3GPP TSG SA WG3 Security — S3#22 **S3-020055**

25 - 28 February 2002

Bristol, UK

3GPP TSG T WG2 Meeting #16 ***T2-020254***
Sophia Antipolis, France, 11th to 15th Feb. 2002

Title: Liaison Statement on coordination of data definitions, identified in GUP development
Source: T2
To: S3, S4, S5, N1, N4, N5, T3
Cc: S1, S2
Response to:

Contact Person:
Name: Peter Neumann
E-mail Address: Peter.Neumann@mch.Siemens.de

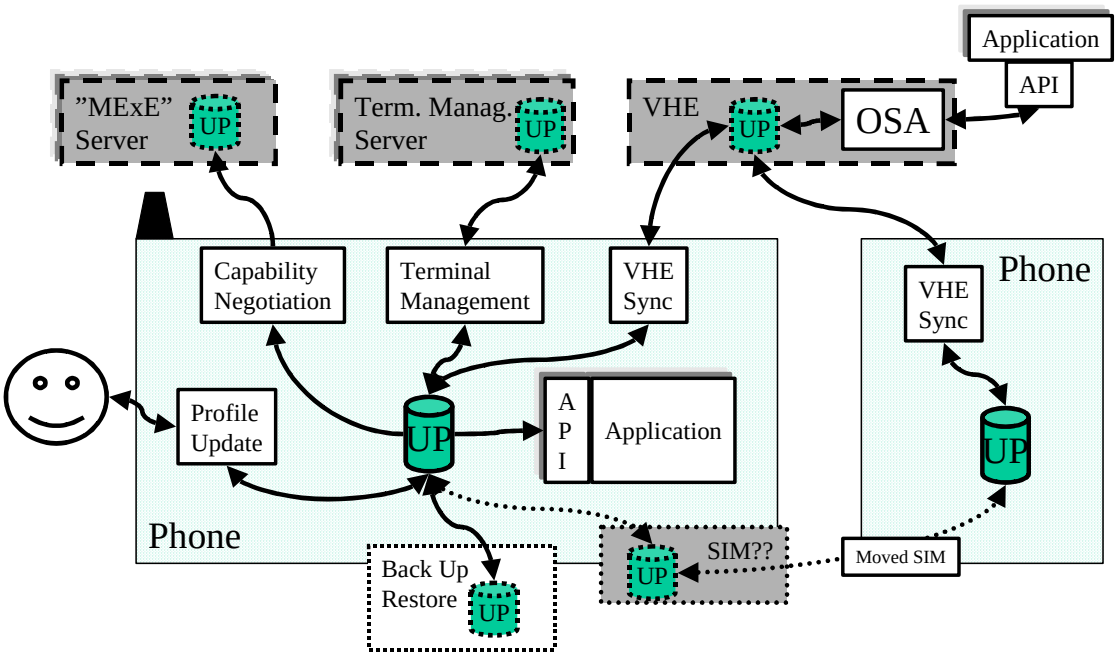
Attachments: T2-020108 (TS 23.241)
T2-020109 (TS 24.241)

---

1. General:

During the recently held T2 GUP ad hoc meeting, a need to clarify to the addressed parties with the intention and current status of the T2 Stage 2 (TS 23.241) and Stage 3 (TS 24.241) GUP work, and it's relationship to other 3GPP working groups was identified.

Problem
In the work with GUP it is realised that many data elements are used by many groups. Some data are shared between these groups. If there is no coordination between these groups, there is a high probability that the same data will be described and defined many times.



The figure above illustrates an example of a distributed view of the 3GPP Generic User Profile.

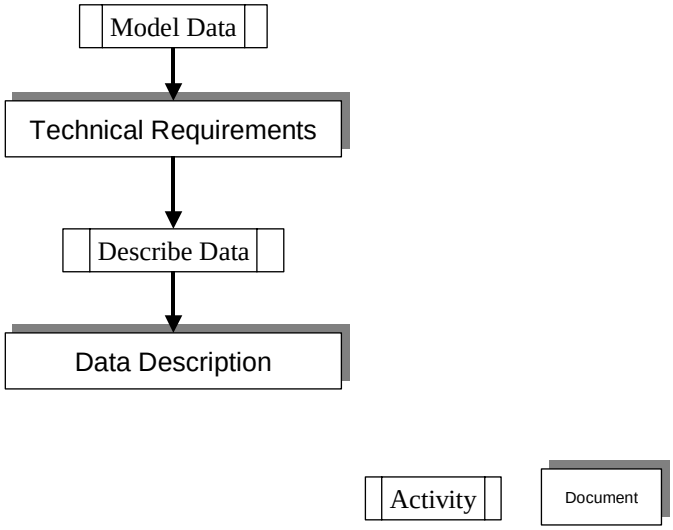
It has been identified for example that there is a lot of commonality between GPRS, WAP and MMS, and similarly SA5 and the GUP appear to have a lot of commonality.

Background

The T2 GUP Ad Hoc is developing a model for developing data descriptions, illustrated below.

The identification and description of data can be divided in two steps:

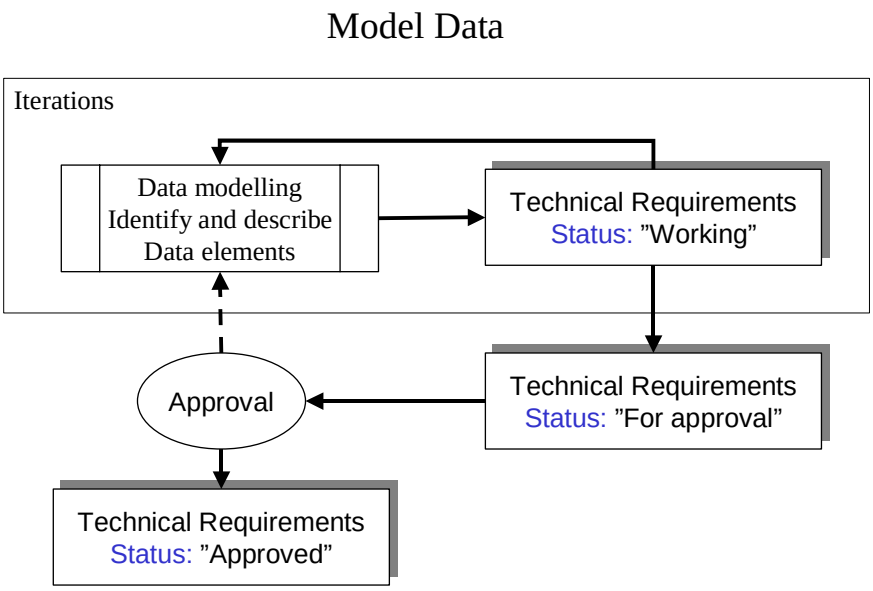
- Data modelling
  To analyse the requirements and to create an informal definition of the data
- Data description
  To make a formal description of the data identified in the data modelling



In both steps there is a need for coordination.

The Data Description Framework (described in T2 Stage 2 (TS 23.241)) is applicable to the data description step. The DDF and resultant DD is not simply restricted to the GUP, but could be relevant to many other areas of the 3GPP system, where data needs to be described in a formalised manner.

The work procedures to create and approve a Data Description is planned to be defined in the T2 (TS 24.241) 3.
The picture below is a first outline of a part of such a work procedure.



The output from the data modelling and the resulting Data Description are urgently needed by many groups, as indicated in the problem description. To shorten the time to delivery the work has to be divided into parallel and coordinated activities.

The GUP ad hoc has identified that such activities are:

- Define the data modelling and data description work procedures
- Define the Data Description Framework
- Do data modelling
- Identify an initial set of common datatypes and dataobjects
- Do Data Descriptions

The addressed groups are asked to note this fact, and to consider using these tools in their own areas of interest. As a result the affected groups may wish to develop models and Data Descriptions for any new services to be introduced, which will then result in an associated DD. This step unifies different datatypes and descriptions to a single library of data descriptions. T2 is potentially responsible for filtering incoming Data Descriptions and generating common Data Descriptions being used across the 3GPP system.

The addressed committees are naturally in charge of providing data modelling and Data Description for services they are responsible for, in a timely manner, to T2. Data modelling and Data Description may be two distinct tasks, accordingly, there is a need to coordinate and agree upon working, publishing and approval procedures.

Our current understanding is that the above addressed committees are directly affected: if any of these committees identifies the need to inform other committees of this work, please forward this Liaison Statement accordingly.

Proposal

To have a single group responsible for the ***coordination*** of the data definitions, whilst noting that the actual data definition work is the responsibility of the respective working groups, where the relevant expertise resides.

2. Actions:

- The addressed groups are invited to indicate a general agreement to the concept of unifying the Data Descriptions (e.g. using GUP as a starting point), and the generation of a common DDF for 3GPP.

- The addressed groups are invited to consider the above proposal.

3. Date of next T2 meetings:

GUP Joint Ad Hoc	2nd - 5th April 2002
T2#17	13th- 17th May, 2002
T2#18	19th - 23rd August, 2002

T2#16 **T2-020108**
Sophia Antipolis, France
11-15 February 2002

3GPP TS 23.241 V0.~~2.1~~3.0 (200~~1-12~~2-02)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Terminals;
3GPP Generic User Profile - Data Description Framework;
Stage 2
(Release 6~~5~~)

Remove GSM logo from the cover page for pure 3rd Generation documents.

Select keywords from list provided in specs database.

Keywords

<keyword[, keyword]>

3GPP

Postal address

3GPP support office address

650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet

http://www.3gpp.org

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.

Introduction

This clause is optional. If it exists, it is always the second unnumbered clause.

1 Scope

This clause shall start on a new page.

The present document …

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] W3C Recommendation: "Extensible Markup Language (XML) 1.0 (Second Edition)". http://www.w3.org/TR/2000/REC-xml-20001006

[2] W3C Recommendation: "Namespaces in XML", 2 May 2001. http://www.w3.org/TR/1999/REC-xml-names-19990114/

[3] W3C Recommendation: "XML Schema Part 0: Primer", 2 May 2001. http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/

[4] W3C Recommendation: "XML Schema Part 1: Structures", 2 May 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[5] W3C Recommendation: "XML Schema Part 2: Datatypes", 2 May 2001. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[6] W3C Recommendation: "XML Path Language (XPath) Version 1.0", 16 November 1999. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[7] W3C Candidate Recommendation: "XML Pointer Language (XPointer) Version 1.0", 11 September 2001. http://www.w3.org/TR/1999/REC-xpath-19991116

[8] ISO (International Organization for Standardization): "ISO 11404, Language-independent Datatypes.

3 Definitions, symbols and abbreviations

Delete from the above heading those words which are not applicable.

Subclause numbering depends on applicability and should be renumbered accordingly.

3.1 Definitions

For the purposes of the present document, the [following] terms and definitions [given in ... and the following] apply.

Definition format

***<defined term>**: <definition>.*

example: text used to clarify abstract rules by applying them literally.

3.2 Symbols

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

Symbol format

<symbol> <Explanation>

3.3 Abbreviations

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

Abbreviation format

<ACRONYM> <Explanation>

4 Data Description Framework

Editor's note: Align with S2 stage 2 on GUP and DDF, 23.xyz.

The data description "matter" can be split in the following domains:

- **Data**
  Data stored and or access in a User Profile
- **Data Description**
  describes the data contained in the User Profile. (This also called the Schema level.)
- **Data Description Framework**
  Defines how to create the data description. (This also called the Schema-Schema level i.e. the Schema describing the Schema, which describes the data.)

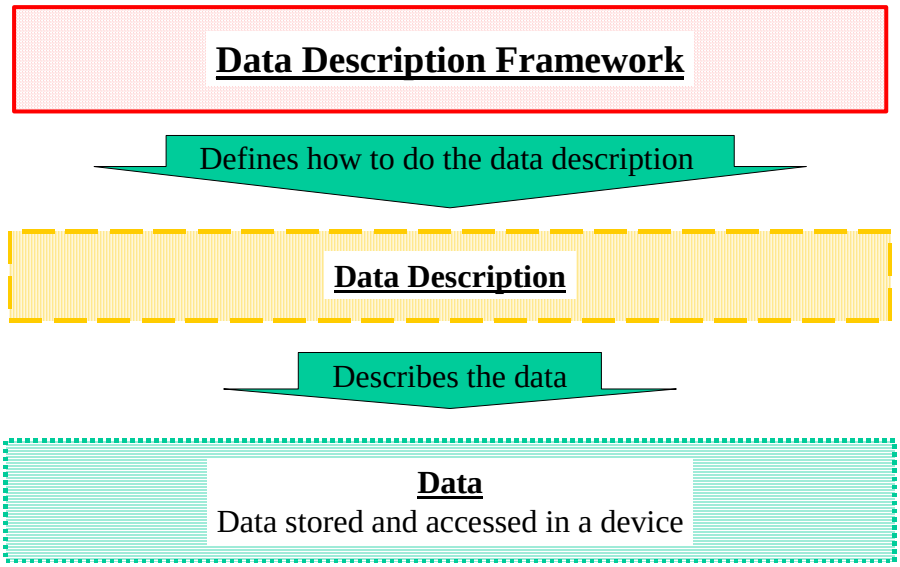


Figure 1: xxx

The Data Description Framework defines the method to describe the data in a Generic User Profile. It defines the structure of the Data Description as based on XML-schema. The Data Description Framework also defines a default representation (or transport format) of Data Descriptions and the data in a Generic User Profiles.

A specific Generic User Profile will be described, according to the Data Description Framework, resulting in a Data Description.

The structure and semantic of the data in a Generic User Profile is described in the Data Description. The Data Description Framework also defines a default representation of the data in the User Profile.

4.1 Basic Structure of the Generic User Profile

The Data Description Framework is used to describe Generic User Profiles with the basic structure shown in the picture below.

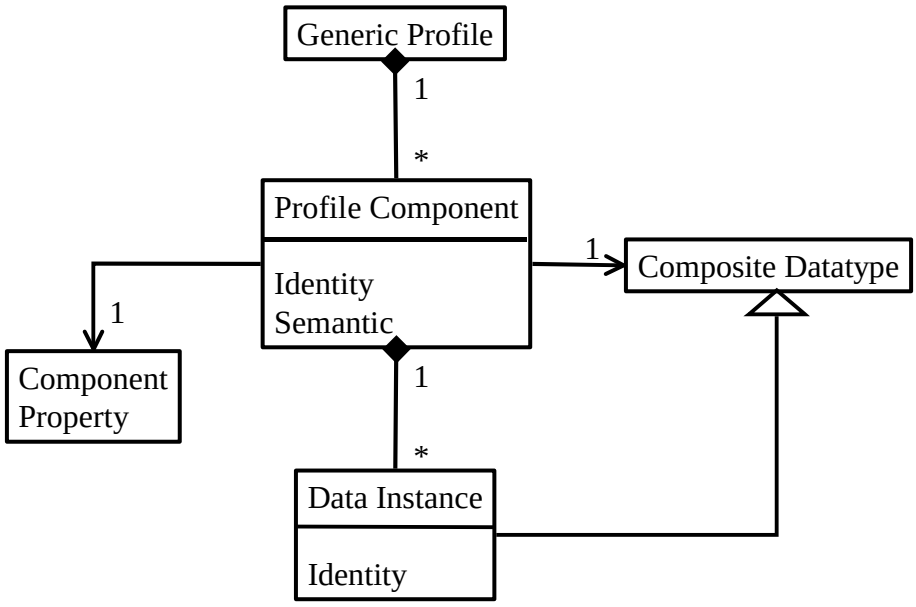


Figure 2: xxx

4.2 Types of Data Description Documents

A Data Description consists of a number of XML-documents.

The picture “Relationship between Data Description Documents” indicates the types of documents and the relation between those documents.

Note: An arrow means a reference between documents.

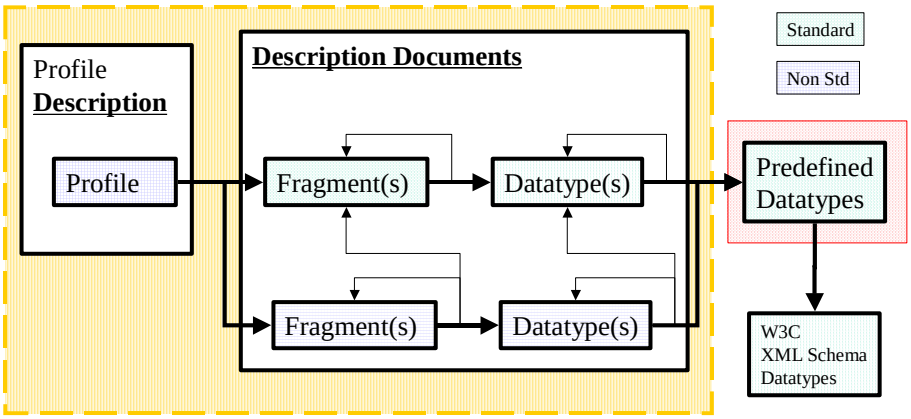


Figure 3: Relationship between Data Description Documents

There are three types of Data Description documents:

- Profile
  A Profile document is the main document in the description of the data in a Generic User Profile. It mainly contains references to Generic User Profile parts defined in Fragment documents. It is specific for a class of Generic User Profiles.

- Fragment
  Generic User Profiles parts containing Profile Components are defined in Fragment documents. Profile Components are declared mainly by connecting a Profile Component identity to a Datatype. Fragment documents can be shared between Data Descriptions.

- Datatype
  In this kind of document Datatypes are defined. Datatypes are defined using in the Data Description Framework predefined Datatypes and **user defined** Datatypes. Datatype documents can be shared between Data Descriptions.

4.3 Data Description Framework Parts

The Data Description Framework consists of:

- XML-schema files (in the appendix of this document)
- Description rules
- Default XML-based transport format

5 Logical Structure of Data Descriptions

Main elements used in the Data Descriptions are:

Profile Declares a Generic User Profile.

Fragment Defines a reusable Generic User Profile part.

Component Declares a Profile Component.

Property Declares a set of properties, which can be associated with some Profile Components.

Semantic Defines the meaning of things. Is used to understand the content of the Generic User Profile.

Comment Provides information intended for (the remainder of)? the Data Description.

Datatype Defines a Datatype.

Each of the elements is briefly described in the following section and in more detail in a separate chapter (NS: what chapter?).

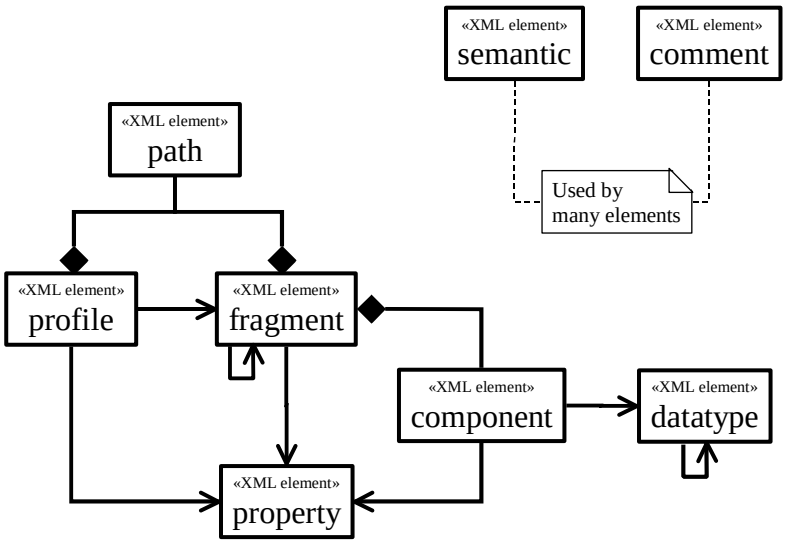


Figure 4: xxx

5.1 Profile

The Profile element is used to declare a Generic User Profile. It mainly contains the references to a number of Profile Fragments. The results of merging the referenced Profile Fragments are:

- A hierarchical name structure containing a number of Profile Components.
- A description of each Profile Component
    - Abstract syntax
    - Semantic

5.2 Fragment

In a Fragment element a number of Profile Components are declared. A Profile Fragment is normally used in many Generic User Profiles.

A Profile Fragment can include other Profile Fragments by referencing them.

5.3 Component

The Component element is used to declare the Profile Component.

The following are defined for a Profile Component :

- Identity
  a hierarchical namespace, similar to that in a file system, is used. The identities are selected in a way that they can be used as Universal Resource Locators, URLs.
- Semantic
  gives the meaning of the Profile Component.
- For the Data Instance
  - Possible number of Data Instances
  - Datatype
    The abstract syntax of the Data Instance is defined by referencing a Composite Datatype.
- Component Properties

5.4 Property

The property element declares a Component Property, which is used to control the usage and handling of Profile Components.

A Profile Component is directly or indirectly referencing one Component Property.

5.5 Semantic

The Semantic element is used in many elements to define the meaning. Examples are defining the meaning of a Profile Component, a Datatype, or a specific value in enumerations. The semantic is given using normal language. It is possible to give it in several different languages.

The information in the semantic elements is used by the interpretation and usage of the values in a Generic User Profile

5.6 Comment

Comment elements are used to add(?) comments to the Data Description.

5.7 Datatype

A group of elements is used to describe Datatypes. In the Data Description Framework there are a number of predefined Datatypes. Simple user defined Datatypes can be defined based on the predefined simple Datatypes. Composite Datatypes are defined using simple and other composite Datatypes.

The abstract syntax of a Component Data Instance is defined by referencing a Composite Datatype.

6 Notation used in this Specification

6.1 Rules

In this document an informal "Extended Backus-Naur Form (EBNF)" like notation is used.

EBNF rules are used in the document. The syntax of a rule is:

```
{symbol} ::= expression.
```

The rule is describing the {symbol} using an expression (or some text).

The special symbols used in the expression are:

{symbol**}**	{symbol} is explained in a rule with {symbol} as its right side (**{**symbol**}::=** …). It can be regarded as a placeholder for the thing described in the rule.
(expression**)**	Expression is treated as a unit when combined as described in the following 5 rows.
A**?**	Zero or one occurrences of A; optional A
A B	A followed by B (Concatenation). This operator has higher precedence than alternation; thus A B | C D is identical to (A B) | (C D).
A **|** B	A or B but not both (Alternation).
A**+**	One or more occurrences of A. Concatenation has higher precedence than alternation; thus A+ | B+ is identical to (A+) | (B+).
A*****	Zero or more occurrences of A. Concatenation has higher precedence than alternation; thus A* | B* is identical to (A*) | (B*).
{* comment ***}**	A comment in the expression.
{xsi:datatype**}**	Data of a simple type defined in "XML Schema Part 2: Datatypes" [5].
elementName	Used in XML-element content models.

6.2 XML-element

The following layout is used in the description of XML-elements:

Synopsis:

```
<tag
  attributeName
  attributeType = {xsi:datatype}
  optionalAttribute?
  enumAttribute = (large | medium | small) : medium
>
Content: expression
</tag>
```

In the start tag there is a list of attribute names (`attributeName`, `attributeType`, `optionalAttribute` and `enumAttribute`).

Optional attributes has a "?" after its name.

Attribute of a simple type defined in “XML Schema Part 2: Datatypes” [5] is indicated with `{xsi:datatype}` as for the `attributeType` above.

Where an attribute is of an enumerated datatype, the possible values are shown separated by vertical bars, as for the `enumAttribute` above; if there is a default value, it is shown following a colon.

The expression following “Content:” is an expression describing the allowed content of the element. Name not surrounded by {}, used in the expression is the name of an element, which may appear as a child element. The optional character following a name or sub-expression, governs whether the element or the sub-expression may occur one or more (+), zero or more (*), or zero or one times (?). The absence of such an operator means that the element or content particle must appear exactly once.

Example:

```
<example
  count = {xsi:integer}
  size? = (large | medium | small) : medium
>
Content: (all | any*)
</example>
```

6.2.1 Headlines used in XML-element descriptions

[editor’s note: need a 11.1.2.2]

The following headlines are used in the description of XML-elements:

Synopsis:

Indicating the syntax in the description of datatype.

Example:

Contains an example of a (part) of a datatype description.

Example data in XML-format:

Contains an example of a data in XML-format conforming to the datatype description.

Synopsis as expanded XML-schema:

The synopsis expressed using in XML-schema.

Example as expanded XML-schema:

The example expressed using in XML-schema.

7 Generic User Profile Description

7.1 Element profile

Synopsis:

```
<profile {propertyRef}?
>
  Content: {semantic} (fragmentRef | pathInProfile)*
</profile>
```

7.2 Element path used in profile element

Synopsis pathInProfile:

```
<path pathName = {pathName}>
  Content: {optSemantic} (fragmentRef | pathInProfile)*
</path>
```

8 Profile Fragment Description

8.1 Element fragment

Synopsis:

```
<fragment
      name = {fragmentName}
      pathName? = {pathName}>
  Content: {optSemantic} (component | fragmentRef | pathInFragment)*
</fragment>
```

The {fargmentName} is used when the fragment is referenced form a profile element or other fragment elements.

8.2 Element fragmentRef

Synopsis:

```
<fragmentRef
      fragmentRef = {fragmentRef}
      pathName? = {pathName}
      propertyRef? = {propertyRef}
>
  Content: {optSemantic}
</fragmentRef>
```

The fragment referenced by {fragmentRef} will in the resulting profile replace this element. It works like an include statement.

It is regarded an error if both the referring fragmentRef-element and the referenced fragment-elements have a pathName attribute.

8.3 Element path used in fragment element

Synopsis pathInFragment:

```
<path pathname =”{pathName}”
      propertyRef?="{propertyRef}">
  Content: {optSemantic} (component | fragmentRef | pathInFragment)*
</path>
```

9 Profile Component Description

9.1 Element component

Synopsis:

```
<component
      pathname? = {pathName}
      datatypeRef = {compositeDatatype}
      propertyRef? = {propertyRef}
      minInstances? = {minOccur}
      maxInstances? = {maxOccur}
>
  Content: {optSemantic}
</component>
```

Synopsis:

```
{minOccur} ::= {xsi:unsignedShort}
{maxOccur} ::= {xsi:unsignedShort} | “unbounded”
```

10 Component Property

The Component Property contains information controlling the usage and handling of a Profile Component. To allow several Profile Components to use the same Component Property, references are used. A Profile Component is directly or indirectly referencing one Component Property. Profile Components sharing Component Properties will be handled in the same way.

Example of property information is:

- Dynamics, change rate of
  - Component creation/deletion
  - Data Instance creation/deletion
  - Data value
- Ownership
- Access rights for users
  - No access, read, write access
  - Right to create, delete

10.1 Element property

Synopsis:

```
<property
      name = {propertyName}
>
  Content: [TBD]
</property>
```

11 Semantic and Comment

Comment elements are used to give comments in English to the Data Description itself.

Semantic is used to define the meaning of the main concepts used in Data Description. Examples are: Profile Component, Datatype, item in Datatype and specific value (in enumerations).

The semantic is given using normal language. It is possible to give it in several different languages.

Synopsis:

```
{semantic} ::= comment? semantic
{optSemantic} ::= comment? semantic?
```

11.1 Element comment

Synopsis:

```
<comment xml:lang="en">
  Content: [TBD]
</comment>
```

Comments elements are used to give comments to the Data Description. It is given in the English language.

11.2 Element semantic

Synopsis:

```
<semantic>
  Content: label+ definition? description?
</semantic>
```

The information in the semantic elements is used by the interpretation and usage of the described content. The semantic can be given in three different levels of detail:

- Label
  A human-readable label.
- Definition
  A statement that describes the essential nature of the element been described.
- Description
  Additional information (optional).

Synopsis:

```
<label xml:lang = {language} >
  Content:
</label>
```

Examples:

```
<label xml:lang="en">Understandable label</label>
<label xml:lang="se">Tolkbar etikett</label>
```

Synopsis:

```
<definition xml:lang = {language} >
  Content:
</definition>
```

Examples:

```
<definition xml:lang="en">
      A short definition</definition>
<definition xml:lang="se">
      En kort definition</definition>
```

Synopsis:

```
<description xml:lang = {language} >
  Content:
</description>
```

Examples:

```
<description xml:lang="en">
  A longer description ...
</description>
<description xml:lang="se">
  En längre beskrivning ...
</description>
```

Synopsis:

```
{languageId} ::= {xsi:language}
```

Language represents natural language identifiers as defined by [RFC 1766].

12 Datatype Description

12.1 Definitions

12.1.1 Datatype

[From XML-schema specification]

In this specification, a datatype is a 3-tuple, consisting of a) a set of distinct values, called its ·value space·, b) a set of lexical representations, called its ·lexical space·, and c) a set of ·facets that characterize properties of the ·value space·, individual values or lexical items.

12.1.2 Atomic datatypes

Atomic datatypes are those having values, which are regarded by as being indivisible or not further decomposable.

12.1.3 Predefined Atomic datatypes

Predefined Atomic datatypes are atomic datatypes, which are defined in this specification.

12.1.4 Derived Atomic datatypes

Derived Atomic datatypes are Atomic datatypes derived from the Atomic predefined datatypes by constraining them or by defining a union of Atomic datatypes

12.1.5 Composite datatypes

Composite datatypes are defined using atomic and other composite datatypes.

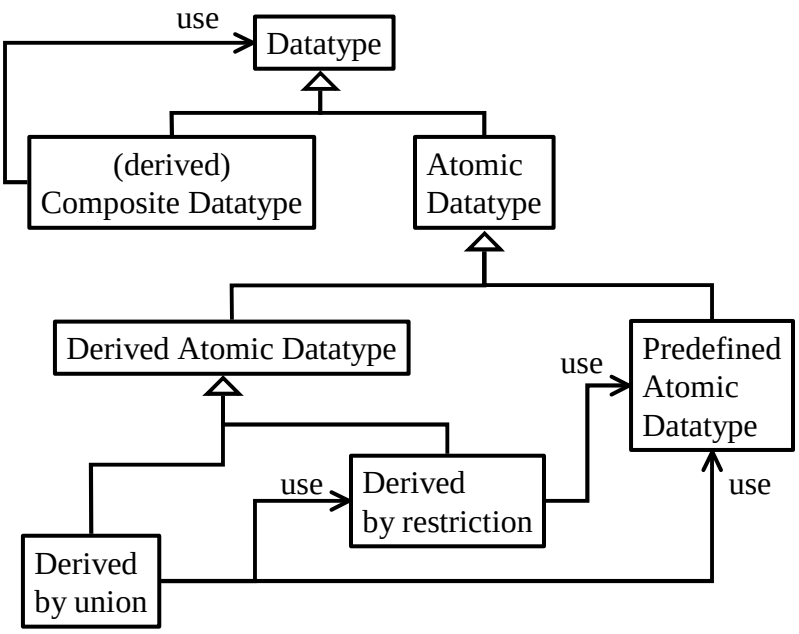


Fig xx Relationship between kinds of datatypes

Synopsis:

```
{datatypeDefinition} ::= atomicType | compositeType
```

{datatypeName} ::= [TBD]

12.2 Atomic Datatypes

12.2.1 Introduction

Atomic datatypes are those having values, which are regarded by as being indivisible or not further decomposable.

There are predefined atomic datatypes and derived atomic datatypes. Derived datatypes can be defined by restricting a predefined atomic datatype or by defining a union datatype.

12.2.2 Predefined atomic datatypes

Predefined atomic datatypes can only be added by revisions to this specification.

The XML-schema primitive datatypes are: string, boolean, decimal, float, double, duration, dateTime, time, date, gYearMonth, gYear, gMonthDay, gDay, gMonth, hexBinary, base64Binary, anyURI, QName, and NOTATION.

The XML-schema primitive derived datatypes are: normalizedString, token, language, NMTOKEN, NMTOKENS, Name, NCName, ID, IDREF, IDREFS, ENTITY, ENTITIES, integer, nonPositiveInteger, negativeInteger, long, int, short, byte, nonNegativeInteger, unsignedLong, unsignedInt, unsignedShort, unsignedByte, positiveInteger.

The predefined atomic datatypes are a subset of the XML-schema primitive datatypes.

The datatypes in the comments "{* … *}" below are for the moment excluded.

```
{predefinedAtomicDatatype} ::=
string | boolean
          {*| decimal | float | double*}
| duration | dateTime | time | date
          {*| gYearMonth | gYear | gMonthDay | gDay | gMonth*}
          {*| hexBinary | base64Binary*}
| anyURI
          {*| QName | NOTATION*}
| normalizedString
          {*| token *}
| language
          {*| NMTOKEN | NMTOKENS | Name | NCName*}
| ID | IDREF
          {*| IDREFS | ENTITY | ENTITIES*}
          {*| integer | nonPositiveInteger | negativeInteger | long*}
| int | short | byte
          {*| nonNegativeInteger | unsignedLong*}
```

```
| unsignedInt | unsignedShort | unsignedByte
          {*| positiveInteger*}
```

12.2.3 Derived Atomic Datatypes

Synopsis:

```
<atomicType name = {datatypeName} >
      Content: {optSemantic} ({restriction} | {union})
</atomicType>
```

Derived atomic datatypes can be defined by restricting a predefined atomic datatype or by defining a union datatype.

Synopsis as expanded XML-schema:

```
<simpleType
      final="list","restriction"
      id = [TBD]
      name = "{dataTypeName}"
>
  {optSemantic} ({restriction} | {union})
</simpleType>
```

Example:

12.2.4 Atomic datatypes derived by restriction

[From XML-schema specification]

A datatype is said to be derived by restriction from another datatype when values for zero or more constraining facets are specified that serve to constrain its value space and/or its lexical space to a subset of those of its base type. A constraining facet is an optional property that can be applied to a datatype to constrain its ·value space.

Note: Atomic datatypes derived by restriction can only be derived directly from Predefined atomic datatypes and not as restriction on derived atomic datatypes as in XML-Schema.

Synopsis:

```
<restriction base = {predefinedAtomicDatatype} >
  Content: {optSemantic} ({constrainingFacet})*
</restriction>
```

Example:

```
<atomicType  name="more-than-ninety-nine">
             <restriction base="int">
   <minExclusive value='99'/>
 <restriction/>
</atomicType>
```

Example data in XML-format:

```
100
```

Synopsis as expanded XML-schema:

```
<xs:simpleType name="{datatypeName}">
  <xs:restriction base="{predefinedAtomicDatatypeName}">
    {constrainingFacet}
  </xs:restriction>
</xs:simpleType>
```

Example as expanded XML-schema:

```
<xs:simpleType name='more-than-ninety-nine'>
  <xs:restriction base='int'>
    <xs:minExclusive value='99'/>
  </xs:restriction>
</xs:simpleType>
```

12.2.5 Constraining Facets

Constraining Facets in XML–schema are: length minLength maxLength pattern enumeration

whiteSpace maxInclusive maxExclusive minExclusive minInclusive

totalDigits fractionDigits.

[Issue: Which XML-schema Constraining Facets to select to be used]

Synopsis:

```
{constrainingFacetTag} =: minExclusive | minInclusive | maxExclusive |
maxInclusive | totalDigits | fractionDigits | length | minLength |
maxLength | enumeration | whiteSpace | pattern
```

Synopsis:

```
<{constrainingFacetTag}  value>
      Content: {optSemantic}
</{constrainingFacetTag}>
```

Example:

12.2.6 Union datatype

A union type enables an attribute value to be one instance of one type draw from the union of multiple atomic.

[From XML-schema specification] Union datatypes are those whose ·value spaces and ·lexical spaces are the union of the ·value spaces and ·lexical spaces of one or more other datatypes.

The datatypes that participate in the definition of a union datatype are called member types of that union datatype.

Synopsis:

```
<union>
      Content: {optSemantic} member*
</union>

<member  ref = {atomicNonUnionDatatype} >
      Content: {optSemantic}
</member>
```

Example:

```
<atomicType name="booleanOrDate">
  <union >
      <member ref="xsi:boolean"/>
      <member ref="xsi:date"/>
  </union>
</atomicType>
```

Example data in XML-format:

```
<someTag xsi:type="xsi:boolean">false</someTag>
```

or

```
<someTag xsi:type="xsi:date">1948-10-11</someTag>
```

Synopsis as expanded XML-schema:

```
<union id=”ID”>
  <simpleType ref="{atomicNonUnionDatatype}">
      {optSemantic}
  </simpleType>
  <simpleType ref= …>
      …
  </simpleType>
  …
</union>
```

Example as expanded XML-schema:

```
<atomicType name="booleanOrDate" … >
  <union … >
      <simpleType ref="xsi:boolean"/>
      <simpleType ref="xsi:date"/>
  </union>
</atomicType>
```

12.3 Composite Datatypes

12.3.1 Introduction

A composite datatype contains a number of name items each with a defined datatype.

```
{compositeType} ::= recordType
```

12.3.2 Record datatype

A record datatype contains a number of named items called fields each with a defined datatype. The field names must be unique with a record datatype. The datatype of a field can be any atomic data type or composite datatype.

Synopsis:

```
<recordType name = {datatypeName} >
      Content: {optSemantic} (field | fieldVector)*
</recordType>
```

Synopsis:

```
{fieldName} ::= [TBD]
```

There are two types of fields: Simple field and vector field. A simple field contain on instance of the field datatype. A vector field contains a vector or a number of instances of the datatype.

Example:

```
<recordType name="screenCoordinate">
  …
</recordType>
```

Synopsis as expanded XML-schema:

```
<xs:complexType name="{datatypeName}">
  <xs:sequence>
    …
  <xs:sequence>
</xs:complexType>
```

Example as expanded XML-schema:

```
<xs:complexType name=" screenCoordinate">
  <xs:sequence>
    …
```

```
  <xs:sequence>
</xs:complexType>
```

12.3.3 Simple field

Synopsis:

```
<field name="{fieldName}" dataType="{dataTypeName}"/>
```

Example:

```
<recordType name="screenCoordinate">
 <field name="x" datatype="xCoordinate"/>
 <field name="y" datatype="yCoordinate"/>
</recordType>
```

Synopsis as expanded XML-schema:

```
<xs:element name="{fieldName}" type="{datatypeName}"/>
```

Example as expanded XML-schema:

```
<xs:complexType name=" screenCoordinate">
  <xs:sequence>
   <xs:element name="x" type=" xCoordinate"/>
   <xs:element name="y" type=" yCoordinate"/>
  <xs:sequence>
</xs:complexType>
```

Example data in XML-format:

```
<x>12</x>
<y>5</y>
```

12.3.4 Vector field

Synopsis:

```
<fieldVector
      name="{fieldName}"
      dataType="{datatypeName}
      minOccurs="{minOccur}"
      maxOccurs="{ maxOccur}"
```

>

Content: {optSemantic}

</fieldVector>

Example:

```
<fieldVector name="c" dataType="coordinate"
      minOccurs="3" maxOccurs="3"/>
```

Example data in XML-format:

```
<c index="1">10</x>
<c index="2">20</x>
      <c index="3">30</x>
```

Synopsis expanded XML-schema:

```
<xs:element name="{fieldName}" type="{datatypeName}"
      minOccurs="{minOccur}" maxOccurs="{maxOccur}">
      <xs:attribute name="index" type="xs:byte"/>
</xs:element>
```

Example expanded XML-schema:

```
<xs:element name="x" type="coordinate"
      minOccurs="3" maxOccurs="3">
      <xs:attribute name="index" type="xs:byte"/>
</xs:element>
```

Annex A (normative): XML-schema files

This annex is a placeholder for the XML-schema files that are part of the Data Description Framework. These files will be used for the creation and consistency check of the Data Descriptions.

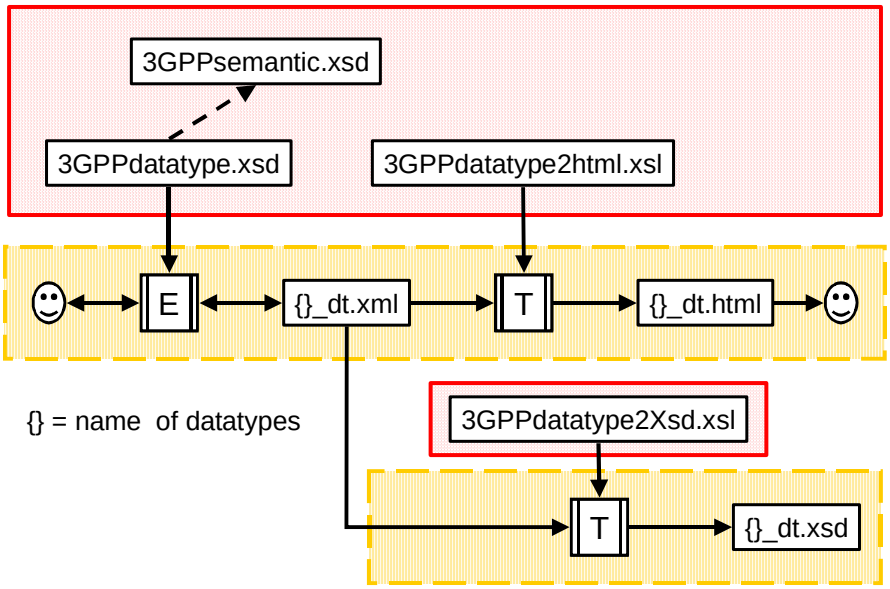
The following files represent XML-schemas. These files correspond to the definitions made above in this TS.

"3GPPsemantic.xsd" is the schema describing how to represent the semantics. "3GPPdatatype.xsd" is the schema describing how to represent the datatypes. "3GPPdatatype.xslt" is the definition of the translation between 3GPP representation of datatypes and the corresponding schema.

The following files are new version of the XML-schemas.

Please observe that the corresponding changes are NOT yet made to this specification.

The following picture shows the relationship between the files above and a datatype description file ({}_dt.xml), the corresponding html ({}_dt.html), and XML-schema ({}_dt.xsd) files.



Annex B (Informative): XML-schema in brief

XML-schema is a Schema definition language. The functionality is above and beyond what is provided by DTDs.

The W3C Recommendation consists of three parts:

- XML Schema Part 0: http://www.w3.org/TR/xmlschema-0/
  Primer is a non-normative document intended to provide an easily readable description of the XML Schema facilities, and is oriented towards quickly understanding how to create schemas using the XML Schema language. This primer describes the language features through numerous examples, which are complemented by extensive references to the normative texts.

- XML Schema Part 1, Structures: http://www.w3.org/TR/xmlschema-1/
  and

- XML Schema Part 2 ,Datatypes: http://www.w3.org/TR/xmlschema-2/
  provide the complete normative description of the XML Schema language.

B.1 XML-Schema Type System

The XML-schema Part 2 defines a Data Type System.

A Datatype is defined as follows:

- A Datatype is a 3-tuple, consisting of:

  a) a set of distinct values, called its·**value space**,

  b) a set of lexical representations, called its·**lexical space**, and

  c) a set of ·facet·s that characterize properties of the·value space, individual values or **lexical items**.

The definition of Boolean is:

- Boolean has the·value space·required to support the mathematical concept of binary-valued logic: {true, false}.

The lexical space of Boolean is defined:

- An instance of a datatype that is defined as ·boolean· can have the following legal literals {true, false, 1, 0}.

B.2 Examples of user defined types

To create a new type of integer called myInteger whose range of values is between 10000 and 99999 (inclusive) can be done by restricting the built-in simple type integer, whose range of values also includes integers less than 10000 and greater than 99999. To define myInteger, we restrict the range of the integer base type by employing two facets called minInclusive and maxInclusive:

```
<xsd:simpleType name="myInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="10000"/>
    <xsd:maxInclusive value="99999"/>
  </xsd:restriction>
</xsd:simpleType>
```

XML Schema defines fifteen facets. Among these, the enumeration facet is particularly useful and it can be used to constrain the values of almost every simple type, except the boolean type. The enumeration facet limits a simple type to a set of distinct values. For example, we can use the enumeration facet to define a new simple type called USState, derived from string, whose value must be one of the standard US state abbreviations:

```
<xsd:simpleType name="USState">
<xsd:restriction base="xsd:string">
   <xsd:enumeration value="AK"/>
   <xsd:enumeration value="AL"/>
   <xsd:enumeration value="AR"/>
   <!-- and so on ... -->
</xsd:restriction>
</xsd:simpleType>
```

New complex types are defined using the complexType element For example, USAddress is defined as a complex type, and within the definition of USAddress we see five element declarations.

```
<xsd:complexType name="USAddress" >
  <xsd:sequence>
   <xsd:element name="name"   type="xsd:string"/>
   <xsd:element name="street" type="xsd:string"/>
   <xsd:element name="city"   type="xsd:string"/>
   <xsd:element name="state"  type="xsd:string"/>
   <xsd:element name="zip"    type="xsd:decimal"/>
  </xsd:sequence>
```

```
                    </xsd:complexType>
```

B.3 DTD and XML-schema in the data Description architecture

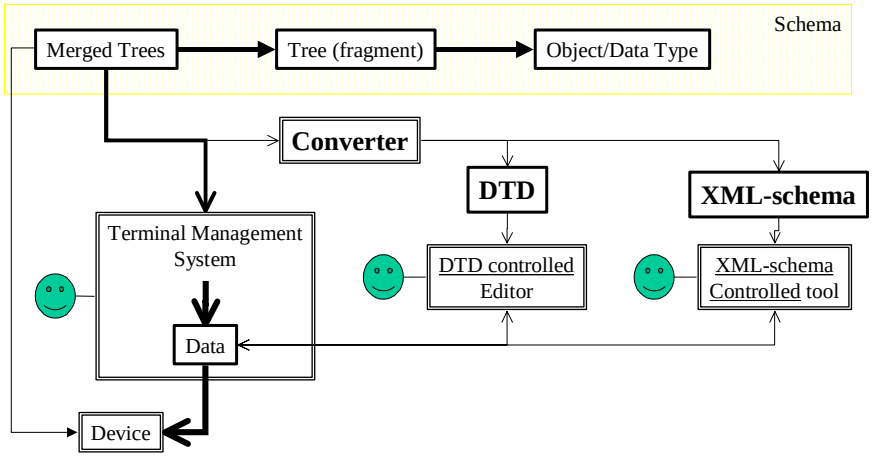
The following picture puts together the data description proposed with the DTD/XML-schema coding.

It also shows some usages of a Data Description:

- Terminal Management System
  The Data Description is defining the syntax and the format of the data sent to the device. The data values can be checked. The text describing the meaning of the parameters is fetched form the Data Description.

- Standard tools
  The Data Description can be translate to a XML DTD and an XML-schema. These can be used by tools, which understand DTDs and XML-schemas.

Data Description Architecture (2)



Annex C (Informative): Examples of Data Modelling Languages

Here follow a number of possible data modelling languages and principles.

Editor's note: Annex C is kept for reference for the time being.

C.1 ASN.1

ASN.1 defines the abstract syntax of information but does not restrict in any way, how the information is encoded.

There are various ASN.1 encoding rules, which provide transfer syntax (a concrete representation) of the data values, whose abstract syntax is described in ASN.1. The standard ASN.1 encoding rules include:

- BER (Basic Encoding+ Rules),
- CER (Canonical Encoding Rules),
- DER (Distinguished Encoding Rules) and
- PER (Packed Encoding Rules).

C.2Interface Definition Language, IDL

The OMG Interface Definition Language (IDL) is the language used to describe the interfaces that client objects call and object implementations provide. An interface definition written in OMG IDL completely defines the interface and fully specifies each operation's parameters. An OMG IDL interface provides the information needed to develop clients that use the interface's operations.

CORBA 2.4.2 OMG IDL Syntax and Semantics chapter http://www.omg.org/cgi-bin/doc?formal/01-02-39

C.3 Unified Modelling Language, UML

The Unified Modelling Language (UML) is a graphical language for visualising, specifying, constructing, and documenting the artifacts of a software-intensive system.

The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components.

OMG Modelling Specifications:
http://www.omg.org/technology/documents/formal/omg_modeling_specifications_avai.htm

UML Forum, a virtual community and knowledge portal that provides current information for modellers interested in UML: http://www.celigent.com/uml/

C.4 Document Type Definition, DTD and XML

The XML document type declaration contains or points to markup declarations that provide a grammar for a class of documents. This grammar is known as a document type definition, or DTD.

The DTD is defined in the XML specification.

XML home page: http://www.w3.org/XML/

Main specification: http://www.w3.org/TR/REC-xml

C.5 Resource Description Framework (RDF)

RDF integrates a variety of web-based metadata activities using XML as interchange syntax.

RDF home page: http://www.w3.org/RDF/

Model and Syntax Specification: http://www.w3.org/TR/REC-rdf-syntax/

Schema Specification 1.0: http://www.w3.org/TR/2000/CR-rdf-schema-20000327/

C.6 XML Schema

XML Schemas express shared vocabularies. It provides a means for defining the structure, content and semantics.

- XML home page: http://www.w3.org/XML/Schema
- XML Schema Part 0: Primer: http://www.w3.org/TR/xmlschema-0/
- XML Schema Part 1: Structures: http://www.w3.org/TR/xmlschema-1/
- XML Schema Part 2: Datatypes: http://www.w3.org/TR/xmlschema-2/

C.7 Composite Capability/Preference Profiles (CC/PP)

The W3C Metadata Activity addressed the combined needs of several groups for a common framework to express assertions about information on the Web. The primary work in this activity was the Resource Description Framework.

Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiation is one of the W3C Metadata Activities.

Here are some links:

http://www.w3.org/Metadata/

- CC/PP Working Group: http://www.w3.org/Mobile/CCPP/
- CC/PP home page: http://www.w3.org/TR/NOTE-CCPP/
- Composite Capabilities/Preference Profiles: Requirements and Architecture: http://www.w3.org/TR/2000/WD-CCPP-ra-20000721/
- Composite Capability/Preference Profiles (CC/PP): Structure: http://www.w3.org/TR/2000/WD-CCPP-struct-20000721/
- CC/PP Attribute Vocabularies: http://www.w3.org/TR/CCPP-vocab/

C.8 Common Information Model (CIM)

The DMTF http://www.dmtf.org/index.html Common Information Model (CIM) is an approach to the management of systems, software, users, networks and more, that applies the basic structuring and conceptualisation techniques of the object-oriented paradigm.

A management model is provided to establish a common conceptual framework for a description of the managed environment. A fundamental taxonomy of objects is defined — both with respect to classification and association, and with respect to a basic set of classes intended to establish a common framework.

The white paper about Common Information Model (CIM) Core Model:
http://www.dmtf.org/var/release/Whitepapers/DSP0111.htm gives a good introduction to CIM.

C.9 Language Independent Datatypes, LID

- ISO/IEC 11404:1995, Information technology - Language-Independent Datatypes
  http://pueblo.lbl.gov/~olken/mendel/w3c/iso11404.html
- ISO/IEC TR 10182:1994 - Binding Techniques for Programming Languages
- ISO/IEC 13886:1996 - Language Independent Procedure Calling
- ISO/IEC TR 14369:1999 - Guidelines for the Preparation of Language Independent Service Specifications
  http://wwwold.dkuug.dk/JTC1/SC22/WG11/docs/n455.rtf
- A taxonomy of datatypes http://www.kcl.ac.uk/kis/support/cit//staff/brian/taxosn.html

C.10 ISO/IEC 11179 - Specification and Standardization of Data Elements

http://pueblo.lbl.gov/~olken/X3L8/drafts/draft.docs.html

International Standard ISO/IEC 11179 parts are:

- Part 1: Framework for the Generation and Standardization of Data Elements;
- Part 2: Classification of Concepts for the Identification of Domains;
- Part 4: Rules and Guidelines for the Formulation of Data Definitions
- Part 5: Naming and Identification Principles for Data Elements; and
- Part 6: Registration of Data Elements.

Annex D (informative): Examples of Vocabularies

Here follow examples on Vocabularies.

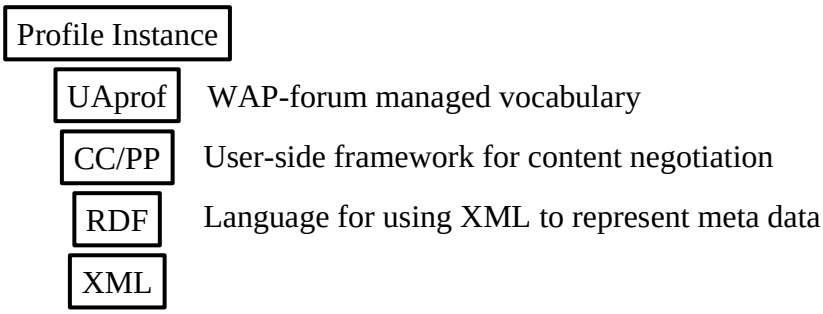
Editor's note: Annex D is kept for reference for the time being.

D.1 WAP UAProf

A WAP Forum specification. WAP UAProf Wireless Application Group, User Agent Profile Specification: http://www1.wapforum.org/tech/terms.asp?doc=SPEC-UAProf-19991110.pdf

WAP UAprof and CC/PP

- http://www.w3.org/Mobile/Activity
- http://www.w3.org/TR/CCPP-struct/
- http://www.w3.org/TR/CCPP-vocab/



D.2 SyncML device specific information

The DevInf.DTD is intended to be used to exchange device specific information. Exchange of device specific information such as available memory and item identifiers, supported local databases is a prerequisite to successful data synchronization. http://www.syncml.org/docs/syncml_devinf_v10_20001207.pdf

D.3 A Comparison of Schemas for Video Metadata Representation

This could give some inputs. http://www8.org/w8-papers/3c-hypermedia-video/comparison/comparison.html

D.4 vCard and vCalendar

vCard and vCalendar defines a transport and platform-independent format for exchanging personal information typically found on a traditional business card calendaring and scheduling information.

Personal Data Interchange (PDI) http://www.imc.org/pdi/

vCard in RDF draft-iannella-vcard-rdf-00.txt http://www.oasis-open.org/cover/draft-dawson-vcard-xml-dtd-00.txt

Annex E (informative): Example of Data Type Description

Data type “screenCoordinate”

This XML document describes a composite datatype screenCoordinate. It has two items x and y of type xCoordinate and yCoordinate. XCoordinate can have values 0 to 60.

```
<?xml version="1.0" encoding="UTF-8"?>

<dataTypes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\ecsbojn\proj\dataDescription\DDcurrent\3GPPdataType.xsd">


    <compositeType name="screenCoordinate">
        <item name="x" dataType="xCoordinate"/>
        <item name="y" dataType="yCoordinate"/>
    </compositeType>

    <atomicType name="xCoordinate" base="int">
        <minInclusive value="0"/>
        <maxInclusive value="60"/>
    </atomicType>

    <atomicType name="yCoordinate" base="int">
        <minInclusive value="0"/>
        <maxInclusive value="30"/>
    </atomicType>


</dataTypes>
```

XML representation of data

This XML document is an example of how data of datatype screenCoordinate can be represented.

The item names x and y is used as tags. The format of text in the x-tag and y-tag follows the XML-schema specification.

```
<?xml version="1.0" encoding="UTF-8"?>

<screenPositionS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\ecsbojn\proj\dataDescription\DDcurrent\dataTypeExample.xsd">

    <screenPosition>
        <x>12</x>
        <y>15</y>
    </screenPosition>
    <screenPosition>
```

```
        <x>5</x>
        <y>13</y>
    </screenPosition>
</screenPositionS>
```

Full XML-schema

This XML document is a XML-schema defining the constrains on a XML-document containing screen Positions. The part in **bold** text can automatically be generated from the data description of screenCoordinate.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.0 beta 3.1 build Aug 27 2001 (http://www.xmlspy.com) by Bo Johansson (Ericsson Mobile Communications AB) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="screenPositionS">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="screenPosition" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="screenPosition" type="screenCoordinate">
        <xs:annotation>
            <xs:documentation>Comment describing screenPosition</xs:documentation>
        </xs:annotation>
    </xs:element>

    <xs:complexType name="screenCoordinate">
        <xs:sequence>
            <xs:element name="x" type="xCoordinate"/>
            <xs:element name="y" type="yCoordinate"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="xCoordinate">
        <xs:restriction base="xs:int">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="60"/>
        </xs:restriction>
    </xs:simpleType>
```

```
    <xs:simpleType name="yCoordinate">
        <xs:restriction base="xs:int">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="30"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

Annex F (informative) Examples of Datatype Definitions

F.1. Introduction

This annex shows some examples of datatypes definition.

F.2. Test Datatype Definitions Examples

The datatypes tested are called:

- "more-than-ninety-nine"
- "booleanOrDate"
- "screenCoordinate"
- "wapId"

F.2.1 Test datatypes definition according to the UP-010089

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.0 U (http://www.xmlspy.com) by Bo Johansson (Ericsson Mobile Platforms AB) -->
<?xml-stylesheet type="text/xsl" href="3GPPdatatype.xslt"?>
<datatypes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="3GPPdatatype.xsd">
    <atomicType name="more-than-ninety-nine">
        <restriction base="int">
            <minExclusive value="99"/>
        </restriction>
    </atomicType>
    <atomicType name="booleanOrDate">
        <union>
            <member ref="xs:boolean"/>
            <member ref="xs:date"/>
        </union>
    </atomicType>
    <recordType name="screenCoordinate">
        <field name="x" datatype="coordinateX"/>
        <field name="y" datatype="coordinateY"/>
    </recordType>
    <atomicType name="coordinateX" base="int">
        <restriction base="int">
            <minInclusive value="0"/>
            <maxInclusive value="60"/>
        </restriction>
    </atomicType>
    <atomicType name="coordinateY" base="int">
        <restriction base="int">
            <minInclusive value="0"/>
            <maxInclusive value="30"/>
        </restriction>
    </atomicType>
    <atomicType name="wapId">
        <semantic>
```

```
            <label lang="en">Globally unique wap id</label>
            <definition xml:lang="en">
      Uniqueness MUST be obtained by either using a fully qualified Internet domain name
      (i.e. hostname as defined in section 3.2.2 of {RFC2396])
      or a globally unique IP address (IPv4 {RFC791] in decimal format with dots as delimiters
      or IPv6 {RFC2373], as hexadecimal numbers with colons as delimiters or as a combination of
hexadecimal and decimal numbers with dots and colons as delimiters)
      </definition>
            <description lang="en">
            http://www.ietf.org/rfc/rfc2396.txt
            http://www.ietf.org/rfc/rfc791.txt
            http://www.ietf.org/rfc/rfc2373.txt
      </description>
         </semantic>
         <union>
            <member ref="wapfqIDN"/>
            <member ref="IPv4Add"/>
            <member ref="IPv6Add"/>
         </union>
      </atomicType>
      <atomicType name="wapfqIDN">
         <semantic>
            <label lang="en">Fully qualfied host name</label>
            <description lang="en">
       http://www.ietf.org/rfc/rfc2396.txt
       </description>
         </semantic>
         <restriction base="string">
            <pattern value=".+"/>
         </restriction>
      </atomicType>
      <atomicType name="IPv4Add">
         <semantic>
            <label lang="en">IPv4 address</label>
            <description lang="en">
       http://www.ietf.org/rfc/rfc791.txt
             </description>
         </semantic>
         <restriction base="string">
            <pattern value="\d{1,3}(.\d{1,3}){3}"/>
         </restriction>
      </atomicType>
      <atomicType name="IPv6Add">
         <semantic>
            <label lang="en">IPv6 address</label>
            <description lang="en">
       http://www.ietf.org/rfc/rfc2373.txt
       </description>
         </semantic>
         <restriction base="string">
```

```
        <pattern value="[0-9a-fA-F]{0,4}(:[0-9a-fA-F]{0,4}){7}|[0-9a-fA-F]{0,4}(:[0-9a-fA-
F]{0,4}){5}(.\d{1,3}){4}"/>
      </restriction>
    </atomicType>
</datatypes>
```

F.2.2 Test datatypes as expanded XML-schema:

```
<?xml version="1.0" encoding="UTF-16" ?>
- <xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  - <xs:simpleType name="more-than-ninety-nine" final="list restriction">
    - <xs:restriction base="xs:int">
        <xs:minExclusive value="99" />
      </xs:restriction>
    </xs:simpleType>
  - <xs:simpleType name="booleanOrDate" final="#all">
      <xs:union memberTypes="xs:boolean xs:date" />
    </xs:simpleType>
  - <xs:complexType name="screenCoordinate">
    - <xs:sequence>
        <xs:element name="x" type="coordinateX" />
        <xs:element name="y" type="coordinateY" />
      </xs:sequence>
    </xs:complexType>
  - <xs:simpleType name="coordinateX" final="list restriction">
    - <xs:restriction base="xs:int">
        <xs:minInclusive value="0" />
        <xs:maxInclusive value="60" />
      </xs:restriction>
    </xs:simpleType>
  - <xs:simpleType name="coordinateY" final="list restriction">
    - <xs:restriction base="xs:int">
        <xs:minInclusive value="0" />
        <xs:maxInclusive value="30" />
      </xs:restriction>
```

```
  </xs:simpleType>
- <xs:simpleType name="wapId">
  - <xs:annotation>
      <xs:documentation>Globally unique wap id</xs:documentation>
      <xs:documentation>Uniqueness MUST be obtained by either using
        a fully qualified Internet domain name (i.e. hostname as
        defined in section 3.2.2 of {RFC2396]) or a globally unique IP
        address (IPv4 {RFC791] in decimal format with dots as
        delimiters or IPv6 {RFC2373], as hexadecimal numbers with
        colons as delimiters or as a combination of hexadecimal and
        decimal numbers with dots and colons as
        delimiters)</xs:documentation>
      <xs:documentation>http://www.ietf.org/rfc/rfc2396.txt
        http://www.ietf.org/rfc/rfc791.txt
        http://www.ietf.org/rfc/rfc2373.txt</xs:documentation>
    </xs:annotation>
    <xs:union memberTypes="wapfqIDN IPv4Add IPv6Add" />
  </xs:simpleType>
- <xs:simpleType name="wapfqIDN">
  - <xs:annotation>
      <xs:documentation>Fully qualfied host name</xs:documentation>
      <xs:documentation>http://www.ietf.org/rfc/rfc2396.txt</xs:do
      cumentation>
    </xs:annotation>
  - <xs:restriction base="xs:string">
      <xs:pattern value=".+" />
    </xs:restriction>
  </xs:simpleType>
- <xs:simpleType name="IPv4Add">
  - <xs:annotation>
      <xs:documentation>IPv4 address</xs:documentation>
      <xs:documentation>http://www.ietf.org/rfc/rfc791.txt</xs:doc
      umentation>
    </xs:annotation>
  - <xs:restriction base="xs:string">
      <xs:pattern value="\d{1,3}(.\d{1,3}){3}" />
```

```
      </xs:restriction>
    </xs:simpleType>
  - <xs:simpleType name="IPv6Add">
    - <xs:annotation>
        <xs:documentation>IPv6 address</xs:documentation>

        <xs:documentation>http://www.ietf.org/rfc/rfc2373.txt</xs:do
        cumentation>
      </xs:annotation>
    - <xs:restriction base="xs:string">
        <xs:pattern value="[0-9a-fA-F]{0,4}(:[0-9a-fA-F]{0,4}){7}|[0-
          9a-fA-F]{0,4}(:[0-9a-fA-F]{0,4}){5}(.\d{1,3}){4}" />
      </xs:restriction>
    </xs:simpleType>
  - <xs:element name="testDatatype">
    - <xs:complexType>
      - <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="more-than-ninety-nine" type="more-than-
            ninety-nine" />
          <xs:element name="booleanOrDate" type="booleanOrDate" />
          <xs:element name="screenCoordinate"
            type="screenCoordinate" />
          <xs:element name="coordinateX" type="coordinateX" />
          <xs:element name="coordinateY" type="coordinateY" />
          <xs:element name="wapId" type="wapId" />
          <xs:element name="wapfqIDN" type="wapfqIDN" />
          <xs:element name="IPv4Add" type="IPv4Add" />
          <xs:element name="IPv6Add" type="IPv6Add" />
        </xs:choice>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

F.2.3 Generated documentation of the test datatype XML-schema.

test_datatype_html.zip

Annex G (informative): Data Description Files and Tools

This annex is planned to contain an introduction of the files and tools that are related to the Data Description Framework and the development of data descriptions. The appended presentation will be used as the basis for this introduction.

Microsoft PowerPoint Presentation

Annex H (informative): Examples of GUP profiles

The appended document contains examples of GUP profile descriptions. Two small profiles are described using the Data Description Framework. When more realistic examples are available an annex with a similar structure will be created using these examples to give an introduction of how GUP profiles are created according to the Data Description Framework.

"T2GUP-020012 (Profile description).

Annex <X> (informative): Change history

It is usual to include an annex (usually the final annex of the document) for specifications under TSG change control which details the change history of the specification using a table as follows:

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
01-11-13		UP-010104			First draft of the specification from UP-010066		
01-11-14		UP-010106			Second Draft of the specification		
01-12-05		UP-010139			v.0.2.1 After UP#07 Editorial changes based on UP-010118: chapter 11.1 moved after chapter 5, becoming chapter 6		
02-02-07		T2GUP-020004 T2GUP-020012			v.0.3.0 after T2GUP#1 Annex A: New XML-schema files added. New Annex G added, describing data description related files and tools. New Annex H added, containing examples of GUP profiles		

T2#16 **T2-020109**
Sophia Antipolis, France
11-15 February 2002

3GPP TS 24.241 V0.~~2~~3.0 (~~2001~~2002-~~12~~02)

3rd Generation Partnership Project;
Technical Specification Group Terminals;
3GPP Generic User Profile Common Objects;
Stage 3
(Release x)

Reference
3TS/TSGT-02xxxxx

Keywords
R4 Specification 3G

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

Foreword

This Technical Specification has been produced by the 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 this TS, 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 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 specification;

1 Scope

This specification serves as a vessel to manage the process of adding new datatypes, Generic User Profile fragments, and other constructs for use in 3GPP applications within various specifications.

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.
- A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

[1] 3GPP TS 27.103 V4.1.0 (2001-04), 3rd Generation Partnership Project; Technical Specification Group Terminals; Wide Area Network Synchronisation Standard (Release 4)

[2] 3GPP 22.240

[3] 3GPP 23.240

[4] 3GPP 23.241

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

vObject: ….

data store ..

Datatype

GUP fragment

…

3.2 Abbreviations

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

[tbd]

4 Background

[Editor's note: to be redrafted]

The request for data synchronisation support for the VHE MExE User Profile extensions brought up the long term need to define standards for and manage the process of adding new vObjects and other constructs as data store types for use in data synchronisation activities. Managing this process targets the following areas:

- Definition of new vObject and Other Constructs standardised formats for use in data synchronisation as required by other groups within 3GPP.
- Management of the process of publishing these new standardised formats for use within and external to 3GPP.
- Support of terminal and network interoperability through the use of a standardised approach to the definition of these new formats.
- Extension of the usefulness of the TSG-T2-defined data synchronisation architecture and mechanisms to new data store semantic content.
- Identification of required protocols and development, if needed

The vObjects and other constructs listed in Section 6 should enhance interoperability and be implemented in a way that ensures backwards compatibility, where possible.

Standardised vObject and Other Construct formats must allow users and operators to keep local copies up to date with remotely stored copies of the user's and the operator's mission-critical data stores in a manner that will allow data synchronisation to a wide variety of potentially disparate data stores. These standardised formats must allow rapid expansion of the nature and type of future data store enhancements.

Data synchronisation of vObjects and Other Constructs should standardise charging mechanisms, especially in roaming situations and between different operators. Other charging mechanisms (e.g. air time) may be needed when data synchronization of vObjects and Other Constructs is attempted outside of the operator's domain.

5 Process for Addition of New Datatypes

[tbd]

The appended presentation is a first try to describe the work process to be used when defining new datatypes. When this and the following two chapters are developed, ideas can be fetched from this presentation.

Microsoft PowerPoint Presentation

6 Process for Addition of New Fragments

[tbd]

7 Process for Addition of New Common Objects

[tbd]

8 Process for Addition of New vObjects or Other Constructs

[Editor's Note: to be redrafted]

8.1 New vObjects or Other Constructs

New vObjects or other constructs shall be defined in a stand-alone 3GPP specification. This specification may be a wholly self-contained definition or it may simply be a reference to an independent SDO's specification, where such exists.

8.2 Formal Recognition

Formal recognition of the new vObject or other construct shall be through the use of a CR to this specification requesting the addition to Section 6 of the specification of the vObject or other construct to be recognized.

8.3 Approval Process

[tbd]

Annex A (Normative)

Generic User Profile Components

Annex A1 Datatypes

Annex A2 GUP Fragments

Annex A3 Other Constructs

Annex B (Informative)

Parameters for Component Construction

Annex B1 GPRS Parameters

Parameter Requirements | Data Description | Generated Documentation | Generated XML Schema


UP-010091 (GPRS parameters).zip


gprsDatatype.xml


gprsDatatype_dt.xml


gprsDatatype.htm


gprsDatatype_dt.html


gprsDatatype.xsd


gprsDatatype_dt.xsd

[Please note that these files are included here for information only. The final structure (i.e., to include the object files within this specification or to reference them as external specifications) has not been decided.

Please note, also, that these are very early, representative drafts. More work must be done by experts on GPRS prior to these being used by developers.]

Annex B2 Subscription Management

[tbd]

Annex B3 MMS Parameters

Parameter Requirements | Data Description | Generated Documentation | Generated XML Schema


"T2GUP-020007 (MMS parameters).d

~~[tbd]~~

Please note that this file is included here for information only.

Please also note that this is a very early draft and more work must be done by experts on MMS.

Annex C (Normative)

Recognised vObjects and Other Constructs

Specification	Title	Comment(s)
(T2-000676)	Bookmark	Bookmark URL standard

Annex D:
Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2001-11					Submitted by T2 to TSG-T for preliminary information		0.1.0
2001-12					Added GPRS examples to Annex B	0.1.0	0.2.0
2002-02					After T2GUP#1 Example of work procedure added to chapter 5 In annex B1the files gprsDatatype.xml gprsDatatype.html and gprsDatatype.xsd are replaced by new versions. In annex B3 draft of~~to~~ data requirements added	0.2.0	0.3.0