidData" or "MissingNewDataElement" is returned respectively.

Finally, before creating the component instance, the RAF entity checks that the requestor data allow the operation based on the authorisation information attached to the component type and/or to the specific subscriber data. If any given part of the data cannot be created due to lack of authorisation, the Create Component procedure shall fail with the result code "ActionNotAuthorized" or "RequestorNotAuthorized". The latter result code is returned if the failure is caused by the given requestor information. The Create element processing shall also fail in other error cases, but the other Create elements may be handled normally.

If the request message contains several Create elements, but the receiver implementation supports only one in a single message, the result code "NoMultipleResources" is returned. If a single Create element contains several CreateItems, but the receiver implementation supports only one, the result code "NoMultipleAllowed" is returned.

CreateResponse is sent with the status information. If not otherwise stated above, the contents of the Status are as described in subclause 9.1.10.

9.2.1.3 List of result codes

The following result codes may appear in CreateResponse:

- ActionNotAuthorized;
- ActionNotSupported;
- ExtensionNotSupported;
- Failed;
- InvalidData;
- InvalidRequestorData;

- InvalidResourceID;
- MissingNewDataElement;
- MissingResourceIDElement;
- NoMultipleAllowed;
- NoMultipleResources;
- OK;
- RequestorNotAuthorized;
- TimeOut;
- UnexpectedError;

9.2.2 Delete Component procedure

9.2.2.1 General description

The Delete Component procedure is used by the application to delete one or more profile components.

The procedure consists of a request and response defined by the `Delete` and `DeleteResponse` messages.

9.2.2.2 Detailed Behaviour

If the referenced component exists and the deletion is authorized, the component is deleted.

The GUP server entity checks the request element contents as described in subclause 9.1. In particular the RAF entity checks whether the component exists. If the component does not exist, the error "InvalidResourceID" is returned.

Finally, before deleting the component instance, the RAF entity checks that the requestor data allow the operation based on the authorisation information attached to the component type and/or to the specific subscriber data. If not authorized, the whole Delete element processing shall fail with the result code "ActionNotAuthorized" or "RequestorNotAuthorized". The latter result code is returned if the failure is caused by the given requestor information. The Delete element processing shall also fail in other error cases, but the other Delete elements may be handled normally.

If the request message contains several Delete elements, but the receiver implementation supports only one in a single message, the result code "NoMultipleResources" is returned. If a single Delete element contains several DeleteItems, but the receiver implementation supports only one, the result code "NoMultipleAllowed" is returned.

DeleteResponse is sent with the status information. If not otherwise stated above, the contents of the Status are as described in subclause 9.1.10.

9.2.2.3 List of result codes

The following result codes may appear in DeleteResponse:

- ActionNotAuthorized
- ActionNotSupported
- ExtensionNotSupported
- Failed
- InvalidRequestorData
- InvalidResourceID
- MissingResourceIDElement

- NoMultipleAllowed
- NoMultipleResources
- OK
- RequestorNotAuthorized
- TimeOut
- UnexpectedError

9.2.3 Modify Data procedure

9.2.3.1 General description

The Modify Data procedure is used by the application to modify profile components. One message may contain several modification request elements and one such element may contain several changes to the profile data of the specific subscriber.

The procedure consists of a request and response defined by the `Modify` and `ModifyResponse` messages.

9.2.3.2 Detailed Behavior

This procedure is defined according to the Liberty ID-WSF Data Services Template Specification [13] and the functionality is exactly as defined by that specification for all those aspects that are clearly determined there. This document is just intending to profile such specification. Note however that the use of the DS is mandatory in GUP just for certain specific scenarios, being optional in another.

The GUP server entity checks the request element contents as described in subclause 9.1. If no errors are found and the modification is authorized, the contents of NewData are stored in the place identified by the ResourceID and Select parameters.

The RAF entity verifies that the given data matches with the data structure identified by the Select element. As a generic rule the upper level data structure need not be created beforehand when providing data for the leaf elements. However there may be profile component specific rules about which data are mandated to be provided.

If the request message contains several Modify elements, but the receiver implementation supports only one in a single message, the result code "NoMultipleResources" is returned. If a single Modify element contains several Modification items, but the receiver implementation supports only one, the result code "NoMultipleAllowed" is returned.

If any error is found in the request data, no modifications shall be made to any data as requested by this modification element. However the possible other modify elements in the message shall be processed normally.

ModifyResponse is sent with the status information. If not otherwise stated above, the contents of the contents of the Status are as described in subclause 9.1.10.

9.2.3.3 List of result codes

The following result codes may appear in responses:

- ActionNotAuthorized
- ActionNotSupported
- ChangeHistoryNotSupported
- ExtensionNotSupported
- Failed
- InvalidData
- InvalidRequestorData

- InvalidResourceID
- InvalidSelect
- MissingNewDataElement
- MissingResourceIDElement
- MissingSelect
- ModifiedSince
- NoMultipleAllowed
- NoMultipleResources
- OK
- RequestorNotAuthorized
- TimeOut
- UnexpectedError

9.2.4 Read Data procedure

9.2.4.1 General description

The Read Data procedure is used by the application to read profile data. One message may contain several Query elements.

The procedure consists of a request and response defined by the `Query` and `QueryResponse` messages.

9.2.4.2 Detailed Behaviour

This procedure is defined according to the Liberty ID-WSF Data Services Template Specification [13] and where not otherwise stated the functionality is exactly as defined by that specification. Note however that GUP does not mandate the use of the Discovery service.

The RAF entity checks the request element contents as described in subclause 9.1. If the ResourceIDGroup and Select have appropriate values and the query is authorized, the requested data are returned in the response element.

If the request message contains several Query elements, but the receiver implementation supports only one in a single message, the result code "NoMultipleResources" is returned. If a single Query element contains several Query Items, but the receiver implementation supports only one, the result code "NoMultipleAllowed" is returned.

If any error is found in the request data when processing a QueryItem, the data already processed for the other QueryItem's may be returned, but the current QueryItem shall fail without any other results than the status element which shows the reason for the error and indicates the faulty QueryItem. In this case the remaining QueryItems in the Query element shall not be processed. However the previous and next Query elements (if they exist) shall be processed normally.

QueryResponse elements are sent with the data and status information. If not otherwise stated above, the contents of the Status are as described in subclause 9.1.10.

9.2.4.3 List of result codes

The following result codes may appear in responses:

- ActionNotAuthorized
- ActionNotSupported
- AllReturned

- ChangeHistoryNotSupported
- ChangedSinceReturnsAll
- ExtensionNotSupported
- Failed
- InvalidRequestorData
- InvalidResourceID
- InvalidSelect
- MissingResourceIDElement
- MissingSelect
- NoMultipleAllowed
- NoMultipleResources
- OK
- RequestorNotAuthorized
- TimeOut
- UnexpectedError

9.2.5 Subscribe To Data procedure

9.2.5.1 General description

The Subscribe To Data procedure is used by the application to request notifications about changes in the profile component data.

The procedure consists of a request and response defined by the `Subscribe` message.

9.2.5.2 Detailed Behavior

This procedure is defined according to the Liberty ID-WSF Data Services Template Specification [13] and the functionality is exactly as defined by that specification for all those aspects that are clearly determined there. This document is just intending to profile such specification. Note however that the use of the DS is mandatory in GUP just for certain specific scenarios, being optional in another.

The GUP server entity checks the request element contents as described in subclause 9.1. If there is no authority to retrieve any data defined by the Select elements, the status shall indicate "Failed" with the second level status code "ActionNotAuthorized" or "RequestorNotAuthorized".

If the FilterData indicates immediate notification, the Notify Data procedure carries the current values for the data defined by the Select element(s). If a time interval is specified in FilterData, the Notify Data procedure is invoked only after that time period has passed after the Subscribe element was received. In this case the Notify element shall contain the current values of all the changed data structures defined by the Select element(s).

The RAF entity creates and returns an InvokeID that shall be unique within the RAF. Once an InvokeID has been deleted because it is no longer required, another InvokeID with the same value can be created, without failing the uniqueness test. The InvokeID is used by the Notify Data procedure to link the notification to the Subscribe element.

A SubscribeResponse element is sent with the status information. If not otherwise stated above, the contents of the Status are as described in subclause 9.1.10.

9.2.5.3 List of result codes

The following result codes may appear in responses:

- ActionNotAuthorized
- ActionNotSupported
- ExtensionNotSupported
- Failed
- InvalidFilterData
- InvalidRequestorData
- InvalidResourceID
- InvalidSelect
- MissingResourceIDElement
- MissingSelect
- NoMultipleAllowed
- NoMultipleResources
- OK
- RequestorNotAuthorized
- TimeOut
- UnexpectedError

9.2.6 Unsubscribe To Data procedure

9.2.6.1 General description

The Unsubscribe To Data procedure is used by the application to cancel one or more existing subscriptions to notification of data change.

The procedure consists of a request and response defined by the `Subscribe` message.

9.2.6.2 Detailed Behavior

This procedure is defined according to the Liberty ID-WSF Data Services Template Specification [13] and the functionality is exactly as defined by that specification for all those aspects that are clearly determined there. This document is just intending to profile such specification. Note however that the use of the DS is mandatory in GUP just for certain specific scenarios, being optional in another.

The GUP server entity checks that whether there is an active subscription for the given InvokeID. If a subscription is active, it will be cancelled. In other cases the error InvalidInvokeID or MissingInvokeID is returned.

An Unsubscribe response element is sent with the status information. If not otherwise stated above, the contents of the Status are as described in subclause 9.1.10.

9.2.6.3 List of result codes

The following result codes may appear in responses:

- ActionNotSupported
- ExtensionNotSupported

- Failed
- InvalidInvokeID
- InvalidRequestorData
- MissingInvokeID
- NoMultipleAllowed
- NoMultipleResources
- OK
- TimeOut
- UnexpectedError

9.2.7 Notify Data procedure

9.2.7.1 General description

The Notify Data procedure is invoked by the RAF when the data which was identified in Subscribe To Data Changes Notification procedure changes, or when the invoked Subscribe To Data procedure requested immediate sending of the current values of the referenced data. The procedure identifies the changed data and provides the new values.

The procedure consists of a response defined by the `SubscribeResponse` message.

9.2.7.2 Detailed Behaviour

This procedure is defined according to the Liberty ID-WSF Data Services Template Specification [13] and the functionality is exactly as defined by that specification for all those aspects that are clearly determined there. This document is just intending to profile such specification. Note however that the use of the DS is mandatory in GUP just for certain specific scenarios, being optional in another.

The Notify Data procedure is executed when the data for which there is a subscription to change notification have changed in the GUP repository. The application can identify its subscription by the InvokeID element. The element named Data contains the changed data.

When the application has processed this request, it shall immediately return a response message to the RAF.

A NotifyResponse element is sent with the status information. If not otherwise stated above, the contents of the Status are as described in subclause 9.1.10.

Editor's note: To be completed with the handling of error conditions

9.2.7.3 List of result codes

The following result codes may appear in responses:

- ActionNotSupported
- ExtensionNotSupported
- Failed
- InvalidData
- InvalidInvokeID
- MissingInvokeID
- OK
- TimeOut

- UnexpectedError

10 Authentication, authorization and security

10.1 Principles

GUP Architecture and Protocols include secure mechanisms for the transfer of User Profile data to, from and between authorised entities. Access to User Profile data shall only be permitted in an authorised and secure manner.

Security mechanisms to be applied over Rg and Rp reference points are defined by Liberty ID-WSF Security Mechanisms [15] specification.

In addition to managing the registration and discovery of GUP Server and profile information a **Discovery Service** (DS) as defined in Liberty ID-WSF Discovery Service Specification [17] may also act as a centralized policy information and decision point issuing the necessary Authentication and/or Authorization statements.

Interactions of GUP requestors towards the Discovery Service shall be as described in Liberty ID-WSF Discovery Service Specification [17].

The GUP architecture with respect to Authentication, Authorization and Security is summarized in Figure XXX.

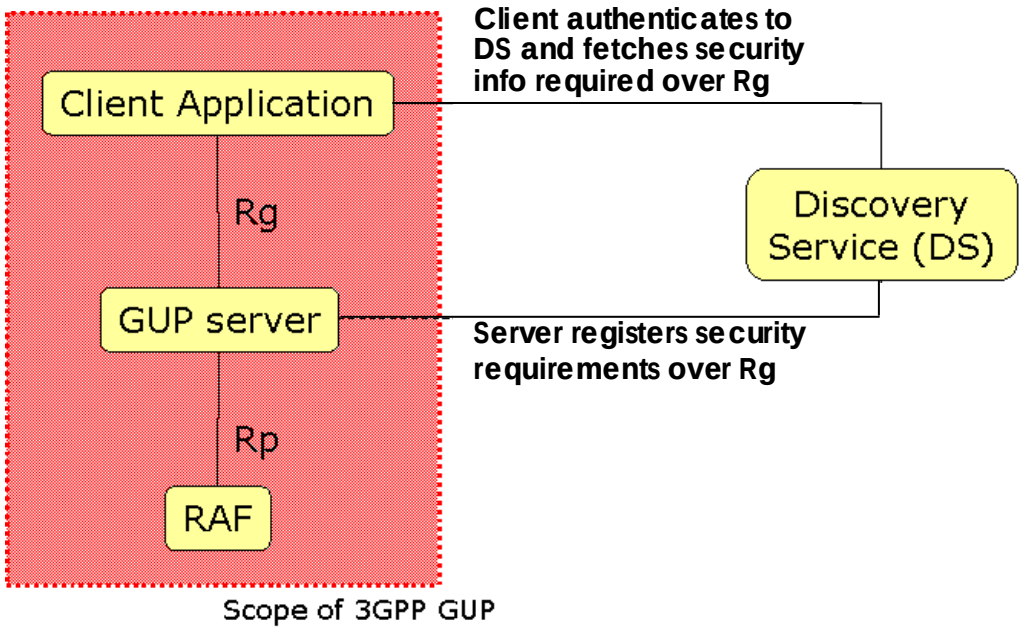

Figure 1: GUP Authentication, Authorization and Security architecture.

Liberty ID-WSF Security Mechanisms Specification [15], Liberty ID-WSF SOAP Binding Specification [14] and Liberty ID-WSF Discovery Service Specification [17] provide normative content for the implementation of the security mechanisms that apply to GUP.

The following chapters under this section of the specification define how the different security mechanisms described in the referred Liberty specifications shall be used in the context of GUP.

The following chapters specify the required set of security mechanisms that shall be supported for GUP.

10.2 Security and Authentication

10.2.1 Rg interface (client application / GUP server)

GUP specifications shall be applicable for multiple deployment scenarios where different security, privacy and trust considerations apply. The applications that may apply GUP Rg reference point may be targeted for different purposes e.g. third party applications for value added services or operator´s own applications for subscription management.

Liberty Alliance Security Mechanisms [15] specification defines a set of combinations of peer authentication and message authentication mechanisms necessary to accommodate various deployment scenarios. Each combination is defined by a URI with the form *urn:liberty:security:2003-08:peer mechanism:message mechanism*. It is up to the security policy of the operator to choose which combination of methods to apply taking into account the security domains where the client and server reside.

10.2.1.1 Peer Entity Authentication and Transport Layer Channel Protection

The Peer entity authentication mechanisms required for GUP Rg reference point are based on the mechanisms prescribed by Liberty ID-WSF Security Mechanisms [15] specification, which rely upon the inherent security properties supplied by SSL 3.0 [SSL] or TLS 1.0 [RFC2246] sometimes referred to as transport-level security (also including means for its confidentiality and integrity protection).

Confidentiality and Integrity at the transport channel is ensured making use of suitable SSL/TLS cipher suites (see sub-clause 12.2.3 for list of supported cipher suites).

The server is authenticated to the client using a x.509 server-side certificate. In general the support of client-side certificates in the context of GUP is not mandated but mutual authentication of the communicating peers may be also supported.

10.2.1.2 Message Authentication

Third party applications external to the Mobile Network Operator domain shall only apply Rg reference point. More precisely, third party applications external to the Mobile Network Operator domain acting as a GUP Requestor over the Rg reference point shall at least support *urn:liberty:security:2003-08:TLS:SAML* as defined by Liberty Alliance Security Mechanisms [15] Specification. The support of other message authentication mechanisms (e.g. *urn:liberty:security:2004-04:TLS:Bearer* and/or *urn:liberty:security:2003-08:TLS:X509*) is optional for third party applications external to the Mobile Network Operator domain. However the use of the *null* message authentication profile (e.g. *urn:liberty:security:2003-08:TLS:null*) is not recommended for the Rg reference point.

Internal applications to the Mobile Network Operator domain acting as a GUP Requestor over the Rg reference point shall at least support *urn:liberty:security:2003-08:TLS:null* as defined by Liberty Alliance Security Mechanisms [15] Specification. The support of other message authentication mechanisms (e.g. *urn:liberty:security:2003-08:TLS:SAML*, *urn:liberty:security:2004-04:TLS:Bearer* and/or *urn:liberty:security:2003-08:TLS:X509*) is optional for Internal applications to the Mobile Network Operator domain.

GUP Server shall at least support *urn:liberty:security:2003-08:TLS:null* and *urn:liberty:security:2003-08:TLS:SAML* as defined by Liberty Alliance Security Mechanisms [15] Specification. The support of other message authentication mechanisms (e.g. *urn:liberty:security:2004-04:TLS:Bearer* and/or *urn:liberty:security:2003-08:TLS:X509*) is optional for the GUP Server.

GUP requestors over Rg reference point may utilise a discovery service as a Trusted Authority providing essential security related information (e.g. preferences in terms of peer entity and message authentication mechanism to be used and authentication and/or authorization assertions). Different policies may be followed in the use of the services of a discovery service. It may be used by different applications in different ways: per each operation, occasionally or not at all. In general terms, third party applications belonging to external security domains shall need to use the services of a discovery service as a normal step, but in operator's services it may not be needed at all.

A Discovery Service as defined by Liberty Alliance Discovery Service [17] specification is able to inform GUP requestors of the peer entity and message authentication mechanisms to be used. Additionally, and in the event that a particular deployment of GUP requires the use of the Web Services SAML security profile (e.g. *urn:liberty:security:2003-08:TLS:SAML* or *urn:liberty:security:2003-08:ClientTLS:SAML)* the Discovery Service shall provide the necessary Authentication assertions as defined by Liberty Alliance Security Mechanisms [15] specification. A Discovery Service may also be capable of providing necessary bearer tokens in the event the Bearer token security profile (e.g. *urn:liberty:security:2004-04:TLS:Bearer* or *urn:liberty:security:2004-04:ClientTLS:Bearer*) is used.

10.2.2 Rp interface (GUP server / RAF)

The required security mechanism for the Rp interface is **urn:liberty:security:2003-08:ClientTLS:null**.

- The server and the RAF communicate over a secure channel protected by SSL/TLS (see 12.2.3 for list of supported cipher suites).

NOTE: Since the number of RAFs is limited and the connections between the server and the RAFs are long-lived (multiple requests sent on the same connections), this should not create too much overhead for either key management or cryptographic processing.

The support of other message authentication mechanisms (e.g. urn:liberty:security:2003-08:TLS:SAML, urn:liberty:security:2004-04:TLS:Bearer and/or urn:liberty:security:2003-08:TLS:X509) is optional for the Rp reference point.

10.2.3 Cryptographic requirements

For signing and verification of protocol messages, communicating entities SHOULD use certificates and private keys that are distinct from the certificates and private keys applied for SSL or TLS channel protection (in accordance to Liberty Alliance Security Mechanisms [15] specification)

The cipher suites to be used for peer-wise encryption are:

- TLS_RSA_WITH_RC4_128_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_CBC_SHA
- TLS_DHE_DSS_WITH_AES_CBC_SHA

GUP entities shall at least support TLS_RSA_WITH_3DES_EDE_CBC_SHA and TLS_RSA_WITH_AES_CBC_SHA cipher suites. The support of other cipher suites is optional.

10.2.4 End-to-End Example (informative)

The following diagram represents a possible interaction between a client application and a GUP server.

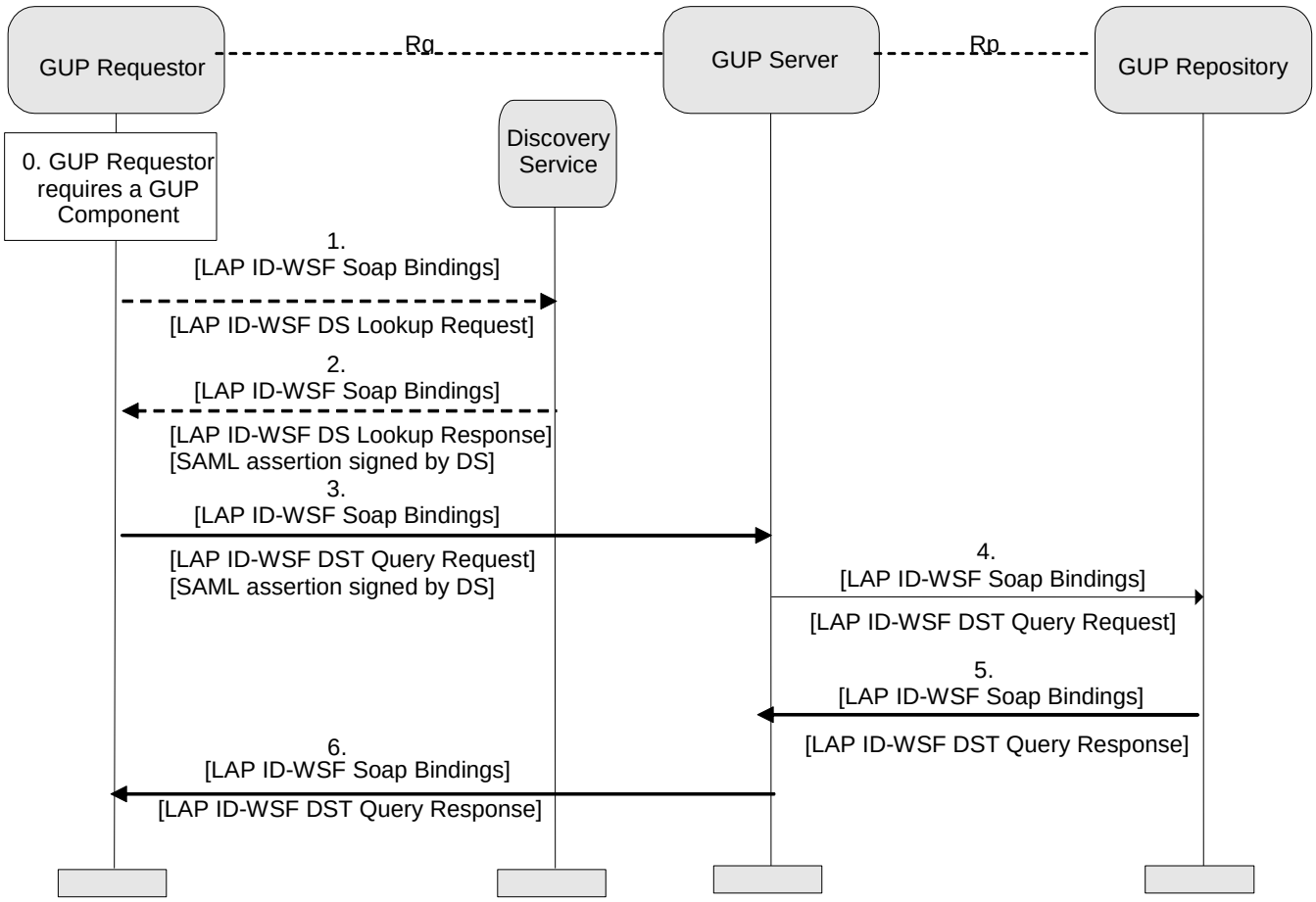

Figure 2: GUP security flow

10.2.5 Example of GUP wsse:Security header (informative)

The following header authenticates the application defined by `application1@3gpp.org`.

```
<<wsse:Security>

<saml:Assertion

xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"

MajorVersion="1" MinorVersion="0"

AssertionID="2sxJu9g/vvLG9sAN9bKp/8q 0NKU="

Issuer="DS.example.com"

IssueInstant="2004-04-01T16:58:33.173Z">


<!-- By placing an audience restriction on the assertion we

can limit the scope of which entity should consume

the information in the assertion. -->


<saml:Conditions
```

```
NotBefore="2004-04-01T16:57:20Z"
NotOnOrAfter="2004-04-01T21:42:43 Z">
<saml:AudienceRestrictionCondition>
<saml:Audience>http://gupserver.com</sa ml:Audience>
</saml:AudienceRestrictionConditi on>
</saml:Conditions>

<!-- The AuthenticationStatement carries information
that describes the identity of the entity this assertion
was issued to (the Subject) and the method the Subject
authenticated to the assertion issuing authority -->

<saml:AuthenticationStatement
AuthenticationMethod="urn:ietf:rfc:2246"
AuthenticationInstant="2004-04-01T16:57:30.000Z">
<saml:Subject>
<saml:NameIdentifier
format="urn:liberty:iff:nameid:entityID"
NameQualifier="http://AffiliationStation.com /">
http:// application1@3gpp.org
</saml:NameIdentifier>
</saml:Subject>
</saml:AuthenticationStatement>
<!-- signature by the authority over the assertion -->
<ds:Signature>...</ds:Signature>
</saml:Assertion>

<!-- this is the reference to the bearer token -->
<wsse:SecurityTokenReference>
<wsse:Reference URI="#2sxJu9g/vvLG9sAN9bKp/8q0NKU="
ValueType="saml:Assertion" />
</wsse:SecurityTokenReference>
</wsse:Security>
```

10.3 Authorization

10.3.1 Principles

Liberty ID-WSF Security Mechanisms Specification [15] defines OPTIONAL mechanisms to convey authorization and resource access information (supplied by a Trusted Authority), which may be necessary to access a service. Authentication and Authorization Authorities may be co-located.

In the context of GUP and when the operator's policies so require, it is recommended that a Discovery Service as described by Liberty ID-WSF Discovery Service Specification [17] also acts as a Trusted Authority issuing Authorization assertions.

In general, GUP authorization can be seen to consist of the following functional components:

- Management of authorization related data (authorization rules) which also refer to actual authorized data
- Access/storage of the authorization rules
- Encapsulation of a set of authorization rules (privacy policy) together with the transported actual data.
- Execution of authorization logic based on the pre-defined authorization rules and information received in the request or otherwise related to the request

The functional components are discussed more in the following subclauses.

10.3.2 Authorization related data

10.3.2.1 Authorization attributes

Authorization attributes are defined to be used as basic input elements of the authorization rules and thus as a basis for the authorization decisions. It is not required that all the authorization attributes defined hereafter are included in a request.

The following types of authorization attributes are considered:

- Identity of the target subscriber (or a group of subscribers) – the GUP subscriber;
- Component type being accessed and more detailed data reference;
- Identity of the requestor (application ID and end-user ID) or group of requestors;
- The Web Services Security SAML Profile [wss-saml] shall be used as the means by which GUP requestor over Rg and/or Rp authenticate to the recipient (i.e. GUP Server and GUP Repository respectively). Each communicating peer performs message level authentication by demonstrating proof of possession of a subject confirmation key. The assertion issuer (i.e. Discovery Service) binds the subject confirmation key to the assertion by signing the assertion. This attestation assures the consumer of the assertion that the subject confirmation key is that of the intended sender. Thus the senders subject confirmation key can be recognized by the recipient as belonging to the remote peer.
- Requestor related data received in the request as Authorization Assertions

The authorization model supports the issuance of assertions, which convey information regarding the resource to be accessed, the entity attempting to access the resource, the mechanism by which the accessing entity must use to demonstrate its identity to the recipient and the ability for the accessing entity to access the resource on behalf of another system entity. Thus the authorization model supports a constrained proxy mechanism, which permits a proxy (the sender) to access the resource on behalf of some other system entity.

Processing rules for the generation of authorization assertions is normatively defined at Liberty ID-WSF Security Mechanisms Specification [15].

- Requested Operation (Query, Modify, Create, Delete, Subscribe, List);
- Other attributes (e.g. the time schedule) related to the request case;

- Privacy Policy included in the request
- Liberty ID-WSF SOAP Binding Specification [14] defines a <UsageDirective> SOAP Header Block in which GUP requestors could express policy in terms of intent of usage of the requested data if released.
- Other data relevant for access control included in the request
  - Liberty ID-WSF SOAP Binding Specification [14] defines a <consent> SOAP Header Block in which GUP requestors could indicate whether the end-user consented the request.
  - Identity information related to active proxies in the communicating channel may be conveyed in `<ProxySubject>`, `<ProxyTransitedStatement>` and `<ProxyInfoConfirmationData>` schema elements within the `<wsse:Security>` SOAP Security Header as defined in Liberty ID-WSF Security Mechanisms Specification [15] specification.
  - Session status of the communicating peers may be conveyed in `<SessionContext>` and `<SessionContextStatement>` schema elements within the `<wsse:Security>` SOAP Security Header as defined in Liberty ID-WSF Security Mechanisms Specification [15] specification.
  - Resource access information is captured in a `<ResourceAccessStatement>` schema elements within the `<wsse:Security>` SOAP Security Header as defined in Liberty ID-WSF Security Mechanisms Specification [15] specification. The purpose of this statement is to convey sufficient information regarding the accessing entity and the resource for which access is being attempted.

10.3.2.2 Authorization rules

The basic set of authorization rules define to whom, to which part of data and for which purpose the authorization is given. All or some of the Authorization data considered above may be applicable.

GUP requestors over Rg and/or Rp reference points shall include relevant authorization data in the SOAP message requests. In particular Authorization assertions shall be conveyed following the processing rules defined at Liberty ID-WSF Security Mechanisms Specification [15].

The authorization rules consist of the following built-in elements:

- authorization attributes and/or references to pre-defined (commonly used) contents of attributes
- references to the actual GUP data
- actions (e.g. the decision, encapsulation of the privacy policy).

10.3.2.3 Management of authorization related data

The GUP procedures defined for the Rp and Rg reference points (see the clauses 8 and 9) may be used for managing authorization related data (using GUP metadata components).

10.3.3 Execution of authorization logic

The authorization logic compares the information conveyed in the request (and possible additional information related to the moment when the request is received) with the information defined in the authorization rules. When the logic leads to a decision to accept the request in principle, the requested data is compared with the authorized set of data. If discrepances are found a GUP authorization entity may either restrict to reply to the authorized content or deny the request. Note that this depends on the policy in the GUP authorization entity and the nature of the request.

10.3.4 Roles of GUP entities related to the authorization

Figure 10.1 shows the logical entities involved in GUP authorization.

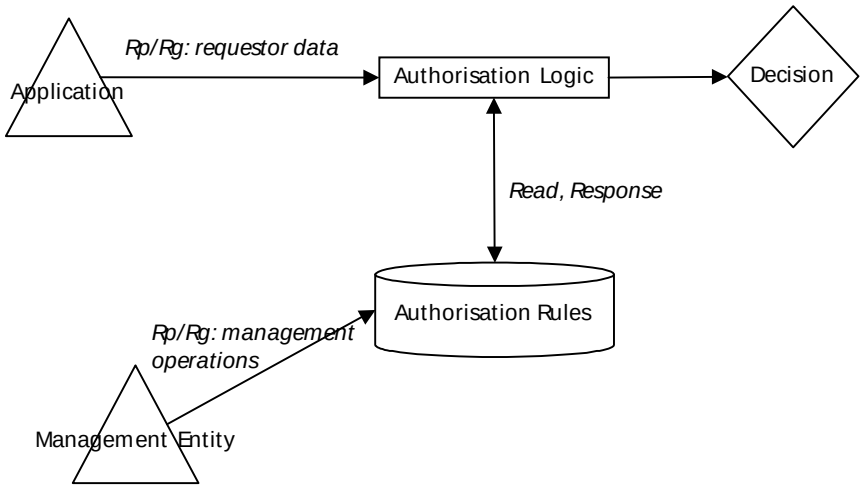

Figure 12.3.4.1. Logical entities of GUP authorization

10.3.4.1 GUP Requestor

GUP requestors over Rg and/or Rp reference points shall include relevant authorization data in the SOAP message requests. In particular, Authorization assertions issued and received from a Discovery Service as defined in Liberty ID-WSF Discovery Service Specification [17] shall be conveyed following the processing rules defined at Liberty ID-WSF Security Mechanisms Specification [15].

Other relevant authorization attributes as defined in chapter 12.2.2.1 of this specification may be also present in the requests.

10.3.4.2 GUP server and RAF

The GUP Server and RAF (including the Data Repository) have access to the authorization rules. The interface between the authorization related data storage and GUP Server (and RAF) is out of the scope of this specification. Note that the GUP Server and RAF may act as data storage entities. The GUP Server or the RAF (or both of them) are responsibles for the execution of the authorization logic.

The authorization issues handled by the GUP Server typically concern the GUP profile and GUP Component level issues; the authorization handled by RAF may be based on more detailed data references (items inside a GUP component). The set of authorization attributes specified for GUP can be used by both the GUP Server and the RAF.

The GUP Server and the RAF may also add to the transported actual data, a <UsageDirective> SOAP Header Block as defined in Liberty ID-WSF SOAP Binding Specification [14] including privacy policies that the GUP requestor shall follow when using, storing and/or distributing the received data. If during the execution of the authorization logic, GUP Server or RAF find out that further authorization information from the end-user is required in order to decide whether or not the request shall be served, the user interaction mechanisms defined in Liberty Alliance Interaction Service [22] specification may be used.

10.3.4.3 Management entity

The authorization rules can be managed by the authorized entities: e.g

- the entity administrating the GUP Data Storage;
- the RAF and/or GUP Server;

- the GUP Subscriber itself.

The GUP Subscriber is normally allowed to manage a limited set of his own user profile data, e.g., certain GUP Components or certain data inside a GUP Component. Additionally, there might be restrictions on the allowed operations.

The entity administering the GUP Data Storage may define common or default authorization rules for GUP Subscribers. The entity may also pre-define contents for authorization attributes, e.g. user groups, to which the authorization rules can refer. The entity administering the GUP Data Storage may also manage GUP Subscriber specific authorization rules (e.g., on behalf of the GUP Subscriber).

Annex A (normative): Component Data Definitions

According to Section 5 of TS23.240 [1], a Generic User Profile consists of a number of independent GUP Components. This Annex defines the content, semantics and specific processing rules of the different GUP Data Components.

Note: For Rel-6 only the HSS IMS GUP component is defined. Further releases of this specification will define additional components.

A.1 HSS IMS GUP Component definition

The purpose of this annex is to provide GUP Profile Component definition for the IMS data of the HSS.

A.1.1 General description

The following Figure A.3 gives an outline of the UML model of the logical view of the HSS IMS GUP Components. The main Component is called HSSIMSData.

Each element in that figure represents a part of the XML Schema structure, either a GUP Component or a lower level block of data contained in a GUP Component. Elements marked with the same background colour make up an independently manageable GUP Component, whose root is marked with '<<GUP Component>>'.

All HSS IMS GUP Components can be managed with the procedures provided by GUP.

Component and attribute naming has been taken mainly from 3GPP TS 23.008 [8] where appropriate. The details about the specific attributes shown in the picture are described in others specs. The ones related Cx protocol are described in the 3GPP TS 29.228 [7] and the ones related Sh interface are described in the 3GPP TS 29.328 [10].

The identified set of GUP data elements with regard to the IMS HSS component should receive different treatment (only read or both read/write access rights), depending on the nature of the data and the nature of the application requesting the data (e.g. provisioning application). In order to provide such differentiated treatment access control mechanisms shall be applied. These access control mechanisms should take into consideration the rights that can be offered over each attribute.

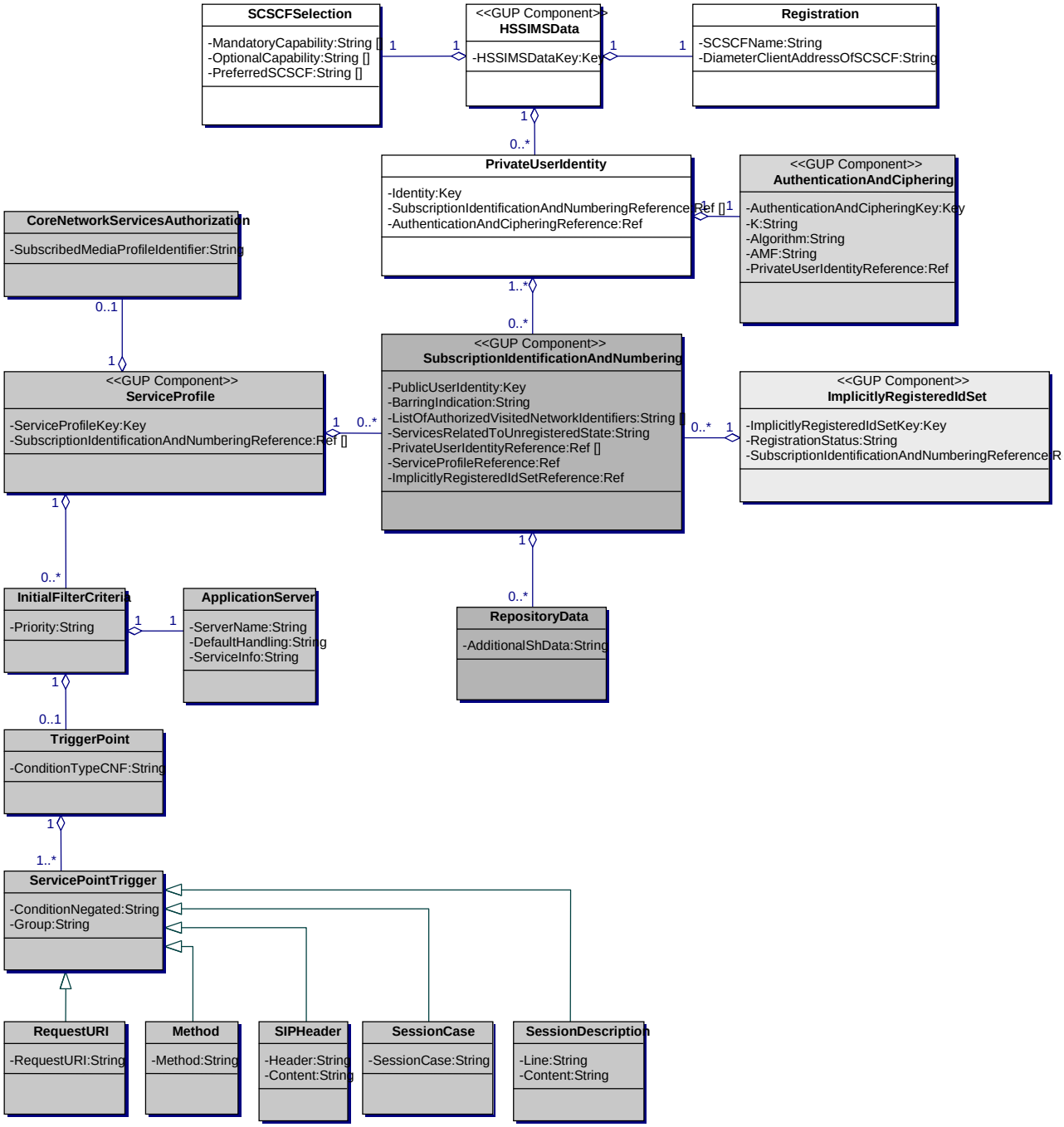

Figure A.3 Logical data model of the HSS IMS GUP Components

A.1.1.1 XML Schema files for HSS IMS GUP Components

XML Schema files attached to this specification combine together whole HSS IMS GUP Component data. XML Schema files are intended to be used by an XML parser.

Table A.1 lists HSS IMS GUP Components and XML Schema files that include those. CommonDataTypes file contains common data types, i.e. it is not a stand alone GUP Component on its own.

Table A.1 HSS IMS GUP Components in XML Schema files

HSS IMS GUP Component	XML Schema file
AuthenticationAndCiphering	AuthenticationAndCiphering.xsd

HSSIMSData	HSSIMSData.xsd
ImplicitlyRegisteredIdSet	ImplicitlyRegisteredIdSet.xsd
SubscriptionIdentificationAndNumbering	SubscriptionIdentificationAndNumbering.xsd
ServiceProfile	ServiceProfile.xsd
Component independent data	**XML Schema file**
-	CommonDataTypes.xsd

A.1.2 Element addressing

Contents for HSS IMS GUP Components can be addressed with XPath representation format as an input parameter for select clauses. Each GUP Component has a unique key.

Example
Following Select element addresses the BarringIndication element in the SubscriptionIdentificationAndNumbering GUP Component. Public User Identity is specified by ResourceID in the procedure message element, e.g. Query.

```
<Select>
    <ComponentType>
            http://3gpp/gup/ns/comp/SubscriptionIdentificationAndNumbering
        </ComponentType>
        <GCLRef>
            //SubscriptionIdentificationAndNumbering/BarringIndication
        </GCLRef>
</Select>
```

A.1.3 Void

A.2 HSS IMS GUP Component structure

HSS IMS GUP Component model includes following Components:

- HSSIMSData
- AuthenticationAndCiphering
- SubscriptionIdentificationAndNumbering
- ServiceProfile

This model is mainly based on the HSS data described in 3GPP TS 23.008 [8]. Authentication information is described in TS 33.203 [12] and TS 33.102 [11]. Data relationships are mainly based on descriptions from 3GPP TS 23.228 [9], 3GPP TS 29.328 [10] and 3GPP TS 29.228 [7].

A.2.1 HSSIMSData GUP Component

HSS IMS Data contains S-CSCF Selection-, IMS location, charging- and Private User Identity information.

HSS IMS Data contains also following data for Component linking purposes.

- HSSIMSDataKey
  - HSS IMS Data Key is a primary key to uniquely identify HSS IMS Data Component.
  - This value must be unique within all HSS IMS Data Components.

A.2.1.1 SCSCFSelection

S-CSCF Selection contains the following data, which are described in 3GPP TS 23.008 [8].

- MandatoryCapability
- OptionalCapability
- PreferredSCSCF (optional parameter)

These data shall be offered only for provisioning purposes.

A.2.1.2 IMS Location

IMS Location contains the following data, which are described in 3GPP TS 23.008 [8].

- SCSCFName
- DiameterClientAddressOfSCSCF (optional parameter)

A.2.1.3 PrivateUserIdentity

Private User Identity contains the following data, which is described in 3GPP TS 23.008 [8].

- Identity (Private User Identity)
  - This data shall be offered only for provisioning purposes.

A.2.2 AuthenticationAndCiphering GUP Component

Authentication and Ciphering contains the following data, which are described in 3GPP TS 33.102 [11].

- K
- Algorithm
- AMF

K parameter can contain either Secret Key or Encrypted Secret Key value. The encryption algorithm is implementation specific.

These data shall be offered only for provisioning purposes.

A.2.3 SubscriptionIdentificationAndNumbering GUP Component

Subscription Identification And Numbering Component contains the following data, which are described in 3GPP TS 23.008 [8].

- PublicUserIdentity
- BarringIndication (This data shall be offered only for provisioning purposes)
- ListOfAuthorizedVisitedNetworkIdentifiers (This data shall be offered only for provisioning purposes)
- ServicesRelatedToUnregisteredState

Public User Identity is a primary key element for the Subscription Identification And Numbering Component.

A.2.3.1 RepositoryData

Repository Data contains additional Sh-interface data, which are described in 3GPP TS 29.328 [10].

- AdditionalShData

A.2.4 ServiceProfile GUP Component

The present sub clause presents the Service Profile GUP Component contents, which are defined based on the 3GPP TS 29.228 [7].

A.2.4.1 CoreNetworkServiceAuthorization

Core Network Service Authorization contains the following data, which are described in 3GPP TS 29.228 [7].

- SubscribedMediaProfileIdentifier

This data shall be offered only for provisioning purposes

A.2.4.2 InitialFilterCriteria

Initial Filter Criteria contains the following data, which are described in 3GPP TS 29.228 [7]. All data under Initial Filter Criteria are described in 3GPP TS 29.228 [7].

- Priority

A.2.4.2.1 ApplicationServer

The present sub clause presents the Application Server element contents, which are defined based on the 3GPP TS 29.228 [7].

- ServerName
- DefaultHandling
- ServiceInfo

A.2.4.2.2 TriggerPoint

The present sub clause presents the Trigger Point element contents, which are defined based on the 3GPP TS 29.228 [7].

- ConditionTypeCNF

A.2.4.2.2.1 ServicePointTrigger

The present sub clause presents the Service Point Trigger element contents, which are defined based on the 3GPP TS 29.228 [7].

Corresponding data element in Cx reference point is SePoTri.

- ConditionNegated
- Group

Service Point Trigger contains one of the following data structures described below.

A.2.4.2.2.1.1 RequestURI

The present sub clause presents the Request URI element contents, which are defined based on the 3GPP TS 29.228 [7].

- RequestURI

A.2.4.2.2.1.2 Method

The present sub clause presents the Method element contents, which are defined based on the 3GPP TS 29.228 [7].

- Method

A.2.4.2.2.1.3 SIPHeader

The present sub clause presents the SIP Header element contents, which are defined based on the 3GPP TS 29.228 [7].

- Header
- Content

A.2.4.2.2.1.4 SessionCase

The present sub clause presents the Session Case element contents, which are defined based on the 3GPP TS 29.228 [7].

- SessionCase

A.2.4.2.2.1.5 SessionDescription

The present sub clause presents the Session Description element contents, which are defined based on the 3GPP TS 29.228 [7].

- Line
- Content

A.3 Common Data Types

GUP Components share some common data types defined in the CommonDataTypes.xsd - XML Schema file.

That XML Schema file includes the following elements or element types used by other GUP XML Schema files:

- Extension
  - Will be used as an extension element
- ExtensionType
  - Extension element type
- GenericDataType
  - Will be used as a generic data type
- GenericComponentReferenceType
  - Will be used as a generic Component reference type

A.4 Data Services Template Checklist tables

GUP is an instance of a data service as described by Liberty Alliance Data Services Template [13] specification and all its stipulations are hereby incorporated unless expressly waived or modified in this specification.

This section presents the checklist tables as defined in Liberty Alliance Data Service Template [13] specification filled-in with the values corresponding of the use of defined GUP Data component.

Table A.4.1/1: General Service Parameters (1/2)

Parameter	Value
<ServiceType>	urn:3gpp:gup-hss-ims:2005-04
Discovery Options	Discovery Options MAY be supported If a Discovery Service is employed, the following Discovery Option Keywords as defined by Liberty ID-WSF Data Services Template Specification [13] MAY be used: urn:liberty:dst:allPaths urn:liberty:dst:can:extend urn:liberty:dst:changeHistoryS upported urn:liberty:dst:extend urn:liberty:dst:fullXPath urn:liberty:dst:multipleResources urn:liberty:dst:multipleQueryItems urn:liberty:dst:multipleModification urn:liberty:dst:noModify urn:liberty:dst:noPagination urn:liberty:dst:noQuery urn:liberty:dst:noQuerySubscriptions urn:liberty:dst:noSorting urn:liberty:dst:noStatic urn:liberty:dst:noSubscribe No specific Discovery Option Keyword is defined for the HSS-IMS GUP Data Componet.
Data Schema	See Apendix C in this specification
SelectType Definition	The SelectType definition for the HSS-IMS GUP data component is described in the subclause 9.3.2.
Semantics of the <Select> element	See Chapter 6.5 in this specification

Table A.4.1/2: General Service Parameters (2/2)

Parameter	Value
Element uniqueness	Not applicable. No multiple elements with same name used.
Data Extension Supported	The <Extension> element MAY be used to add additional HSS-IMS GUP elements, but the use is not specified by this specification.

Table A.4.2/1: Query Parameters (1/2)

Parameter	Value
Support querying	Queries MUST be supported
Multiple <Query> elements	Multiple <Query> elements MAY be supported.
Multiple <QueryItem> elements	Multiple <QueryItem> elements MAY be supported.
Support sorting	Not Applicable
SortType definition	Not Applicable as sorting is not supported, empty definition used: <xs:complexType name="SortType"> <xs:complexContent> <xs:restriction base="EmptyType"/> </xs:complexContent> </xs:complexType> changedSince MAY be supported.
Support changedSince	All Queries MUST be supported
Supported formats	Multiple <Query> elements MAY be supported.

Table A.4.2/2: Query Parameters (2/2)

Parameter	Value
Support includeCommon Attributes	Common Attributes MUST be supported. See Chapter 6.4 in this specification.
Support pagination	Not Applicable. MUST NOT be used.
Support static sets	Not Applicable. MUST NOT be used.
<Extension> in <Query>	The <Extension> element in <Query> MAY be used for new parameters, but the use is not specified by this specification.

Table A.4.3: Modify Parameters

Parameter	Value
Support modification	Modifications MUST be supported. Multiple <Modify> elements MAY be supported.
Multiple <Modify> elements	Multiple <Modification> elements MAY be supported.
Multiple <Modification> elements	Partial Success MAY be supported.
Support partial success	notChangedSince MAY be supported.
Support notChangedSince	The <Extension> element in <Modify> MAY be used to specify new parameters, but the use is not specified by this specification.
<Extension> in <Modify>	Modifications MUST be supported. Multiple <Modify> elements MAY be supported.

Table A.4.4/1: Subscribe Parameters (1/2)

Parameter	Value
Support subscribing to notifications	Subscriptions to Notifications MUST be supported.
Use of the <Subscribe> element for modifying and renewing subscriptions.	Creation, Renewal, Cancellation and Modification of subscriptions MUST be supported.
Multiple <Subscribe> elements	Multiple <Subscribe> elements MAY be supported.
Multiple <Subscription> elements	Multiple <Subscription> elements MAY be supported.
Use of the <NotifyEndedTo> element	The <NotifyEndedTo> element MUST be supported, if end notifications are used.

TypeType definition	The <Type> element is not used, so an empty definition is used for it:
	`<xs:complexType name="TypeType">`
	`<xs:complexContent>`
	`<xs:restriction base="EmptyType"/>`
	`</xs:complexContent>`
	`</xs:complexType>`

Table A.4.4/2: Subscribe Parameters (2/2)

Parameter	Value

TriggerType definition	The <Trigger> element is not used for the HSS-IMS GUP Data Component (just default change notifications are supported).
	An empty type definition shall be used:
	`<xs:complexType name="TriggerType">`
	`<xs:complexContent>`
	`<xs:restriction base="EmptyType"/>`
	`</xs:complexContent>`
	`</xs:complexType>`

Start of a subscription	starts attribute MUST be supported.
Subscription expiration	Subscription expiration MUST be used.
Use of expires and duration attributes	Both expires and duration MUST be supported.
Support expires==starts Support zero duration	The same value for both the starts and the expires attribute MUST be supported. The value zero MUST be supported for the duration attribute.
<Extension> in <Subscribe>	The <Extension> element in <Subscribe> MAY be used for new parameters, but the use is not specified by this specification.

Table A.4.5: QuerySubscriptions Parameters

Parameter	Value
Support querying existing subscriptions	Query Subscriptions MAY be supported.
Multiple <QuerySubscriptions> elements	Multiple <QuerySubscriptions> elements MAY be supported.
<Extension> in <QuerySubscrip	<Extension> element in <QuerySubscriptions> MAY be used for new parameters, but the use is not specified by this specification.

Table A.4.6: Notify Parameters

Parameter	Value
Support notifications	Notifications MUST be supported.
Are notifications acknowledged	End notifications MUST be acknowledged.
`<Extension>` in `<Notify>`	The `<Extension>` element in `<Notify>` MAY be used to pass additional data, but the use is not specified by this specification.

Table A.4.7: EndNotify Parameters

Parameter	Value
Support end notifications	End notifications MAY be supported.
Are end notifications acknowledged	End notifications SHOULD be acknowledged.
`<Extension>` in `<Ended>`	The `<Extension>` element in `<Ended>` MAY be used to pass additional data, but the use is not specified by this specification.

Annex B (normative): WSDL Definitions

Ediror's note: This shall include operations

Annex C (normative): XML Schema Definitions

Ediror's note: This shall include content for operations

Annex D (informative): XML Schema Structure

There are several different XML Schemas needed for different purposes in GUP. One set of schemas is needed to describe the procedure level data (procedure names and their parameters) while another set of schemas are used to describe the profile data carried with different procedures.

The following figure describes different kind of XML Schemas and their relations.

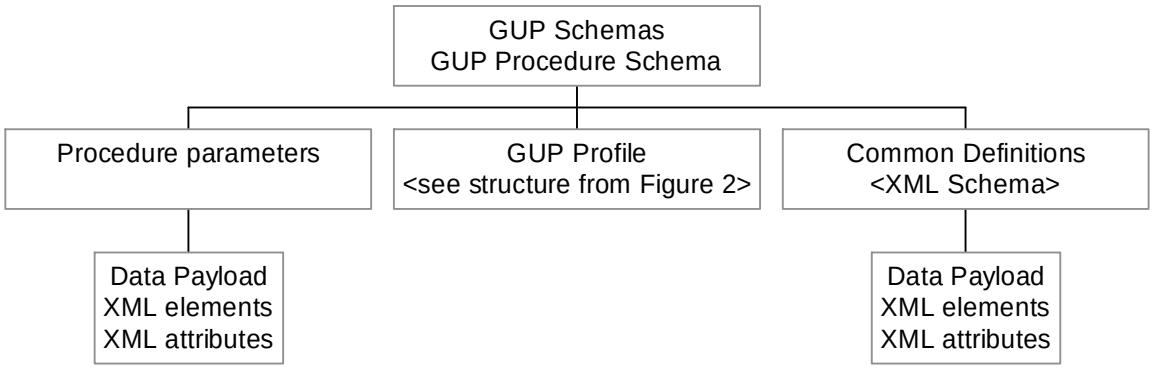

Figure D.1. XML Schemas and their relations

The purpose of the GUP Procedure Schema is to describe all the GUP procedures and their parameters.

The XML Schemas of the GUP procedures include pre-defined places for conveying the actual profile data which have XML Schemas of their own. E.g., the XML Schema definition of the Query procedure (the response element) includes the 'Data' element for placing data of GUP Components.

The following data specific schemas are defined:

- Profile Component Schema Template
- Component type specific Profile Component Schemas
- GUP Common Definitions Schemas.

The purpose of the Profile Component Schema Template is to give an example how guidelines of this specification are implemented in practise in defining a new XML Schema. The template describes different items of definition. It is recommended to check the definitions from the template when defineing Profile Component Schemas in order to have a uniform structure for all Profile Components.

For each type of GUP Component there is one Profile Component Schema, which may contain or refer to other XML Schemas. GUP specifies certain Profile Component Schemas, e.g., HSS specific Profile Component Schema(s), and guidelines on how to specify additional GUP compatible Profile Component schemas. The Profile Component Schema Template gives an example of a generic Profile Component Schema and shows how the guidelines are implemented in practise.

The GUP Common Definitions Schemas describe sets of XML items and XML data type definitions commonly used by other GUP specific schemas. Also the needed metadata items are defined. The GUP Common Definitions Schemas have no usage of their own, but they are utilized by most of the GUP specific schemas.

There is no separate XML Schema describing the whole User Profile needed for the Rg reference point. However, the present document shows how the profile XML document containing data from several GUP Components is composed from a number of separate Profile Component Schemas (see Figure 2). Even when there is no XML Schema, the Profile Component Schemas known by both the sending and receiving entity can be used to validate and interpret the data of individual GUP Components. Note that the GUP Components may reference other GUP Components that may be e.g. for common use.

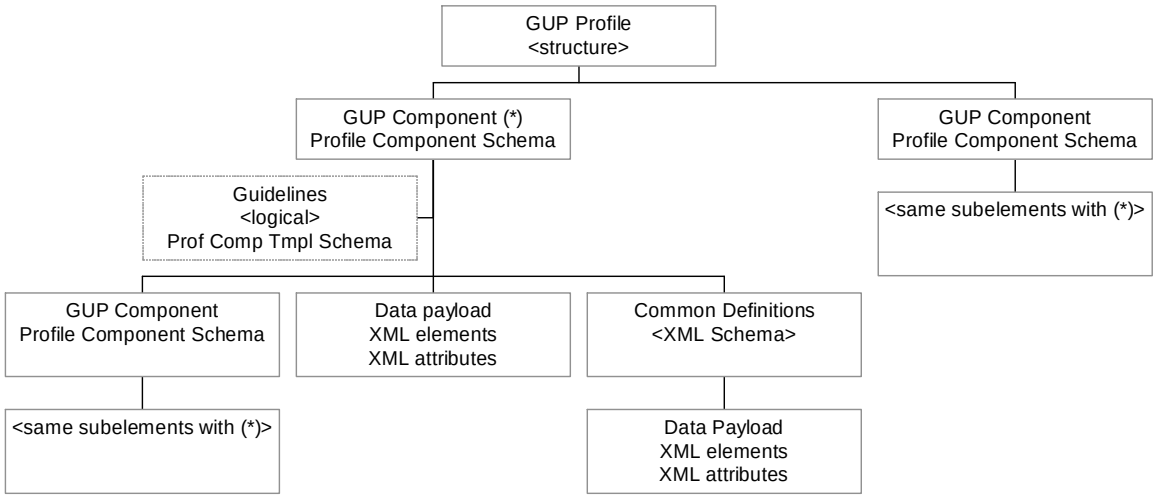

Figure D.2. XML Schemas composing a GUP Profile

More information about the XML Schemas and their structures can be found in clause 6 of the present document.

Annex E (normative): SOAP binding for GUP headers

The SOAP protocol provides a mechanism for exchanging structured and typed information between peers using XML. It is a very generic protocol, which can also be used to carry remote procedure calls. Each SOAP message has an element "Envelope" and its immediate child elements "Header" and "Body".

SOAP is applied in the Rp and Rg reference points. SOAP carries the GUP procedure elements in its body part in compliance with the SOAP standard [5]. The GUP Procedure elements are placed immediately below the Body element. If there are several requests or responses, the GUP Procedure elements are carried one after another.

SOAP headers used in the context of GUP shall be as defined by Liberty Alliance ID-WSF SOAP Bindings [14] specification. This Liberty Alliance specification includes the definition of a number of SOAP header blocks for simple message correlation, as well as provider declaration, processing context, consent claims, usage directives and a number of other optional headers. Liberty Alliance ID-WSF SOAP Bindings [14] specification also defines how messages are bound into SOAP message bodies, and how the Liberty defined SOAP header blocks are bound into SOAP message headers.

Additionally, other SOAP header blocks defined by other Liberty Alliance specifications (i.e. Liberty Alliance ID-WSF Security Mechanisms [15] and Liberty Alliance ID-WSF Interaction Service [23]) shall be also applicable in the context of GUP and may be used concurrently with the header blocks defined in Liberty Alliance ID-WSF SOAP Bindings [14] specification.This Annex presents a brief description of the different SOAP header blocks used in the context of GUP. Other sections in this specification also refer to the use of such headers in the context of GUP.

E.1 Correlation header

SOAP does not define a mechanism to correlate one SOAP message with another message, such as in a request-

response paradigm. The **correlation** header block provides a means for being able to correlate one SOAP message with another SOAP message. Message correlation is achieved by inserting a `<Correlation>` element in SOAP-bound GUP message headers and using a `messageID` attribute to identify individual messages. Additionally, a message may refer to another message by setting its `refToMessageID` attribute to the value of the `messageID` of the message of interest.

Normative semantics, definitions, schemas and processing rules for the correlation header block are defined in Liberty Alliance ID-WSF SOAP Bindings [14] specification.

E.2 Provider header

A Provider in the context of GUP is an entity that is able to handle (issue and/or receive) SOAP messages, and which is univocally identified at the SOAP level by its providerID. This header block provides means for a sender to claim that it is represented by a given providerID value.

Normative semantics, definitions, schemas and processing rules for the provider header block are defined in Liberty Alliance ID-WSF SOAP Bindings [14] specification.

E.3 ProcessingContext header

A GUP requestor may need to express additional context for a given request, for example indicating that the

requester expects to make such requests in the future when the end-user may or may not be online. This header block may be employed by a sender to signal to a receiver that the latter should add a specific additional facet to the overall *processing context* in which any action(s) are invoked as a result of processing any message also conveyed in the overall message.

Normative semantics, definitions, schemas and processing rules for the processing context header block are defined in Liberty Alliance ID-WSF SOAP Bindings [14] specification.

E.4 Consent header

GUP applications may wish to claim whether they obtained the end-user's consent for carrying out any given operation. This header block is used to explicitly claim that the Principal consented to the present interaction.

Normative semantics, definitions, schemas and processing rules for the consent header block are defined in Liberty Alliance ID-WSF SOAP Bindings [14] specification.

E.5 UsageDirective header

GUP requestors may wish to indicate their policies for handling data at the time of data request, while GUP server and RAFs may wish to specify their policies for the subsequent use of data at the time of data release. To facilitate this, GUP requestors may add one or more UsageDirective header blocks to the message being sent. A UsageDirective header appearing in a request message expresses *intended usage*. A UsageDirective header appearing in a response expresses *how* the receiver of the response is to use the response data.

Normative semantics, definitions, schemas and processing rules for the usage directive header block are defined in Liberty Alliance ID-WSF SOAP Bindings [14] specification.

E.6 ServiceInstanceUpdate header

It may be necessary for an entity receiving a message to indicate that messages from the sender should be directed to a different endpoint, or that they wish a different credential to be used than was originally specified by the entity for access to the requested resource.

The ServiceInstanceUpdate header allows a message receiver to indicate that a new endpoint, new credentials, or new security mechanisms should be employed by the sender of the message.

The use of this header block allows the sender of the message to convey updates to security tokens, essentially providing a token renewal mechanism. This is not discussed further in this specification.

Normative semantics, definitions, schemas and processing rules for the service instance update header block are defined in Liberty Alliance ID-WSF SOAP Bindings [14] specification.

E.7 Timeout header

A requesting entity may wish to indicate that they would like a request to be processed within some specified amount of time. Such an entity would indicate their wish via the Timeout header block.

Normative semantics, definitions, schemas and processing rules for the timeout header block are defined in Liberty Alliance ID-WSF SOAP Bindings [14] specification.

E.8 CredentialsContext header

The receiver of a GUP message might indicate that credentials supplied in the request did not meet its policy in

allowing access to the requested resource. The `<CredentialsContext>` header block allows such policies to be expressed to a GUP requester.

Normative semantics, definitions, schemas and processing rules for the credentials context header block are defined in Liberty Alliance ID-WSF SOAP Bindings [14] specification.

E.9 Security header

OASIS WS-Security compliant header elements as defined in OASIS SOAP Message Security [24] specification communicate the relevant security information in SOAP.

The construction and decoration of the `<wsse:Security>` header element in the context of GUP MUST adhere to the rules specified in OASIS SOAP Message Security [24] specification.

Additionally the construction, decoration and usage of the `<wsse:Security>` header element in the context of GUP MUST also adhere to the mechanisms defined in Liberty Alliance Security Mechanisms [15] specification.

Normative semantics, definitions, schemas and processing rules for the security header block are defined both in OASIS SOAP Message Security [24] and Liberty Alliance ID-WSF Security Mechanisms [15] specifications.

E.10 UserInteraction header

In some scenarios it might be necessary for GUP Servers and/or RAFs to interact with the owner of GUP component data exposed in order to e.g. ask for explicit end-user consent for the release of the requested information. To this end, GUP messages may include a `<UserInteraction>` SOAP header, which controls the possibilities that GUP Servers and/or RAFs have to engage in interactions with end-users when serving a request from a GUP Requestor.

Normative semantics, definitions, schemas and processing rules for the user interaction header block are defined in Liberty Alliance ID-WSF Interaction Service [23] specification.

Annex G (informative): Change history

Date	TSG #	TSG Doc.	CR	Rev	Subject/Comment	New
2005-03	CN#27	NP-050043			TS 29.240 Rel-6 approved at CN#27	6.0.0
2005-06	CT#28	CP-050091 CP-050091 CP-050091 CP-050091	0003 0004 0005 0006	1	GUP HSS-IMS Component Definition Clarification of the content of Annex <A> GUP Security and Authentication GUP SOAP Headers	6.1.0
2007-06	CT#36				Upgraded unchanged from Rel-6	7.0.0
2008-12	CT#42				Upgraded unchanged from Rel-7	8.0.0
2009-12	-	-	-	-	Update to Rel-9 version (MCC)	9.0.0
2011-03	-	-	-	-	Update to Rel-10 version (MCC)	10.0.0
2012-09	-	-	-	-	Update to Rel-11 version (MCC)	11.0.0
2014-09	-	-	-	-	Update to Rel-12 version (MCC)	12.0.0
2015-12	CT#70	-	-	-	Update to Rel-13 version (MCC)	13.0.0
2017-03	CT#75	-	-	-	Update to Rel-14 version (MCC)	14.0.0
2018-10	CT#81	-	-	-	Update to Rel-15 version (MCC)	15.0.0
2020-07	CT#88e	-	-	-	Update to Rel-16 version (MCC)	16.0.0

History

Document history		
V16.0.0	August 2020	Publication