Technical Document

G23-UMTS PROTOCOL STACK

SYNTAX DESCRIPTION FOR AIR INTERFACE MESSAGE DOCUMENTS

USER GUIDE

Document Number:	06-03-22-UDO-0001
Version:	0.4
Status:	Draft
Approval Authority:	
Creation Date:	2001-Jul-20
Last changed:	2015-Mar-08 by XGUTTEFE
File Name:	8350_300_MSG_Syntax.doc

Important Notice

Change History

Date	Changed by	Approved by	Version	Status	Notes
2001-Jul-20	MVJ		0.1	Being Processed	1
2001-Aug-24	MVJ		0.2	Being Processed	
2001-Sep-06	MVJ		0.3	Being Processed	
2003-May-07	XGUTTEFE		0.4	Draft	

Notes:

1. Initial version

Table of Contents

List of Figures and Tables

List of References

[ISO 9000:2000]	International Organization for Standardization. Quality management systems - Fundamentals and vocabulary. December 2000

1 Introduction

This document describes how to write Condat Air Interface Message documents.

Air interface messages are peer-to-peer messages between a MS and its network peer. These messages are standardized by ETSI/3GPP, and thus have well defined formats.

The Condat Air Interface Message documents describe how data in air interface messages are organized, and how they can be used in entities. They operate at bit-level as opposed to Service Access Points (SAPs) which normally operate at byte level.

Air Interface Messages are documents written in Word as part of the high level design phase. When the air interface messages they describe are needed in actual code, the documents are run through the Condat tool chain (FIXME: reference) which produce header files and other data needed in program code.

This document is divided into sections, which describe how the Air Interface Message documents are organized, and what possibilities are available in each section.

2 Message Document Structure

The Air Interface Message document is created from a Condat Word template (FIXME: reference) by choosing "air interface message" document type when invoking the template.

This will create the title page, initialise the document history, and an introductory headline.

The rest of the document sections must be structured as in the list below. The order must be preserved, but some sections may be left out. These sections are marked [optional].

2	Constants [optional]	Contains global constants
3	Types	Lists coding types used (e.g. type 4 TLV)
4	Messages	The actual air interface message descriptions (e.g. ATTACH REQUEST)
5	Structured Elements [optional]	Elements in air interface messages (e.g. MS class mark)
6	Basic Elements	Basic types/values (e.g. MS type 2)

The section numbering shown in the list above is the one traditionally use in Condat Air Interface Message documents, so it is recommended to stick to it.

The sections above contain a number of subsections which act as keywords, separating different types of information in each subsection, see Figure 1.

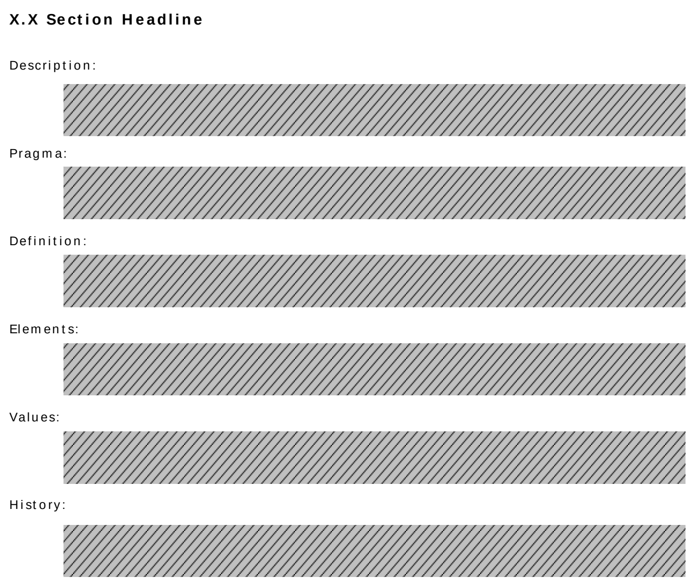

Figure 1 – Use of subsections as keywords in an AIM document

All subsections are listed in section 8.1.

Most sections have an associated Word style in Air Interface Message documents, i.e. a table of elements has Word style "ElementsTabelle". In each section where such a style is applicable, it will be mentioned. While using these styles is not mandatory, you are strongly urged to do so – it will make Condat's planned transition from Word documents into XML much easier.

The air interface message document sections listed above are described in detail in the following chapters.

3 Constants

This section contains information about global constants used in the document. The section must contain a table of constants, and a "History" list to track changes.

3.1 Description of Subsections

In the Constants section, the following subsections are possible:

- Description
- Pragma
- Definition
- History

The use of each subsection is described below.

3.1.1 Description

The "description" subsection is optional (but strongly recommended) – It contains a textual description of the information in the contants table below.

3.1.2 Pragma

This section is optional. It is used for modifying the behaviour of the Condat tool chain, e.g. for defining a prefix for identifiers (constants, values, etc).

For more information, please refer to section 8.1.2.

3.1.3 Definition

The Definition table in the constant section is similar to Basic Elements (section 7) definitions. It contains a **name** column, a **value** column, and optionally a **comment** column.

The value from the **value** column is then bound to the identifier named by entry in the **name** column.

3.1.4 History

The "History" subsection contains a brief changelog for the table. For more information, see the "History" section.

3.2 Example

The example below shows a section defining the constant "L3MAX" to the value "251". The **value** field may specify the constant value in a number of formats, e.g. decimal (including negative values), hexadecimal, binary etc. Please see 8.2.15 for a complete list.

Definition:

name	value	comment
L3MAX	251	maximum size of a L3 buffer

History:

01-Jan-2001	DEV	Initial

4 Types

This section contains information about the air interface message coding types used in the document. The section must contain a table of coding types, and a "History" list to track changes.

4.1 Description of Subsections

In the Types section, the following subsections are possible:

- Description
- Definition
- History

The use of each subsection is described below.

4.1.1 Description

The "description" subsection is optional (but strongly recommended) – It contains a textual description of the information in the contants table below.

4.1.2 Definition

The Definition table in the types section contains a **name** column, an **add bit** column, and optionally **ctrl** and **comment** columns.

For each type mentioned in the **name** column, the **add bit** column tells how many bits are allocated to type and length fields, and the **ctrl** field may set a type to be optional by default.

For example, a GSM1_TV type is a type 1 IE with 4 extra bits of type information. Therefore, its **add bits** column should be set to 4. The GSM1_TV type is also optional by default, so its **ctrl** column could be set to "optional".

4.1.3 History

The "History" subsection contains a brief changelog for the table. For more information, see the "History" section.

4.2 Example

Definition:

name	add bit	ctrl	comment
GSM1_V	0		mandatory IE V-component in one nibble
GSM1_TV	4	optional	optional with V-Component
GSM2_T	8	optional	optional IE, contains IEI only (no V-component)
GSM3_V	0		mandatory / conditional, no IEI, V-component only
GSM3_TV	8	optional	optional IE with V-component
GSM4_LV	8		mandatory / conditional, length and V-component
GSM4_TLV	16	optional	optional IE with length indicator and V-component
GSM5_V	0	optional	optional IE (bitstream to end of message)
BCD	4		binary coded decimal number
BCDODD	4		binary coded decimal number starting with digit1
BCDEVEN	4		binary coded decimal number starting with digit2

History:

01-Jan-2001	DEV	Initial

5 Messages

This section defines the air interface message descriptions. Typically, the air interface message definitions contain a structure of a number of complex and simple elements.

Each air interface interface definition subsection must contain a "Definition" table, and a "History" list to track changes. Having a "Description" subsection is also strongly recommended.

5.1 Description of Subsections

In the Messages section, the following subsections are possible:

- Description
- Definition
- Elements
- History

The use of each subsection is described below.

5.1.1 Definition

The "definition" subsection defines the name of an air interface message, its type, and which direction(s) (uplink/downlink) it applies. Optionally, a field may define when to include the message in the output (using **Version**) or, if the element is defined in another document, a **Link** column detailing where to find it. In case the message is included using the link facility, no elements are allowed.

The "definition" subsection consists of a subsection caption and a definition table which contains the information mentioned above. The subsection caption must be "Definition:" and have Word type "Definition".

The "Definition" table in the messages section contains a **Long name** column, a **Short name** column, an **ID** column, and a **Direction** column. Optionally, **Version** and **Comment** columns may be present.

The **Long name** column is informational only. The **Short name** column names the C identifier, with which this message is associated (C struct). This struct should be used in program code to read from/fill in the message structure.

The **ID** column contains the air interface message msgid (message id) as defined in ETSI/3GPP specs. Used to differentiate messages, and code/decode messages appropriately.

Finally, the **Direction** column specifies whether a message can travel "uplink", "downlink", or "both".

5.1.2 Elements

The “elements” subsection defines which structured and/or basic elements form the air interface message, as well as their coding type, optionality, and size. These elements correspond to IEs (information elements) in 3GPP specs. Optionally, a field may define when to include the message in the output (using **Version**).

The “elements” subsection consists of a subsection caption and a elements table which contains the information mentioned above. The subsection caption must be “Elements:” and have Word type “Elements”.

A minimal ”Elements” table (which must be of Word type “ElementsTabelle”) in the messages section contains a **Long name** column, a **Short name** column, a **Reference** column (typically abbreviated to “**ref**”), a **Type** column, and a **Length** column (typically abbreviated to “**len**”). Optionally, an **ID** defines IE identifiers for optional TLV elements. Elements may be declared optional using a **Pres** column. Optional inclusion of an element can be realized using a **Version** column, and finally a **Comment** column can give further information about the elements. For easy back-tracking, a **Spec ref** column is recommended. It should point to the 3GPP specification that defines the element in question.

The **Long name** column is informational only. The **Short name** column names the C identifier, with which this element is associated (in C). This struct should be used in program code to read from/fill in the message structure.

The **Reference** column tells where to find the element definition, in case it is not a spare. Spares have no reference (See **Short name** for more information on spares). The reference column may be substituted with a **Link** column, if the element is defined outside the current document.

Type is the coding type to use for the element. It may be any of the type suffixes defined in the Types section, i.e. the types without “GSM” and type number (BCD, 1, 2, …). This usually boils down to a “[T][L]V” suffix. For example, an optional type 4 TLV element would have a **Type** of “TLV”.

Either **Bit len** or **Length** must be present and define the bit or byte length of the element, respectively. The length may be variable. For example, a type 4 TLV element which can be from 4 to 16 bytes would have a **Length** field of “4-16”.

The **ID** column contains the information element identifier (IEI) as defined in ETSI/3GPP specs. An element with a value in the **ID** column is automatically marked as optional, as its presence depends whether the IEI is included in a message.

5.1.3 History

The “History” subsection contains a brief changelog for the table. For more information, see the “History” section.

5.2 Example

The table below shows an example of an air interface message section (Session Management ACTIVATE PDP CONTEXT ACCEPT message).

Note: The **ref** column is for example only – it refers to sections in the originating document, and its content is invalid here.

Description:

This message is sent by the network to the MS to acknowledge activation of a PDP context.
If the MS did not request a static address in the corresponding ACTIVATE PDP CONTEXT REQUEST message, the network shall include the PDP address IE in this message, otherwise not. Protocol configuration options are included in the message when the network wishes to transmit protocol configuration options for the external PDN. Likewise for the GSM only packet flow identifier (PFI).
Reference : [3G 24.008, 9.5.2]

Definition:

long name	short name	ID	direction
activate PDP context accept	activate_pdp_acc	66	downlink

Elements:

ID	long name	short name	ref	spec ref	pres	type	len	version
	message type	msg_type	6.4	[3G 24.008, 10.4]	M	V3	1	
	negotiated LLC SAPI	llc_sapi	5.2	[3G 24.008, 10.5.6.9]	M	V	1	
	negotiated QoS	qos	5.3	[3G 24.008, 10.5.6.5]	M	LV	12	
	radio priority	radio_prio	5.7	[3G 24.008, 10.5.7.2]	M	V	½	
	spare half octet	.0000		[3G 24.008, 10.5.1.8]	M	V	½	
0x2B	packet data protocol address	address	5.4	[3G 24.008, 10.5.6.4]	O	TLV	4-20	
0x27	protocol configuration options	pco	5.6	[3G 24.008, 10.5.6.3]	O	TLV	3-253	
0x34	packet flow identifier	pfi	5.9	[3G 24.008, 10.5.6.11]	O	TLV	3	R99 | UMTS

History:

01-Jan-2001	DEV	Initial

6 Structured Elements

This section contains information about the complex elements used in air interface message descriptions. Typically, the complex element definitions contain a structure or array of (possibly optional) bit-coded elements. This section is optional – that is if there are no complex parameters in any of the air interface messages, this sections should be skipped.

While identical types may be declared in the same section, all different element definitions must be declared in a separate subsection (except for definitions imported from other documents; see 8.2.7). Each subsection must contain a "Definition" table, and a "History" list to track changes. Having a "Description" subsection is also strongly recommended.

6.1 Description of Subsections

In the Structured Elements section, the following subsections are possible:

- Description
- Definition
- Elements
- History

The use of each subsection is described below.

6.1.1 Definition

The "definition" subsection defines the name of a structured element, and its type and size. Optionally, a field may define when to include the message in the output (using **Version**) or, if the element is defined in another document, a **Link** field detailing where to find it. In case the element is included using the link facility, no other element content is allowed (i.e. elements or values).

The “definition” subsection consists of a subsection caption and a definition table which contains the information mentioned above. The subsection caption must be “Definition:” and have Word type “Definition”.

The definition table (which must have type “DefinitionTabelle”) must have the following columns: **Short name**, **Type**, and **Bit len** or **Length** (the latter typically abbreviated to “**len**”).

The **Short name** column names the identifier with which this basic element is associated. This will end up being the name of the C struct containing the elements defined in the table.

Type is the coding type to use for the element. It may be any of the type numbers defined in the “Types” section, i.e. without “GSM” and “[T][L]V”. The “[T][L]V” suffix is determined where the structured element is used (i.e. in the air interface message). For example, an optional type 4 TLV element would have a **Type** of “4”.

Either **Bit len** or **Length** must be present and define the bit or byte length of the structured element, respectively. The length may be variable. For example, a type 4 TLV element which can be from 4 to 16 bytes would have a **Length** field of “4-16”.

Optionally, a **Version** column may dictate when to include the basic element.

6.1.2 Elements

The “elements” subsection defines the basic elements that form the structured element. For each element, their names and sizes are defined, and there must be a reference where to find them. Optionally, a field may declare elements optional, while a **Version** column may define when to include the message in the output (using) or, if the element is defined in another document, a **Link** field detailing where to find it.

The “elements” subsection consists of a subsection caption and a elements table which contains the information mentioned above. The subsection caption must be “Elements:” and have Word type “Elements”.

A minimal ”Elements” table (which must be of Word type “ElementsTabelle”) in the messages section contains a **Long name** column, a **Short name** column, a **Reference** column (typically abbreviated to “**ref**”), and a **Bit len** column. Elements may be declared optional using a **Pres** column. Optional inclusion of an element can be realized using a **Version** column, and finally a **Comment** column can give further information about the elements.

The **Long name** column is informational only. The **Short name** column names the C identifier, with which this element is associated (in C). This struct should be used in program code to read from/fill in the message structure.

The **Reference** column tells where to find the basic element definition, in case it is not a spare. Spares have no reference (See **Short name** for more information on spares). The reference column may be substituted with a **Link** column, if the element is defined outside the current document.

Type is the coding type to use for the element. It may be any of the type suffixes defined in the “Types” section, i.e. the types without “GSM” and type number (BCD, 1, 2, ...). This usually boils down to a “[T][L]V” suffix. For example, an optional type 4 TLV element would have a **Type** of “TLV”.

Either **Bit len** or **Length** must be present and define the bit or byte length of the basic element, respectively. The length may be variable. For example, a type 4 TLV element which can be from 4 to 16 bytes would have a **Length** field of “4-16”.

6.1.3 History

The “history” subsection contains a brief changelog for the table. For more information, see the “History” section.

6.2 Example

The example below shows the structured element definitions for the Session Management “Quality of Service” parameter. Note: The “**ref**” column is for example only – it refers to sections in the originating document, and its content is invalid here.

Description:

The purpose of the *quality of service* information element is to specify the QoS parameters for a PDP context.
Reference : [3G 24.008, 10.5.6.5]

Definition:

long name	short name	type	len
quality of service	qos	4	12

Elements:

long name	short name	ref	bit len	version
spare	.00		2	
delay class	delay	6.2	3	
reliability class	reliability	6.3	3	
peak throughput	peak	6.6	4	
spare	.0		1	
precedence class	precedence	6.7	3	
spare	.000		3	
mean throughput	mean	6.8	5	
traffic class	tc	6.9	3	R99 | UMTS
delivery order	order	6.10	2	R99 | UMTS
delivery of erroneous SDU	del_err_sdu	6.11	3	R99 | UMTS
maximum SDU size	max_sdu	6.12	8	R99 | UMTS
maximum uplink bit-rate	max_br_ul	6.13	8	R99 | UMTS
maximum downlink bit-rate	max_br_dl	6.14	8	R99 | UMTS
residual BER	ber	6.15	4	R99 | UMTS
SDU error ratio	sdu_err_ratio	6.16	4	R99 | UMTS
transfer delay	xfer_delay	6.17	6	R99 | UMTS
traffic handling priority	handling_pri	6.18	2	R99 | UMTS
guaranteed upink bit-rate	guar_br_ul	6.19	8	R99 | UMTS
guaranteed downlink bit-rate	guar_br_dl	6.20	8	R99 | UMTS

History:

01-Jan-2001	DEV	Initial

7 Basic Elements

This section contains information about the basic elements used in air interface message descriptions. Typically, the basic element definitions contain a single bit-coded element with a number of possible values.

While identical types may be declared in the same section, all different element definitions must be declared in a separate subsection (except for definitions imported from other documents; see 8.2.7). Each subsection must contain a ”Definition“ table, and a “History” list to track changes. Having a “Description” subsection is also strongly recommended.

7.1 Description of Subsections

- Description
- Definition
- Values
- History

The Values subsection is mandatory only when actually defining values.

The use of each subsection is described below.

7.1.1 Definition

The "definition" subsection, as the name says, defines the basic element. In fact, it defines the basic element's name and (bit) size, and optionally when to include it (using **Version**) or, if the element is defined in another document, a **Link** field detailing where to find it. In case the element is included using the link facility, no other element content is allowed (i.e. elements or values).

The "definition" subsection consists of a subsection caption and a definition table which contains the information mentioned above. The subsection caption must be "Definition:" and have Word type "Definition".

The definition table (which must have type "DefinitionTabelle") must have the following columns: **Short name** and **Bit len** or **Length**. The **Short name** column names the identifier with which this basic element is associated. Typically, this will end up being a byte/short/int in a structure which uses this basic element. Either **Bit len** or **Length** must be present and define the bit or byte length of the basic element, respectively.

Optionally, a **Version** column may dictate when to include the basic element.

7.1.2 Values

If the basic element has values, they must be declared in a "values" subsection. The "values" subsection consists of a values subsection caption and a value table.

The table contains value definitions, and how these values are bound to C identifiers (#defines). The values are defined in a **Value** column, while the C identifier, to which the value is bound, is named in a **C-Macro** column. Normally, a **Comment** column is appended in order to explain the individual values. See section 7.2 below for an example.

7.1.3 History

The "history" subsection contains a brief changelog for the table. For more information, see the "History" section.

7.2 Example

The example below shows the basic element definitions for the Session Management "SDU Error Ratio" parameter.

Description:

Reference: [3G 24.008, 10.5.6.5] (Quality of service, R99)

Definition:

long name	short name	bit len
SDU error ratio	sdu_err_ratio	4

Values:

value	c-macro	Comment
0	QOS_SDU_ERR_SUB	Subscribed SDU error ratio
0	QOS_SDU_ERR_RES_DL	Reserved (downlink only)
1	QOS_SDU_ERR_1E_2	$1*10^{-2}$
2	QOS_SDU_ERR_7E_3	$7*10^{-3}$
3	QOS_SDU_ERR_1E_3	$1*10^{-3}$
4	QOS_SDU_ERR_1E_4	$1*10^{-4}$
5	QOS_SDU_ERR_1E_5	$1*10^{-5}$
6	QOS_SDU_ERR_1E_6	$1*10^{-6}$
7	QOS_SDU_ERR_1E_1	$1*10^{-1}$
15	QOS_SDU_ERR_RES	Reserved

History:

01-Jan-2001	DEV	Initial

8 Common Sections/Table Columns

This section describes the subsections and table columns that are used in the Air Interface Message documents.

8.1 Subsections Described

8.1.1 Description

The "description" subsection is for describing the object in the section. It is entirely informational, i.e. it is not used by the tool chain. Therefore, it is not mandatory in any section, but it is strongly recommended that a description is included in any section.

The "description" subsection caption **must** be of Word type "Description" and all descriptive text **must** be of Word type "DescriptionText".

8.1.2 Pragma

The "pragma" subsection is used for modifying the behaviour of the Condat tool chain. It must contain a table with the same format as a constant definition table (section 3.1.2).

Currently only two pragmas are supported. The pragma **PREFIX** allows all constants, elements and types generated from the SAP document to be automatically prefixed with a letter combination. The pragma **COMPATIBILITY_DEFINES** makes the tool chain generate C pre-processor directives, redefining legacy style declarations to the current standard.

8.1.3 Definition

The "definition" subsection names an object, and defines a number of high level attributes for the object. The attributes defined depend on the object type. The "definition" may be used in the "Constants", "Types", "Messages", "Structured Elements", and "Basic Elements" sections.

The definition subsection caption **must** be of Word type "Definition" and all history entries **must** be of Word type "DefinitionTabelle".

8.1.4 Elements

The "elements" subsection consists of a subsection caption and a table. The table contains element definitions, and how these values are bound to C identifiers (C struct or type definitions).

The elements subsection caption **must** be of Word type "Elements" and all history entries **must** be of Word type "ElementsTabelle".

8.1.5 Values

The "values" subsection consists of a subsection caption and a table. The table contains value definitions, and how these values are bound to C identifiers (#defines).

The values subsection caption **must** be of Word type "Values" and all history entries **must** be of Word type "ValuesTabelle".

8.1.6 History

The "history" subsection contains a manually updated list of changes. It is mandatory in the "Constants" and "Types" sections as well as in all subsections of the remaining sections (i.e. subsections of "Messages", "Structured Elements", and "Basic Elements"). The developer working on the document is responsible for updating the history list.

The history subsection caption **must** be of Word type "History" and all history entries **must** be of Word type "HistoryText".

8.1.7 Combination matrix

Table 1 shows which subsections may be present in which sections. Note that the description and pragma sections are not mandatory; all other sections are mandatory.

	Description	Pragma	Definition	Elements	Values	History
Constants	●	●	●	○	○	●
Types	●	○	●	●	○	●
Messages	●	○	●	●	○	●
Structured Elements	●	○	●	●	○	●
Basic Elements	●	○	●	○	●	●

Table 1 – Subsections (keywords) allowed in Air Interface Message sections

Legend:

●	Allowed
○	Not allowed

8.2 Table Columns

The subsections below lists all table columns allowed in Air Interface Documents. Some columns are allowed in all tables, while others are restricted to one or more specific tables. Such restrictions are mentioned in the respective subsections below.

Worth noting is that table column names are not case sensitive, so “ShoRt NaME” will work the same as “short name”. Also, the order of columns is insignificant, so for example a “comment” column may be inserted at any position.

8.2.1 Short name

The **short name** column defines the C identifier with which this object is associated. In other words, the name in the **short name** column will be the name of the object in the C header file. There can be no more than one **short name** column.

Spare elements may be declared using the special “.<binary>” notation. For example, a 4-bit zero-filled spare element would be declared by “.0000”. A 7-bit spare element filled with 0x2b would be declared as “.0101011”. Spare elements are automatically inserted/removed by the Condat tools, and therefore are not included in C structs.

8.2.2 Long name

The **long name** column is for human readability only. It is not visible in the C code/header files. It is included in the test application for debugging purposes, though. There can be no more than one long name column.

8.2.3 Comment

Is a comment, as the name says. It is not included anywhere but the Air Interface Message document. Use it for descriptive purposes, or to hide columns with other content (the tools will skip the column when it is a **comment**). There may be any number of comment columns, and it may be in any table.

8.2.4 Version

The **version** column provides a way to conditionally include definitions, sections, elements etc. The **version** column contains a Boolean expression, which must evaluate to true in order for the element to be included. If the expression is false, the object is not included in the output C code, so all identifiers will remain undefined.

In the Boolean expression, an identifier that has a value (i.e. has a value other than the empty string) evaluates to true, while an undefined identifier evaluates to false. Between identifiers, Boolean operators may be used to refine the expression. See below for a list of operators allowed.

If, for example, a **version** column for an air interface message (in section 4 by convention) says “R99”, the message will be included if and only if the identifier R99 is defined. A version column which says “R99 && ! UMTS” will be included if and only if the identifier R99 is defined, and the identifier UMTS is not defined.

The following Boolean expressions are allowed in the version column:

!	Logical not. True if the identifier is not defined
|	Logical or. True if either the expression on the left or right is true.

&	Logical and. True only if both expressions on the left and right are true.
( )	Grouping operators. Changes operator precedence.

Note: The **version** column syntax is not yet finalized, so the table above should serve as information on what is possible. If the syntax differs when finalized, a later version of this document should reflect it.

8.2.5 Reference

The **reference** column (typically shortened to "**ref**") is used for linking from a table to an element. The **ref** column may appear once in message element tables and structured elements element tables. In place of a **ref** column, one may use a "**Link**" column if the element is defined in an external document.

8.2.6 Spec ref

The **spec ref** column is informational only. It allows the developer to tell where the object in question was defined in external specifications (such as ETSI/3GPP). It is allowed to appear once in message, element, and value tables.

8.2.7 Link

The **Link** column allows inclusion of elements defined in other documents. It contains a hyperlink to a section in another air interface/SAP document, from which the Condat tool chain fetches the definitions.

A **Link** columns is mutually exclusive with a **Reference** column. They serve the same purpose, only **Link** columns refer to information outside the current document.

Condat convensions say that all linked elements in a section should go in the same table. That is, the **Link** column should be placed in one message definition, structured elements definition, and basic elements definition tables, and may contain a number of elements.

8.2.8 Ctrl

The **ctrl** column is the most complex of all columns. It contains instruction for the Condat tool chain on how to define or interpret the object in question. Section 9.1 contains the formal definition of the possible contents in the **Ctrl** column.

Arrays are defined through the "[<from> .. <to>]" or "[<number>]" notation. The latter defines a constant length array with <number> elements, while the former defines a variable length array with anything from <from> to <to> elements, inclusive.

Note: Arrays of unions are not supported. Unions must be encapsulated in a structure in order to create an array. The structure requirement is due to the extra union controller ("ctrl_") element inserted by the Condat tool chain; This element is outside the union, and thus needs a structure to contain it.

Conditional inclusion of elements is possible by using the "{<variable> <operator> <number/constant>}" syntax. The condition is checked at runtime, so for example "{ TI = 7 }" in an element's **ctrl** column would include the element only if the TI (transaction identifier) has the value 7. <variable> may be either a defined constant or an element defined earlier in the message/structured element. Operator may be "=", "#", "<", "<=", ">", and ">=".

Calculations are possible by using the calculator built into Condat's runtime tools. The calculator uses reverse polish notation, i.e. it is a stack machine (not unlike the venerable HP48). With it, one can write quite complex expressions, and move the bit-pointer back and forth in the message. Please refer to the **ctrl** column syntax description in section 9.1 for a list of possible operations.

Dynamic arrays (or pointer types) may be declared using the keywords "ptr" or "dyn".

8.2.9 ID

The **ID** column is allowed in air interface message definition and element tables only. In message definition tables, **ID** defines the air interface message ID; In message element tables, **ID** defines the IEI value for the (optional) element is included in the air interface message.

Message definition example: An air interface message has an **ID** of 66 (session management ACTIVATE PDP CONTEXT ACCEPT). This tells the Condat tool chain that the elements in the table below are associated with the **ID** 66. For an incoming message (downlink) with **ID** 66, the Condat tools will unpack data into the fields/variables defined in the elements below. For outgoing messages (uplink) with **ID** 66, a message will be formed using **ID** and the value from the elements below.

Message element example: A type 4 TLV element may have an IEI of 0x2b. This tells the Condat tools chain that this element is included in its parent message only if the IEI 0x2b appears after the mandatory elements. For an incoming message, this means that a valid flag is set to true for this element by the Condat tools, indicating to your code, that the element is present. For an outgoing message, your code need only set the valid flag to true for the element, and the Condat tools will include the IEI and the element in the air interface message.

8.2.10 Direction

The **direction** column is allowed in message definition tables only. Its value may be *uplink*, *download*, and *both*. Apart from its informational value, the tells the Condat tools whether to create insert the message in the list of allowed uplink and/or downlink messages.

If a message is defined as uplink, the tools will allow sending the message, but report an error if received. Note that the tools will not recognize the message, if it has the wrong **direction**.

8.2.11 Pres

The **pres** column (short for “presence”) indicates whether elements are mandatory or optional. It is purely informational, i.e. it is not used by the Condat tool chain (it checks the type in stead). The **pres** column may contain the values *optional* or *mandatory* (or any shorthand hereof, including just “O” and “M”). It may appear in message element and structured element tables only, and only once.

8.2.12 Type

The **type** column defines the coding type for the element in question. The **type** column is allowed in air interface message element tables and structured element tables only.

8.2.13 Length

The **length** column (which may be shortened to **len**) defines the length in bytes for the object in question. It is mandatory in basic element definition rows, whereas it is optional (and informational, i.e. not used) in message element table rows, structured element rows, structured element definition rows. In most places, the **len** column may be replaced by a **Bit len** column.

For variable length IEs, the **length** may be expressed as a range, i.e. <min> - <max>, where <min> and <max> may be either a number or a constant.

8.2.14 Bit len

The **bit len** column defines the length in bits for the object in question. It may be used in message element table rows, structured element rows, structured element definition rows, and basic element definition rows, where it may replace the **Len** column.

8.2.15 Value

The **value** column binds individual values to C identifiers when paired with a **C-Macro** column. It is allowed in value tables only.

The **value** parameter defines the value bound to the C identifier named in the **C-Macro** column. The **value** parameter may be entered in decimal, hexadecimal (e.g. 0x0a), or binary (e.g. 0b00001010) notation.

8.2.16 C-Macro

The **C-Macro** column binds individual values to C identifiers when paired with a “Value” column. It is allowed in value tables only.

The **C-Macro** parameter is a C identifier as it may be used in program code later. Be careful concerning name clashes!

8.3 Table Column/Section Combination Matrix

Table 2 below lists the Air Interface Message document sections, and which columns are allowed in the respective sections table.

	Active Section / Table							
	Con-stants	Types	Messages		Structured elements		Basic elements	
	Definition	Definition	Definition	Element	Definition	Element	Definition	Value
name	●	○	○	○	○	○	○	○
long name	○	●	●	●	●	●	●	●
short name	○	●	●	●	●	●	●	○
comment	●	●	●	●	●	●	●	●
version	○	○	●	●	●	●	●	●
reference / ref	○	○	○	●	○	●	○	●
spec ref	○	○	○	●	○	●	○	●
link	○	○	●	○	●	○	●	○
ctrl	○	⦿	○	⦿	○	⦿	○	⦿
ID	○	○	●	●	○	○	○	○
direction	○	○	●	○	○	○	○	○
pres / presence	○	○	○	●	○	●	○	○
type	○	○	○	●	●	○	○	○
length / len	○	○	○	⦿	⦿	⦿	⦿	○
bit len	○	○	○	⦿	⦿	⦿	⦿	○
add bit	○	●	○	○	○	○	○	○
value	○	○	○	○	○	○	○	●
c-macro	○	○	○	○	○	○	○	●

Table 2 - List of Columns Allowed in Different AIM Tables

Legend:

●	Allowed
⦿	Allowed (with restrictions)
○	Not allowed

9 Formal Grammar

At some point in time, this section may end up containing a formal grammar of what may be contained in an Air Interface Message document.

So far, only the Ctrl column is specified formally.

9.1 Ctrl column

```
CtrlColumnContents:
        CtrlInfos

CtrlInfos:
        CtrlInfo
        CtrlInfos  CtrlInfo

CtrlInfo:
        CtrlGroup
        CtrlCondition
        CtrlArray
        CtrlPreamble
        CtrlLength
        CtrlType

CtrlGroup:
        "+"                     /* Signals start of multi-octet GSM extended group:  Bit 7 set = present */
        "-"                     /* Signals end of multi-octet GSM extended group */
        "*"                     /* Signals single octet GSM extended group:  Bit 7 set = present */

CtrlCondition:
        "{"  ConditionalExpression  "}"

CtrlArray:
        DynamicOpt?  "["  ArrayInfo  "]"

CtrlPreamble:
        "("  PreambleInfo  ")"

CtrlLength:
        Digits

CtrlType:
        <any word>

ConditionalExpression:
        OperatorExpression
        OperatorExpression  BindingOperator ConditionalExpression

OperatorExpression:
        Identifier  ComparisonExpression  Digits
        Identifier  ComparisonExpression  Constant
        Identifier  ComparisonExpression  Identifier

ComparisonOperator:
        "="                     /* Comparison for equality */
        "#"                     /* Comparison for inequality */
        "<"                     /* Less than comparison */
        ">"                     /* Greater than comparison */

BindingOperator:
        "A"                     /* Logical AND */
        "O"                     /* Logical OR */
        "X"                     /* Logical XOR (exclusive or) */

PreambleInfo:
        RPExpression
        RPExpression "," PreambleInfo

RPExpression:                   /* Expressions in reverse polish notation */
        Digits                  /* Constant to push on stack */
        Identifier        /* Push value of identifier on to stack */
        StackOperator
        NumericOperator
```

```
StackOperator:
        "SetPos"            /* Push bit-pointer (in message) on stack */
        "GetPos"            /* Pop bit-pointer (in message) from stack */
        ":"                 /* Duplicate top element in stack */
        "^"                 /* Swap upper two elements in stack */

NumericOperator:
        "+"                 /* Adds upper two elements on stack and leaves result */
        "-"                 /* Subtracts upper two elements on stack and leaves result */
        "*"                 /* Multiplies upper two elements on stack and leaves result */
        "/"                 /* Divides upper two elements on stack and leaves result */
        "&"                 /* Performs logical "and" between upper two elements and leaves result */
        "|"                 /* Performs logical "or" on upper two elements and leaves result */

DynamicOpt:
        "DYN"               /* Dynamic array specifier – code transparent */
        "PTR"               /* Dynamic array specifier – non-code transparent */

ArrayInfo:
        "."Min ".." Max     /* Variable size – length specified in bits */
        Min ".." Max        /* Variable size – length specified in bytes */
        "."Size             /* Constant size – length specified in bits */
        Size                /* Constant size – length specified in bytes */

Min, Max, Size:
        Identifier
        Digits

Digits:
        Digit
        Digits Digit

Digit:
        "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
```

10 Example

[FIXME! Refer to some ClearCase based AIM document.]

Appendices

A. Acronyms

DS-WCDMA	Direct Sequence/Spread Wideband Code Division Multiple Access

B. Glossary

International Mobile Telecommunication 2000 (IMT-2000/ITU-2000)	Formerly referred to as FPLMTS (Future Public Land-Mobile Telephone System), this is the ITU's specification/family of standards for 3G. This initiative provides a global infrastructure through both satellite and terrestrial systems, for fixed and mobile phone users. The family of standards is a framework comprising a mix/blend of systems providing global roaming. <URL: http://www.imt-2000.org/>