

# DIAMETER Protocol Module Generator for TTCN-3 Toolset with TITAN, Function Description

Tímea Moder

Version 1551-CNL 113 462, Rev. AG, 2018-06-01

# Table of Contents

About This Document .....	1
How to Read This Document .....	1
Presumed Knowledge .....	1
General Overview .....	1
Background .....	1
Overview .....	1
Functionality .....	2
Naming Conventions .....	2
Key to Unique Naming of Identifiers .....	3
How to Model Enumeration Type AVPs in DDFs? .....	4
Script Operation .....	5
Load and Parse All Input Files .....	5
Type Identifiers .....	5
Vendor_Id .....	5
AVP_Code_<Official-Vendor-Id> .....	6
AVP_Code .....	6
<b>AVP_Header</b> .....	7
AVP_Data .....	7
AVP .....	7
GenericAVP .....	8
Command_Code .....	8
Output TTCN-3 Module .....	8
Output Encoder/Decoder .....	8
Detailed VMP and RPET Bits .....	9
Bigint Support for Unsigned32 and 64 Bit Integer .....	9
Backward Incompatibilities .....	9
<i>CxDxInterface_Ericsson_1551_FAY301_0059_PC26.ddf</i> .....	9
System Requirements .....	9
Installation .....	10
Generation of the <i>_5DIAMETER_Types.ttcn_5</i> .....	10
Supported Options .....	10
<i>Makefile</i> Preparation .....	11
Helper Functions .....	12
Encoding/Decoding Functions .....	13
Error Handling .....	13
Limitations .....	14
Protocol Versions .....	14
Product Contents and Structure .....	14
Protocol Version Implemented .....	14
Modifications/deviations related to the protocol specification .....	19

Unimplemented Messages, Information Elements and Constants .....	19
Protocol Modifications/Deviations .....	19
Upgrading Templates Used by the DIAMETER Test Port .....	22
Examples .....	24
Mapping module .....	24
Client Mode .....	24
Server Mode .....	26
ASPs of the DIAMETERmsg_PT port .....	27
Demo Module .....	28
Test Cases .....	28
Configuration Files .....	28
Examples for Building the Project .....	29
Script to Modify <i>Makefile</i> .....	29
Abbreviations .....	30
Terminology .....	30
References .....	30

# About This Document

## How to Read This Document

This is the Function Description for the DIAMETER Protocol Module Generator. The DIAMETER Protocol Module Generator is developed for the TTCN-3 Toolset with TITAN.

## Presumed Knowledge

To use this protocol module the knowledge of the TTCN-3 language [1] is essential.

Basic knowledge of the Diameter protocol [3] is valuable to use this protocol module.

# General Overview

## Background

Former solution for testing Diameter applications is based on the Diameter Test Port. This test port implies a number of limitations:

1. AVPs and other application-specific data are hard-coded in the Test Port, which makes the extension hard. New AVPs need to be added, encoded and decoded manually.
2. The support of different revisions of same Diameter application is required by different projects. Different revisions may contain, for example, overlapping AVP codes or other contradictory type definitions, which can only be handled using run-time switches.

DPMG provides solution to this problem by dynamically generating the type definition module containing the AVPs and definitions of the chosen applications.

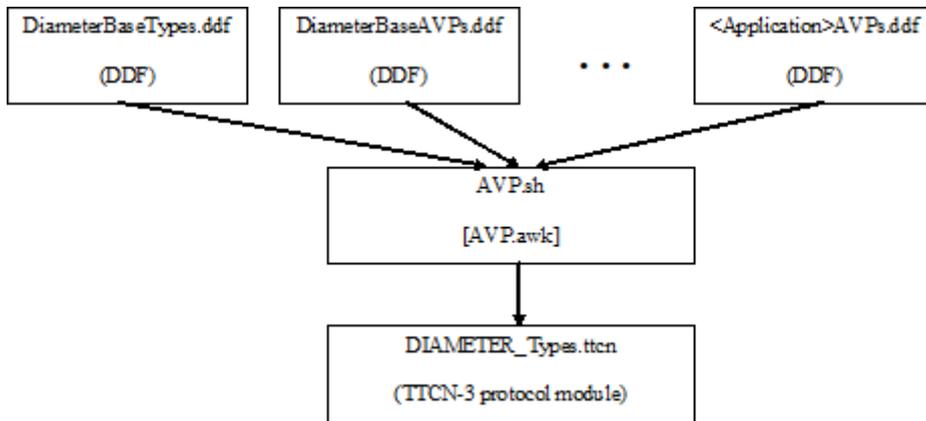
The generated protocol module implements the message structures of the Diameter protocol in a formalized way, using the standard specification language TTCN-3. This allows defining of test data (templates) in the TTCN-3 language [1] and correctly encoding/decoding messages when executing test suites using the Titan TTCN-3 test environment.

The protocol module uses either the Titan's RAW encoding attributes [2] for Diameter encoder or the generated speed optimized C++ encoder and hence is usable with the Titan test toolset only.

## Overview

Protocol modules implement the message structure of the related protocol in a formalized way, using the standard specification language TTCN-3. This allows defining of test data (templates) in the TTCN-3 language [1] and correctly encoding/decoding messages when executing test suites using the Titan TTCN-3 test environment.

See the DPMG Architecture below:



# Functionality

The DIAMETER protocol module (for example, *DIAMETER\_Types.ttcn*) is generated dynamically from the input DDF files using a shell script, which performs this with the help of an AWK script ([DPMG Architecture](#)).

The naming of DDFs should follow the `<Official-Vendor-Id><Official-Application-Name>AVPs_<Application-Version>` scheme. The extension MUST NOT be *.ttcn*; *.ddf* is recommended.

If the application uses only a minor sub-set of some standard specification then it is acceptable to define these AVPs in the framework of the new application instead of including all unnecessary AVPs of the standard.

The type definitions for testing protocols comply the Diameter Base Specification are split in two DDFs:

DDF name	Contents
<code>BaseTypes_IETF_RFC3588</code>	Diameter Base Protocol [3] related type definitions
<code>Base_IETF_RFC3588</code>	Diameter Base Protocol [3] AVP type definitions

The *AVP.awk* script merges its argument DDFs into a single valid TTCN-3 module called *DIAMETER\_Types* by default.

The input DDFs must comply with the naming and typographical conventions described herein in order for the *AVP.sh* script to produce a syntactically and semantically valid TTCN-3 module.

# Naming Conventions

The generated identifiers of types are based on information provided in comments within the DDFs themselves.

1. Applications are distinguished using the unique `<Application-Name>` and `<Application-Revision>`, which are assigned by TCC. The `<Application-Name>` is used to prefix type as well as certain field identifiers in the generated TTCN-3 module to ensure unique naming. The `<Application-Revision>` is only optionally used in prefixes. The `<Application-Name>` and `<Application-Revision>`

are hard-coded in each TTCN-3 FILE using the following format:

```
// APPLICATION-NAME: NASREQ  
  
// APPLICATION-REVISION: Draft17
```

2. AVP properties (e.g. name, code, vendor-id) are enlisted in C++ style comment right before the AVP type definition using the following format:

```
// AVP: <Official-AVP-Name> (<Official-AVP-Code>) <Official-Vendor-Id> (<Official-  
Vendor-Id-Code>)  
  
type Type_Specifier Official_AVP_Name ...
```

3. The entire comment line with the information MUST stand in the same line (no line breaks when it splits to multiple lines).
4. The TTCN-3 type definition following a properly formed comment line is interpreted as an AVP definition, if the type identifier matches the **<Official-AVP-Code>** appearing in the comment before.
5. The **<Official-AVP-Name>**, **<Official-AVP-Code>**, **<Official-Vendor-Id>** and **<Official-Vendor-Id-Code>** shall come from the relevant RFC, IETF Draft or other specification.
6. The **<Official-Vendor-Id>** and **<Official-Vendor-Id-Code>** must be omitted if V bit is not set (i.e. **<Official-AVP-Name>** and **<Official-AVP-Code>** are unique)!
7. The TTCN-3 identifiers used in **<Official-AVP-Name>** and **<Official-Vendor-Id>** must keep the original naming except when this collides with TTCN-3 identifier's naming rules:
  - a. Hyphens and spaces must be replaced by a single underscore
  - b. Trailing **AVP** SHOULD be omitted if not part of the name
8. The **<Official-AVP-Code>** and **<Official-Vendor-Id-Code>** must be given as integer numbers!

## Key to Unique Naming of Identifiers

The following uniqueness criteria – derived from Diameter [3] – must hold for identifiers used in DDFs:

1. **<Application-Name>** or **<Application-Revision>** MAY NOT be globally unique.
2. **<Application-Name>** AND **<Application-Revision>** MUST be globally unique:  
Each application identifier must be formulated so that it is always unique. It can happen that different drafts of the same application are used together that is why it is strongly recommended to prefix with **<Application-Revision>**, too!

Example 1

```
<Application-Name>s: BASE, NASREQ
```

#### Example 2

```
<Application-Revision>s: RFC3588, Draft17  
Combined prefixes: BASE_RFC3588,  
NASREQ_Draft17
```

### 3. **<Official-AVP-Name>** may not be unique:

It happens that the same AVP name is used in the same or in different Diameter applications. The script is designed to cope with this, thus it is recommended to keep the standard AVP name with respect to naming conventions.

#### Example

```
<Official-AVP-Name>s: Multi_Round_Time_Out
```

4. **<Official-AVP-Code>** AND **<Official-Vendor-Id>** MUST be globally unique (except within different revisions of the same application!) since these two 32Bits numbers determine the AVP.
5. **<Official-AVP-Name>** AND **<Official-Vendor-Id>** MUST be unique within an application
6. **<Application-Name>** AND **<Official-AVP-Name>** AND **<Official-Vendor-Id>** MUST be globally unique

## How to Model Enumeration Type AVPs in DDFs?

It is important to ensure the unique naming of enumeration type identifiers and enumeration items. Each enumerated type AVP requires a single type definition: The enumerated type definition containing the valid enumeration items. The identifier of the enumeration type shall be **<Official-AVP-Name>**.

The `AVP.sh` script generates Unsigned32 type AVP for each enumerated type AVP when the `enum_2_Unsigned32` option is turned on.

All enumerations in DDFs will get the following attributes automatically assigned to enumeration type AVPs' enumerations:

```
with {  
  variant "FIELDLENGTH(32)"  
  variant "BYTEORDER(last)"  
  variant "COMP(2scompl)"  
}
```

**Command\_Code** enumeration type can be extended in Diameter applications. DPMG merges them together into a single type definition with proper attributes. Duplicates are removed when some enumeration items appear multiple times within **Command\_Code** definitions of the input DDF files.

# Script Operation

The TTCN-3 module, containing all relevant type definitions, is generated automatically from the relevant DDFs by a script. This ensures that no collision can appear between proper Diameter applications.

The top-level Diameter PDU to send/receive is always `PDU_DIAMETER`.

## Load and Parse All Input Files

If overlapping AVP codes (same AVP code and Vendor-Id) are found during parsing of DDFs then the created TTCN-3 module (for example, `DIAMETER_Types`) will contain only the AVP found first. (This can happen when trying to use many different or identical revisions of the same Diameter application.)

## Type Identifiers

The script changes AVP type identifiers in order to avoid name collisions. The `<Official-Application-Name>` (and optionally the `<Official-Application-Revision>`) and `<Official-Vendor-Id>` will prefix the `Official_AVP_Name` defined in DDF.

Example of AVP type definition in DDF:

```
// RFC 3588 8.14
// AVP: User-Name (1)
type AVP_UTF8String User_Name;
```

The corresponding type definition in the generated module (no Vendor-Id is allowed for User-Name AVP of Diameter Base specification (`<Application-Name>=BASE`)):

```
// AVP: User-Name (1)
type AVP_UTF8String BASE_NONE_User_Name;
```

If the `<Official-AVP-Name>` begins with `<Official-Vendor-Id>` then it is recommended to remove this from the `<Official-AVP-Name>` as the `<Official-Vendor-Id>` is always used to prefix AVP type definitions!

When the `<Vendor-Id>` of `<Official-AVP-Name>` is in category MUST NOT, then the `<Vendor-Id>` MUST NOT appear in the AVP comment line. "NONE" is used in the identifier of the generated AVP type definition when `<Official-Vendor-Id>` is absent.

## Vendor\_Id

Create `Vendor_Id` enumerated type containing all vendor ids that were found in the comment fields. The `Vendor_Id` type shall be used to determine the valid AVP code set (`AVP_Code_<Application-Name>_<Official-Vendor-Id>`) in the `AVP_Code` union.

```

type enumerated Vendor_Id {
    // for each vendor id found in FILES
    vendor_id_<Official-Vendor-Id> (<Official-Vendor-Id-Code>)
} with {
    variant "FIELDLENGTH(32)"
    variant "BYTEORDER(last)"
    variant "COMP(2scompl)"
}

```

The `Vendor_Id` codes are assigned by IANA according to ASSIGNNO [RFC3232], which is now obsolete by an on-line database at <http://www.iana.org/assignments/enterprise-numbers>. The database contains over 23000 entries thus using a predefined `Vendor_Id` type is not appropriate!

#### NOTE

This `Vendor_Id` type will not clash with the Vendor-Id AVP of Diameter base specification as the AWK script alters the identifier of the latter type definition!

## AVP\_Code\_<Official-Vendor-Id>

Putting all AVP codes into a single enumerated type does not work because `<Official-AVP-Code>` is not globally unique. We can create unique identifiers for enumeration items but some enumeration items could have the same numeric value assigned, which is forbidden in TTCN-3. Separate `AVP_Code_<Official-Vendor-Id>` enumerations have to be created for each found Vendor-Id. The enumeration items themselves will be the AVP names prefixed with `avp_code_`, `<Application-Name>` and `<Official-Vendor-Id>`!

```

type enumerated AVP_Code_<Official-Vendor-Id> {
    avp_code_<Application-Name>_<Official-AVP-Name>
    (<Official-AVP-Code>)
} with {
    variant "FIELDLENGTH(32)"
    variant "BYTEORDER(last)"
    variant "COMP(2scompl)"
}

```

For those AVPs where the `<Official-Vendor-Id>` MUST NOT be present, "NONE" shall be used as prefix!

## AVP\_Code

The `AVP_Code` itself is a union type consisting of the `AVP_Code_<Official-Vendor-Id>` enumerations.

```

type union AVP_Code {
    // for each vendor id found in DDFs
    AVP_Code_<Official-Vendor-Id> vendor_id_<Official-Vendor-Id>
}

```

## AVP\_Header

The **AVP\_Header** type must be generated because the RAW attributes must be inserted for correct decoding of **AVP\_Code** union.

```
type record AVP_Header {
    AVP_Code    avp_code,
    BIT8        VMPxxxxx,
    UINT24      avp_length,
    Vendor_Id   vendor_id optional
} with {
    variant (vendor_id) "PRESENCE( {
        VMPxxxxx = '10000000'B,
        VMPxxxxx = '10100000'B,
        VMPxxxxx = '11000000'B,
        VMPxxxxx = '11100000'B
    } )"
    variant (avp_code) "CROSSTAG(
        // for all AVP_Code union members
        vendor_id_<Official-Vendor-Id>,
        vendor_id = vendor_id_<Official-Vendor-Id>;
    )"
}
```

For proper decoding it is important to set the spare bits to zero as required by the Diameter base specification [3].

## AVP\_Data

The **AVP\_Data** type is a generated union type containing all AVP types found in the DDFs:

```
type union AVP_Data {
    <Application-Name>_<Official-Vendor-Id>_<Official-AVP-Name>
    avp_<Application-Name>_<Official-Vendor-Id>_<Official-AVP-Name>,
    octetstring avp_UNKNOWN
}
```

**avp\_UNKNOWN** contains the erroneous AVP when something went wrong during the decoding of the AVP data.

## AVP

The AVP type is a record that consists of two fields: the header **avp\_header** and the data **avp\_data**.

```

type record AVP {
    AVP_Header avp_header,
    AVP_Data avp_data
} with {
    variant "PADDING(dword32)"
    variant (avp_header) "LENGHTO(avp_header, avp_data)"
    variant (avp_header) "LENGTHINDEX(avp_length)"
    variant (avp_data) "CROSSTAG(
        // for all union fields
        avp_<Application-Name>_<Official-Vendor-Id>_<Official-AVP-Name>,
        avp_header.avp_code.vendor_id_<Official-Vendor-Id> =
        avp_code_<Application-Name>_<Official-AVP-Name>;
        // last entry
        avp_UNKNOWN, OTHERWISE
    )"
}

```

## GenericAVP

The `GenericAVP` type is a union that was defined for error handling purposes.

```

type union GenericAVP {
    AVP avp,
    octetstring avp_UNKNOWN
}

```

The `avp` field contains an AVP if it was correctly decoded, while the `avp_UNKNOWN` will contain the erroneous AVP with the header when something went wrong during the decoding.

## Command\_Code

`Command_Code` enumeration type is merged together from the DDF file of different application's `Command_Code` definition by the AWK script. All enumeration item defined in different application are collected together and written to the generated `DIAMETER_Types.ttcn` file. Proper encoding attributes are added to the `Command_Code` type by the script.

## Output TTCN-3 Module

All definitions of DDF files, which are not subject to change are written to the output TTCN-3 module (for example, `DIAMETER_Types.ttcn`) file as is.

## Output Encoder/Decoder

Optionally it is possible to generate a speed optimized `DIAMETER_EncDec.cc` encoder/decoder instead of the RAW encoder and the default `DIAMETER_EncDec.cc`.

## Detailed VMP and RPET Bits

The type definition of the RPET bits of the `Diameter` header and the VMP bits of the `AVP` header can be:

- 8 bit wide bitfield (BIT8) (traditional representation)
- Every bit is represented as a single bit (BIT1)

The handlings of these bits are controlled by the parameter of the generator script.

## Bigint Support for Unsigned32 and 64 Bit Integer

The 32 bit unsigned integer and 64bit integer types can be represented as:

- 4 or 8 octet long octetstring
- integer

The used type is controlled by the parameter of the generator script.

## Backward Incompatibilities

### *CxDxInterface\_Ericsson\_1551\_FAY301\_0059\_PC26.ddf*

Until version R24B The ddf file contained duplicated AVPs with *Ericsson\_Specific\_AVPs.ddf*. In version R24C, these duplications were removed and the prefix of the AVPs was chaged from `ECX_` to `E_`. This change is not backward compatible.

## System Requirements

The `DIAMETER` protocol module generator consist of several DDF files, contains different application definitions of Diameter protocols, a `GNU AWK` and shell script which reads the DDF files and generates the type definition module (*DIAMETER\_Types.ttcn* by default).

Protocol modules are a set of TTCN-3 source code files that can be used as part of TTCN-3 test suites only. Hence, protocol modules alone do not put specific requirements on the system used. However in order to compile and execute a TTCN-3 test suite using the set of protocol modules the following system requirements must be satisfied:

- TITAN TTCN-3 Test Executor R7A (1.7.pl0) or higher installed. For installation guide see [\[2\]](#).

#### NOTE

This version of the protocol module is not compatible with TITAN releases earlier than R7A.

# Installation

The set of protocol modules can be used for developing TTCN-3 test suites using any text editor. However, to make the work more efficient a TTCN-3-enabled text editor is recommended (for example, `nedit`, `xemacs`). Since the Diameter protocol is used as a part of a TTCN-3 test suite, this requires Titan TTCN-3 Test Executor be installed before the module can be compiled and executed together with other parts of the test suite. For more details on the installation of TTCN-3 Test Executor see the relevant section of [2].

The `AVP.sh` shell script runs on Bourne Shell, which is usually available on a all UNIX like workstations. The `AVP.awk` script, which processes the DDF files and creates the DIAMETER protocol module, can be executed with GNU AWK version 3.1.6 or later [4] so it must be available on the system.

## Generation of the `_5DIAMETER_Types.ttcn_5`

First you need to obtain the required DDF files. After you have the DDF files containing the definitions of the selected Diameter applications, you can generate the proper Diameter type definitions module by issuing for example the following command:

```
AVP.sh DiameterBaseAVPs.ddf DiameterBaseTypes.ddf OtherApplications.ddf
```

The above command generates the TTCN-3 type definition by merging the content of DDF files into module `DIAMETER_Types` into file `DIAMETER_Types.ttcn`. The script filters out duplicate AVP definitions by placing only the first one into the generated TTCN-3 module. Skipped definitions are annotated with warnings.

The script can be optionally invoked with some options. The options modify script operation. The options must appear in the argument list before the DDF files. Each option is introduced with the `-v` flag. The options must not immediately follow the `-v` flag; whitespace separation is required. The options themselves must be written in one word.

## Supported Options

The supported options are:

- `module_id`:  
This option can be used to alter the identifier of generated TTCN-3 module. The `module_id` shall be a valid TTCN-3 identifier but must not contain underscore.
- `use_application_revision`:  
This option results the application revision string to be added as prefix to generated identifiers. The application revision string is hardcoded into the input DDF files. This option is disabled by default.

`use_bigint`:

This option makes the script to use integer for UINT32, INT64 and UINT64 types instead of octetstring. This option is disabled by default.

### **detailed\_bits:**

This option makes the script to generate detailed type definition for VMP and RPET bits. This option is disabled by default.

### **disable\_prefix:**

This option makes the script to generate identifier names and a fixed "AVP" prefix instead of using the application name specified in the ddf file. This option is disabled by default.

### **enum\_2\_Unsigned32\_list=<list file name>:**

This option makes the script to generate the listed AVP as Unsigned32 instead of enumerated type.

### **custom\_enc:**

This option makes the script to generate c++ encoder/decoder function instead of using the RAW encoder. This function generates a complete new *DIAMETER\_EncDec.cc*.

### **use\_UTF8\_encoding:**

This option defines AVP\_UTF8String as universal charstring which will be encoded in UTF-8. This option can be used only together with **custom\_enc**.

The next command stores the generated TTCN-3 definitions in module XYZ and translates all enumeration type AVPs to Unsigned32:

```
AVP.sh -v module_id=XYZ -v enum_2_Unsigned32=true DiameterBaseAVPs.ddf
DiameterBaseTypes.ddf _OtherApplications.ddf_
```

#### **NOTE**

**AVP.sh** requires **AVP.awk** and **-** in case of specifying the **module\_id** option – the *DIAMETER\_EncDec.cc* C++ source file for its operation!

The next command stores the generated TTCN-3 definitions in module XYZ, generates a *DIAMETER\_EncDec.cc* and **AVP\_UTF8String** will be defined as universal charstring encoded as UTF-8.

```
AVP.sh -v module_id=XYZ -v custom_enc=DIAMETER_EncDec.cc -v use_UTF8_encoding=true
DiameterBaseAVPs.ddf DiameterBaseTypes.ddf _OtherApplications.ddf_
```

## **Makefile Preparation**

In case you want to add the task of generation of *DIAMETER\_Types.ttcn* module into your *Makefile*, you should do the following:

1. Generate the *GNU Makefile* for your existing TTCN-3 and C++ files, except *DIAMETER\_Types.ttcn*.
2. Add *DIAMETER\_Types.ttcn* manually to the list of the TTCN-3 sources into the generated *Makefile*.
3. Add the following rules to your *Makefile*:

```
DIAMETER_Types.ttcn: DiameterBaseTypes.ddf DiameterBaseAVPs.ddf <Input FILEs
containing AVP definitions>
```

```
AVP.sh latexmath:[$(filter %.ddf,$)^)
```

When you use GUI for building executable, on top of normal project creation you should take care of the following:

1. Add **AVP.awk** script and all DDF files you need to add to the **misc** files section.

**NOTE**

You must generate softlinks to the build directory with selecting the files and select 'Generate Softlinks' menu item manually as it is not generated automatically by the GUI.

1. Create a script to modify the generated *Makefile*. List all the DDF files you need when executing the **AWK** script.
2. Add script to modify the *Makefile* in the project properties.
3. If you want to have the *DIAMETER\_Types.ttcn* file added to your project, you can, but after adding, you should exclude the file from build in order to avoid double occurrence in the *Makefile*.

## Helper Functions

Two separate external functions are available for generating End-to-End and a Hop-by-Hop identifiers:

```
external function f_DIAMETER_genHopByHop() return octetstring;
```

This function generates a 4 octet long Hop-by-Hop identifier. The values returned are based on random number generation.

```
external function f_DIAMETER_genEndToEnd() return octetstring;
```

The function above generates a 4 octet long End-to-End identifier according to [5]. The high order 12 bits contain the low order 12 bits of current time, and the low order 20 bits contain a random value.

It is important to mention, that if the Hop-by-Hop-Identifier or the End-to-End-Identifier is set to "0", the encoder function automatically generates a value with the help of the presented two functions.

One external function is available to acquire an AVP by AVP code from an encoded Diameter PDU.

```
external function f_DIAMETER_GetAVPByListOfCodes(in octetstring pl_oct, in integerList
pl_codeList) return octetstring;
```

The function accepts a list of AVP codes and will return the octetstring AVP value of the first AVP in the encoded Diameter PDU whose AVP code is in the list. Providing multiple AVP codes can be useful if the same AVP type can appear in the message with different AVP codes (for example, public id).

```
external function f_DIAMETER_GetAVPByListOfCodesCombined(in octetstring pl_oct, in
integerList pl_codeList, in integerList pl_groupcodeList) return octetstring;
```

The purpose of this function is the same as the `f_DIAMETER_GetAVPByListOfCodes`, except `f_DIAMETER_GetAVPByListOfCodesCombined` searches also within the grouped AVPs listed in the `pl_groupcodeList` list.

## Encoding/Decoding Functions

This product also contains encoding/decoding functions that assure correct encoding of messages when sent from Titan and correct decoding of messages when received by Titan. Implemented encoding/decoding functions:

Name	Type of formal parameters	Type of return
<code>valuef_DIAMETER_Enc</code>	(in PDU_DIAMETER pl_pdu)	octetstring;
<code>f_DIAMETER_Dec</code>	(in octetstring pl_oct)	PDU_DIAMETER;

## Error Handling

During the decoding of a Diameter message the following error scenarios can be identified:

- If a Diameter message arrives with a command code not known by the Diameter protocol module, an error message is generated, where the unknown command code value appears at `<value>`:

```
Warning: While RAW-decoding type '@DIAMETER_Types.PDU_DIAMETER': Invalid enum value
<value> for '@DIAMETER_Types.Command_Code'
```

- In case of an unknown AVP code, the AVP is decoded into a special `avp_UNKNOWN` field that contains the entire AVP with the header in its octetstring form. If the length of the AVP cannot be determined then the rest of the Diameter message is also put into this kind of AVP.
- If there is an AVP in the Diameter message, where the V bit is incorrectly set to "1", but it doesn't contain an optional Vendor-Id field, the decoder first tries to interpret the octets as a Vendor-Id and if it isn't a known Vendor-Id value, then the decoder can detect it and sets this field to omit. The octets will be treated as data further on.

- In case there is an AVP in the Diameter message, where the V bit is incorrectly set to "0", but it contains an optional Vendor-Id field, it is decoded into the special `avp_UNKNOWN` field.
- When an AVP appears with an unexpected Vendor-Id, it is decoded into the special `avp_UNKNOWN` field.

## Limitations

`<Application-Revision>` field in DDF files are not handled yet by the script. The reason is no application makes the `<Application-Revision>` info essential.

# Protocol Versions

## Product Contents and Structure

The major parts of DPMG are:

- a. `AVP.sh` script - This is the front-end of the protocol module generator.
- b. `AVP.awk` script - This is the most important part of the product.
- c. A pair of encoder and decoder functions to invoke RAW encoder/decoder or the generated encoder/decoder.

The Diameter Base Protocol [1] and other Diameter applications are specified in DDFs developed by TCC as part of the DPMG product.

The TTCN-3 module that is generated by the script varies between applications, thus it is NOT a product.

## Protocol Version Implemented

Currently the following applications are supported:

DDFs	Refs.
<i>BaseTypes_IETF_RFC3588.ddf</i> <i>Base_IETF_RFC3588.ddf</i>	[3]
<i>CLCInterface_Vodafone_Rev2.ddf</i>	[50]
<i>ChargingApplications_3GPP_TS32299_850.ddf</i>	[24]
<i>ChargingApplications_3GPP_TS32299_870.ddf</i>	[25]
<i>ChargingApplications_3GPP_TS32299_900.ddf</i>	[26]
<i>ChargingApplications_3GPP_TS32299_940.ddf</i>	[62]
<i>ChargingApplications_3GPP_TS32299_9b0.ddf</i>	[63]
<i>ChargingApplications_3GPP_TS32299_a60.ddf</i>	[63]

<b>DDFs</b>	<b>Refs.</b>
<i>ChargingApplications_3GPP_TS32299_c60.ddf</i>	[102]
<i>ChargingApplications_3GPP_TS32299_d70.ddf</i>	
<i>ChargingApplications_3GPP_TS32299_be0.ddf</i>	[104]
<i>ChargingApplications_3GPP_TS32299_d40.ddf</i>	[103]
<i>CreditControl_IETF_RFC4006.ddf</i>	[6]
<i>CxDxInterface_3GPP_TS29229_6a0.ddf</i>	[27]
<i>CxDxInterface_3GPP_TS29229_840.ddf</i>	[28]
<i>CxDxInterface_3GPP_TS29229_880.ddf</i>	[29]
<i>CxDxInterface_3GPP_TS29229_920.ddf</i>	[62]
<i>CxDxInterface_3GPP_TS29229_c30.ddf</i>	[103]
<i>DigestAuthentication_IETF_RFC5090.ddf</i>	[52]
<i>GiInterface_3GPP_TS29061_770.ddf</i>	[32]
<i>GiSGiInterface_3GPP_TS29061_810.ddf</i>	[33]
<i>GiSGiInterface_3GPP_TS29061_930.ddf</i>	[61]
<i>GiSGiInterface_3GPP_TS29061_980.ddf</i>	[34]
<i>GiSGiInterface_3GPP_TS29061_930_QoS_Detailed.ddf</i>	[61]
<i>GiSGiInterface_3GPP_TS29061_b90.ddf</i>	[105]
<i>GiSGiInterface_3GPP_TS29061_b90_QoS_Detailed.ddf</i>	[105]
<i>GiSGiInterface_3GPP_TS29061_d70.ddf</i>	[117]
<i>GmbInterface_3GPP_TS29061_6f0.ddf</i>	[30]
<i>GmbInterface_3GPP_TS29061_720.ddf</i>	[31]
<i>GmbInterface_3GPP_TS29061_810.ddf</i>	[33]
<i>GmbInterface_3GPP_TS29061_930.ddf</i>	[61]
<i>GmbInterface_3GPP_TS29061_970.ddf</i>	[84]
<i>GmbInterface_3GPP_TS29061_980.ddf</i>	[34]
<i>GmbInterface_3GPP_TS29061_d70.ddf</i>	[117]
<i>GqInterface_PC_3GPP_TS29209_670.ddf</i>	[19]
<i>GqInterface_S3_ETSI_TS183017_V231.ddf</i>	[35]
<i>GxInterface_CRP_3GPP_TS29210_670.ddf</i>	[18]
<i>GxInterface_PCC_3GPP_TS29212_740.ddf</i>	[36]
<i>GxInterface_PCC_3GPP_TS29212_820.ddf</i>	[37]
<i>GxInterface_PCC_3GPP_TS29212_830.ddf</i>	[38]
<i>GxInterface_PCC_3GPP_TS29212_840.ddf</i>	[39]
<i>GxInterface_PCC_3GPP_TS29212_910.ddf</i>	[40]
<i>GxInterface_PCC_3GPP_TS29212_930.ddf</i>	[60]
<i>GxInterface_PCC_3GPP_TS29212_970.ddf</i>	[78]

<b>DDFs</b>	<b>Refs.</b>
<i>GxInterface_PCC_3GPP_TS29212_9b0.ddf</i>	[78]
<i>GxInterface_PCC_3GPP_TS29212_d70.ddf</i>	[112]
<i>GxInterface_PCC_3GPP_TS29212_f10.ddf</i>	
<i>NetworkAccessServer_IETF_RFC4005.ddf</i>	[11]
<i>RxInterface_PCC_3GPP_TS29214_830.ddf</i>	[42]
<i>RxInterface_PCC_3GPP_TS29214_990.ddf</i>	[42]
<i>RxInterface_PCC_3GPP_TS29214_a70.ddf</i>	[92]
<i>RxInterface_PCC_3GPP_TS29214_c10.ddf</i>	[100]
<i>RxInterface_PCC_3GPP_TS29214_f20.ddf</i>	
<i>ShInterface_3GPP_TS29329_620.ddf</i>	[43]
<i>ShInterface_3GPP_TS29329_750.ddf</i>	[44]
<i>ShInterface_3GPP_TS29329_820.ddf</i>	[45]
<i>ShInterface_3GPP_TS29329_a30.ddf</i>	[79]
<i>ShInterface_3GPP_TS29329_a50.ddf</i>	[99]
<i>SLgInterface_3GPP_TS29172_d10.ddf</i>	[110]
<i>SLhInterface_3GPP_TS29173_d00.ddf</i>	[111]
<i>Verizon_Specific_AVPs.ddf</i>	[51]
<i>e2Interface_ETSI_ES283035_121.ddf</i>	[53]
<i>e4Interface_ETSI_ES283034_220.ddf</i>	[54]
<i>TCOM_Specific_AVPs.ddf</i>	[55], [56]
<i>a4Interface_ETSI_ES183066_211.ddf</i>	[57]
<i>NGN_NetworkAccesses_ETSI_ES183020_111.ddf</i>	[58]
<i>a2Interface_ETSI_ES183059_1_211.ddf</i>	[59]
<i>AAAInterface_3GPP_TS29273_840.ddf</i>	[64], [65]
<i>AAAInterface_3GPP_TS29273_940.ddf</i>	[66], [67]
<i>AAAInterface_3GPP_TS29273_b30.ddf</i>	[90]
<i>AAAInterface_3GPP_TS29273_d60.ddf</i>	
<i>AAAInterface_3GPP_TS29273_f00.ddf</i>	[122]
<i>MobileIPv6_NAS_IETF_RFC5447.ddf</i>	[69]
<i>MobileIPv6_HA_IETF_RFC5778.ddf</i>	[68]
<i>MobileIPv4_Application_IETF_RFC4004.ddf</i>	[93]
<i>GmbInterface_3GPP_TS29061_930.ddf</i>	[70]
<i>Ericsson_Specific_AVPs.ddf</i>	[71]
<i>AAAInterface_3GPP_TS29272_940.ddf</i>	[72]
<i>AAAInterface_3GPP_TS29272_950.ddf</i>	[72]
<i>AAAInterface_3GPP_TS29272_970.ddf</i>	[76]

<b>DDFs</b>	<b>Refs.</b>
<i>AAAInterface_3GPP_TS29272_a30.ddf</i>	[83]
<i>AAAInterface_3GPP_TS29272_a60.ddf</i>	[79]
<i>AAAInterface_3GPP_TS29272_d70.ddf</i>	[113]
<i>AAAInterface_3GPP_TS29272_f10.ddf</i>	[121]
<i>GxInterface_PCC_3GPP_TS29212_8a0.ddf</i>	[74]
<i>GxInterface_PCC_3GPP_TS29212_8b1.ddf</i>	[75]
<i>RqInterface_ETSI_ES283026_241.ddf</i>	[77]
<i>Vimpelcom_Specific.ddf</i>	
<i>Vodafone_Specific.ddf</i>	
<i>ExtensibleAuthenticationProtocol_IETF_RFC4072.ddf</i>	[81]
<i>AAAInterface_3GPP_TS29273.ddf</i>	[79]
<i>WgInterface_3GPP_TS29234_910.ddf</i>	[80]
<i>SGmbInterface_3GPP_TS29061_980.ddf</i>	[34]
<i>SGmbInterface_3GPP_TS29061_b90.ddf</i>	[105]
<i>SGmbInterface_3GPP_TS29061_d70.ddf</i>	[117]
<i>GxaInterface_3GPP2_X_S0057_0_300.ddf</i>	[85]
<i>Alcatel_Lucent_Specific_AVPs.ddf</i>	[87],[97],[98]
<i>S9Interface_3GPP_TS29215_b40.ddf</i>	[88]
<i>MobileIPv6_HAAA_IETF_RFC5779.ddf</i>	[91]
<i>AAAInterface_3GPP_TS29272_b60.ddf</i>	[94]
<i>GxInterface_PCC_3GPP_TS29212_aa0.ddf</i>	[95]
<i>SyInterface_3GPP_TS29219_b30.ddf</i>	[96]
<i>Acision_Specific.ddf</i>	
<i>GxInterface_PCC_3GPP_TS29212_c52.ddf</i>	[101]
<i>DelegatedIPv6Prefix_IETF_RFC4818.ddf</i>	[106]
<i>AAAInterface_3GPP_TS29272_bd0.ddf</i>	[107]
<i>SKT_Specific_AVPs.ddf</i>	[108]
<i>DiameterRoutingMessagePriority_IETF_RFC7944.ddf</i>	[114]
<i>GxInterface_PCC_3GPP_TS29212_e00.ddf</i>	[115]
<i>ChargingApplications_3GPP_TS32299_d90.ddf</i>	[116]
<i>S6Interfaces_3GPP_TS29336_f00.ddf</i>	[118]
<i>T6Interfaces_3GPP_TS29128_f00.ddf</i>	[119]
<i>S6cInterface_3GPP_TS29338_f00.ddf</i>	[120]
<i>SGdGddInterface_3GPP_TS29338_f00.ddf</i>	[120]
<i>DOIC_RFC7683.ddf</i>	[125]

The DDF files can be used together without limitations except the DDF files for the same Diameter

application but with different version.

DDFs are separated according to standards. This induces the necessity of using multiple DDF modules to provide complete functionality of an interface. For details about which DDFs are necessary to assemble a complete interface, read the comments in the header of the given DDF modules!

<b>DDFs (in obsolete)</b>	<b>Refs.</b>
<i>DiameterBaseAVPs.ddf</i>	[3]
<i>3GPPChargingApplicationAVPs.ddf</i>	[17]
<i>3GPPCreditControlApplicationAVPs_v6110.ddf</i>	[9]
<i>3GPPCreditControlApplicationAVPs_v670.ddf</i>	[7]
<i>3GPPCreditControlApplicationAVPs_v690.ddf</i>	[8]
<i>3GPPShInterfaceAVPs_v620.ddf</i>	[43]
<i>3GPPShInterfaceAVPs_v750.ddf</i>	[44]
<i>DiameterChargingApplicationAVPs.ddf</i>	[17]
<i>DiameterCreditControlApplicationAVPs.ddf</i>	[6]
<i>DiameterCreditControlApplicationAVPs_aug05.ddf</i>	[6]
<i>DiameterMultimediaApplicationAVPs_v770.ddf</i>	[20]
<i>DiameterMultimediaApplicationAVPs_v810.ddf</i>	[21]
<i>DiameterNetworkAccessServerApplicationAVPs.ddf</i>	[11]
<i>DiameterOffLineCharging.ddf</i>	[13]
<i>EricssonChargingInterrogationProtocol_to_SDP_IP.ddf</i>	[8]
<i>EricssonProprietaryCxDx.ddf</i>	[46]
<i>EricssonServiceChargingApplicationAVPs.ddf</i>	[5]
<i>GiSpecificAVPs.ddf</i>	[32]
<i>GmbSpecificAVPs.ddf</i>	[31]
<i>GqSpecificAVPs.ddf</i>	[19]
<i>GxSpecificAVPs.ddf</i>	[18][36]
<i>GyPlusSpecificAVPs.ddf</i>	[12]
<i>IMSSpecificAVPs.ddf</i>	[16]
<i>PsSpecificAVPs.ddf</i>	[15]
<i>SRAPSpecificAVPs.ddf</i>	[14]
<i>VodafoneSpecificAVPs.ddf</i>	[22]
<i>GxPlus_Ericsson_5_1551_AXB250_10_4RevF.ddf</i>	[47]
<i>GyPlus_Ericsson_6_1551_AXB250_10_4RevC.ddf</i>	[48]
<i>GyPlus_Ericsson_6_1551_AXB250_10_4RevK.ddf</i>	[49]
<i>CxDxInterface_Ericsson_1551_FAY301_0059_PC26.ddf</i>	[46]

**NOTE** The *DiameterCreditControlApplicationAVPs.ddf* and *DiameterCreditControlApplicationAVPs\_aug05.ddf* describe the same Diameter application. The only difference between them is that the *DiameterCreditControlApplicationAVPs.ddf* module mapped the enumerated AVPs to Unsigned32 types, while the latter describes them as they are defined in the RFC.

## Modifications/deviations related to the protocol specification

### Unimplemented Messages, Information Elements and Constants

None.

### Protocol Modifications/Deviations

#### *DiameterCreditControlApplication.ddf* [6]

The following enumerated AVPs have been implemented as Unsigned32 AVPs in order to allow arbitrary values:

- CC-Request-Type (416)
- CC-Session-Failover (418)
- CC-Unit-Type (454)
- Check-Balance-Result (422)
- Credit-Control (426)
- Credit-Control-Failure-Handling (427)
- Direct-Debiting-Failure-Handling (428)
- Final-Unit-Action (449)
- Multiple-Services-Indicator (455)
- Redirect-Address-Type (433)
- Requested-Action (436)
- Subscription-Id-Type (450)
- Tariff-Change-Usage (452)
- User-Equipment-Info-Type (459)

This module must not be used together with the *DiameterCreditControlApplication\_aug05.ddf*!

## *DiameterNetworkAccessServerApplicationAVPs.ddf* [11]

The following enumerated AVPs have been implemented as Unsigned32 AVPs in order to allow arbitrary values:

- NAS-Port-Type (61)
- Service-Type (6)
- Tunnel-Type (64)
- Tunnel-Medium-Type (65)

## *PsSpecificAVPs.ddf* [16]

The following enumerated AVPs have been implemented as Unsigned32 AVPs in order to allow arbitrary values:

- CC-Request-Type (416)
- Multiple-Services-Indicator (455)
- CC-Session-Failover (418)
- Credit-Control-Failure-Handling (427)

## *IMSSpecificAVPs.ddf* [17]

The following enumerated AVPs have been implemented as Unsigned32 AVPs in order to allow arbitrary values:

- Requested-Action (436)
- Multiple-Services-Indicator (455)
- CC-Session-Failover (418)
- Credit-Control-Failure-Handling (427)
- CC-Request-Type (416)

Because of the missing AVP codes and types the following AVPs are not implemented:

- Extended-Information
- Operation-Event-Failure-Action

## *DiameterChargingApplicationAVPs.ddf* [18]

The following enumerated AVPs have been implemented as Unsigned32 AVPs in order to allow arbitrary values:

- CC-Request-Type (416)
- CC-Session-Failover (418)
- CC-Unit-Type (454)

- Check-Balance-Result (422)
- Credit-Control (426)
- Credit-Control-Failure-Handling (427)
- Direct-Debiting-Failure-Handling (428)
- Final-Unit-Action (449)
- Multiple-Services-Indicator (455)
- Redirect-Address-Type (433)
- Requested-Action (436)
- Subscription-Id-Type (450)
- Tariff-Change-Usage (452)
- User-Equipment-Info-Type (459)

Because of the missing AVP code and type the following AVP is not implemented:

- Operator-Name

### *3GPPChargingApplicationAVPs.ddf* [18]

The following enumerated AVPs have been implemented as Unsigned32 AVPs in order to allow arbitrary values:

- 3GPP-PDP-Type (3)
- Application-Service-Type (2102)
- MBMS-2G-3G-Indicator (907)
- Type-Number (1204)

### *GxSpecificAVPs.ddf* [19], [19]

The following enumerated AVPs have been implemented as Unsigned32 AVPs in order to allow arbitrary values:

- QoS-Class-Identifier (1028)
- CC-Request-Type (416)

Because of the missing AVP codes the following AVPs are not implemented:

- QoS-Negotiation
- QoS-Upgrade

### *GqSpecificAVPs.ddf* [22]

The following enumerated AVPs have been implemented as Unsigned32 AVPs in order to allow arbitrary values:

- Media-Type (520)

*GxInterface\_PCC\_3GPP\_TS29212\_910.ddf* [42] and  
*GxInterface\_PCC\_3GPP\_TS29212\_930.ddf* [60]

The following enumerated AVPs have been implemented as Unsigned32 AVPs in order to allow arbitrary values:

- QoS-Class-Identifier (1028)

*AAAInterface\_3GPP\_TS29272\_950.ddf* [73]

The following enumerated AVPs have been implemented as Unsigned32 AVPs in order to allow arbitrary values:

Trace-Depth (1462)

*GmbInterface\_3GPP\_TS29061\_980.ddf* [34]

The following enumerated AVPs have been implemented as Unsigned32 AVPs in order to allow arbitrary values:

MBMS-HC-Indicator (922)

*CxDxInterface\_Ericsson\_1551\_FAY301\_0059\_PC26.ddf*

Until version R24B The ddf file contained duplicated AVPs with *Ericsson\_Specific\_AVPs.ddf*. In version R24C, these duplications were removed and the prefix of the AVPs was changed from **ECX\_** to **E\_**. This change is not backward compatible.

## Upgrading Templates Used by the DIAMETER Test Port

The DPMG type structure differs from the one that is used in the DIAMETER message test port. This causes backward incompatibilities in the TTCN-3 type definition module. Therefore, functions and templates developed for DIAMETER message test port need to be updated according to the changes of the type definition so that they can be used with DPMG.

In case new fields were added into existing record or set types, the new templates should contain these fields set to omit.

In case a type has changed completely the whole template or part of template must be changed.

If a function is accessing a field that has changed that function needs to be updated as well.

Here you can find a list of major changes:

1. The module name containing DIAMETER type definitions has been changed, thus you should replace `DIAMETERmsg_Types` by `DIAMETER_Types` in import lines of modules using DIAMETER type definitions.
2. The name of top level PDU changed from `DIAMETER_message` to `PDU_DIAMETER`.
3. Command flags of the PDU are handled as an 8 bit length bitfield instead of separated bits.
4. The field of the PDU contains the list of AVPs renamed from `AVPs` to `avps`.
5. The enumerated type that contains command codes is renamed from `message_code` to `Command_Code`, and the names of enumerated items are changed according to the naming convention of the protocol module generator.
6. The type of fields `hop_by_hop_id` and `end_to_end_id` is changed from integer to a 4 octets long octetstring.
7. The type tree that models the AVPs was modified. A `GenericAVP` type was introduced for error-handling purposes. Its `avp` branch contains the correctly decoded `AVP`, but if something goes wrong during decoding, the `avp_UNKNOWN` branch is used instead, which is of type octetstring.
8. Instead of a union type `AVP`, an `AVP` record is applied with two fields that contain the `AVP_Header` and the `AVP_Data`, respectively.
9. The field names of the union type `AVP_Data` are denominated according to the naming convention.
10. All type names of AVPs are changed according to the naming convention.
11. In the `AVP_Header` type the name and type of the field contains the AVP code changed. The name is changed from `AVP_code` to `avp_code`. The type has changed from integer to a union of enumerations.
12. AVP flags in `AVP_Header` type are handled as an 8 bit long bitfield instead of separated bits.
13. The type of `vendor_id` field is changed from octetstring to enumeration.
14. Name of enumeration types and values within AVPs are changed according to 3.1.2.
15. To ease the process of template development the `DPMG AWK` script generates `AVP_Code` constants. These make it possible to avoid using the enumeration union and provide an easy way to reference an AVP code. The names of the constants take the following form:

```
c_AVP_Code_<Application-Name>_<Official-Vendor-Id>_<Official-AVP-Name>
```

Example:

```
const AVP_Code c_AVP_Code_SCAP_Ericsson_Cost :=
{
    vendor_id_Ericsson := avp_code_SCAP_Ericsson_Cost
}
```

It is recommended to use these constants in order to prevent incompatibilities with future versions of DPMG.

# Examples

The "demo" directory of the deliverable contains examples (*DIAMETER\_Demo.ttcn*) and reusable modules (*DIAMETER\_Mapping.ttcn*) for DPMG.

## Mapping module

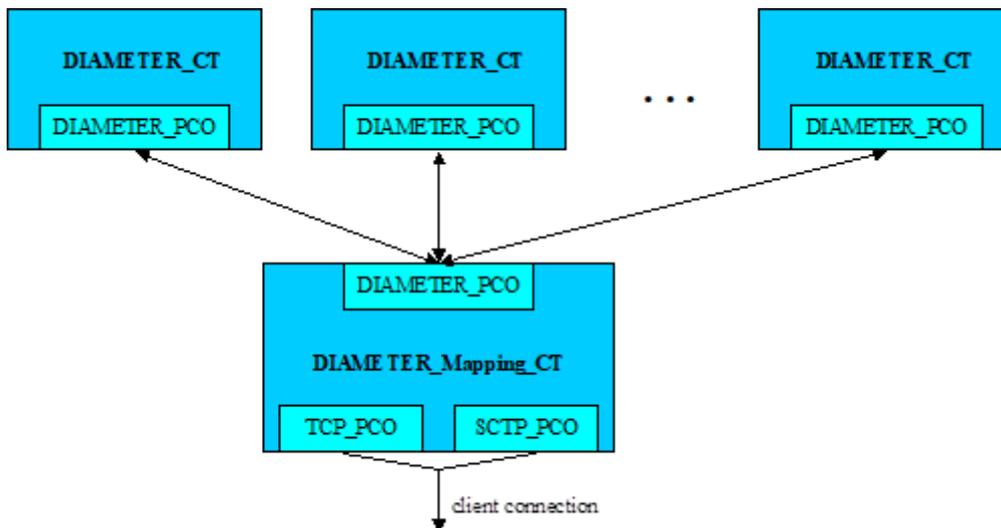
The *DIAMETER\_Mapping\_CT* component implemented in the *DIAMETER\_Mapping.ttcn* module provides the connection between the DIAMETER protocol module and the SCTP (CNL 113 469) or the TCP (CNL 113 347) test port. It maintains SCTP or TCP connections and encodes/decodes Diameter messages.

The mapping component supports client and server mode operations and sends notifications about the state of the underlying TCP or SCTP connections to the mapping users.

## Client Mode

### Overview

See the client mode mapping below:



In client mode the *DIAMETER\_Mapping\_CT* initiates connection to the destination host using either the *TCP\_PCO* or the *SCTP\_PCO* port. Several users may connect to the mapping component (*Client mode mapping*). The users can send *PDU\_DIAMETER* messages to the mapping component, which will be encoded and will be sent through the *TCP\_PCO* or the *SCTP\_PCO* ports. The mapping component keeps track of the end-to-end id and hop-by-hop id of each Diameter message. The corresponding answers (with the same hop-by-hop and end-to-end ids) are routed back to the originating user.

The mapping component can inform the users about the state of the connection. The users must register themselves in the mapping component using the *ASP\_DIA\_Mapping\_Registration* ASP in order to receive notifications, which will be sent to them via the *ASP\_DIA\_Mapping\_Notification* ASP.

In client mode, the mapping component supports reconnection: whenever the connection is

disconnected, the component detects it and automatically tries to re-establish it again.

The above-described functionality is implemented for each supported underlying protocol in separate functions of the *DIAMETER\_Mapping.ttcn* module:

1. SCTP: `f_DIA_SCTP_Mapping_Client()`
2. TCP: `f_DIA_TCP_Mapping_Client()`

## Configuration

The following module parameters are used in client mode:

Parameter Name	Type	Description
<code>tsp_hostname</code>	charstring	Mandatory. Contains the IP address of the destination host in dot notation.
<code>tsp_portnumber</code>	integer	Mandatory. Contains the port number of the destination host.
<code>tsp_reconnect</code>	boolean	Optional, its default value is true. Enables reconnect mode .
<code>tsp_reconnect_timeout</code>	float	Optional, its default value is "2.0". Specifies the time interval between two connection attempts in reconnect mode.
<code>tsp_connect_timeout</code>	float	Optional, its default value is "5.0". Specifies the time the mapping component waits for an answer after a connection request was sent.

If SCTP connection is used, the SCTP test port must be configured in the following way:

- `server_mode := "no"`
- Other SCTP test port parameters must not be used.

If TCP connection is used, the TCP test port must be configured in the following way:

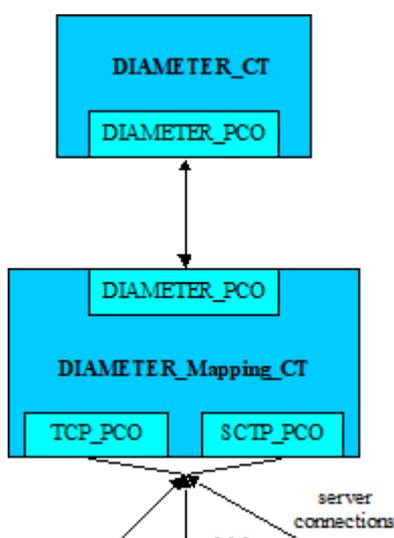
- `use_connection_ASs := "yes"`
- `server_mode := "no"`
- `halt_on_connection_reset := "no"`
- `client_TCP_reconnect := "yes"`
- `packet_hdr_length_offset := "1"`
- `packet_hdr_nr_bytes_in_length := "3"`
- `packet_hdr_byte_order := "MSB"`
- Other TCP test port parameters should not be used.

## NOTE

When the TCP port is not able to connect to the destination host it exits with a dynamic test case error, therefore the mapping component is not able to control the reconnection process using TCP test port ASPs. The reconnection in case of TCP can be enabled with the help of the `client_TCP_reconnect` test port parameter. Delays and the number of attempts can be configured using the `TCP_reconnect_delay` and `TCP_reconnect_attempts` optional TCP test port parameters. For further information, see [8].

## Server Mode

See server mode mapping below:



In server mode the `DIAMETER_Mapping_CT` starts listening on a configured port using either the `TCP_PCO` or the `SCTP_PCO` port and waits for incoming connections. Only one user component connect to the mapping component (see [Server mode mapping](#)). The user component can receive notifications about connection establishments and disconnections (`ASP_DIA_Mapping_Notification` ASP) and can send/receive Diameter messages (`PDU_DIAMETER_Server` PDU). These ASPs has a `client_id` field that appoints which connection it is related to.

The above-described functionality is implemented in separate functions of the `DIAMETER_Mapping.ttcn` module for each supported underlying protocol:

1. Sctp: `f_DIA_SCTP_Mapping_Server()`
2. TCP: `f_DIA_TCP_Mapping_Server()`

## Configuration

The following module parameters are used in client mode:

Parameter Name	Type	Description
<code>tsp_hostname</code>	charstring	Mandatory. Contains the IP address of the listening interface in dot notation.

Parameter Name	Type	Description
<code>tsp_portnumber</code>	integer	Mandatory. Contains the port number of the listening socket.

If SCTP connection is used, the SCTP test port must be configured in the following way:

- `server_mode := "yes"`
- `local_IP_address` contains the IP address of the server in dot notation.
- `local_port` contains the port number of the server
- Other SCTP test port parameters should not be used.

#### NOTE

In case of SCTP the listening interface must be given using the `local_IP_address` and `local_port` SCTP test port parameters. Setting the `tsp_hostname` and `tsp_portnumber` module parameters has no effect, since the SCTP test port has no ASP for initiating listening.

If TCP connection is used, the TCP test port must be configured in the following way:

- `use_connection_ASps := "yes"`
- `server_mode := "yes"`
- `halt_on_connection_reset := "no"`
- `packet_hdr_length_offset := "1"`
- `packet_hdr_nr_bytes_in_length := "3"`
- `packet_hdr_byte_order := "MSB"`
- Other TCP test port parameters should not be used.

## ASPs of the `DIAMETERmsg_PT` port

The users can connect to the mapping component via a `DIAMETERmsg_PT` (DIA\_PCO) port. This port conveys the following messages and ASps:

- `PDU_DIAMETER` - This type contains the Diameter message representation in TTCN-3
- `PDU_DIAMETER_Server` - This type is for server mode. It has two fields:
  - `data` - Its type is `PDU_DIAMETER` and contains a Diameter PDU
  - `client_id` - Its type is integer. Each separate connection has a unique id in server mode. This field appoints which connection the Diameter message is related to.
- `ASP_DIA_Mapping_Notification` - It is the type for carrying notifications. The following fields are available:
  - `notification` - It is of type enumeration and describes the notification type. It can be one of the following values, which are self-explanatory:
    - `CONNECTION_IS_UP`
    - `CONNECTION_IS_DOWN`
    - `SEND_FAILED`

- `TRANSMISSION_FAILED`
- `pdu` - This field is optional. It is present in case of a `TRANSMISSION_FAILED` notification and contains the Diameter PDU, that wasn't delivered.
- `client_id` - This field is optional, and only present in server mode. Appoints which connection the notification is related to.
- `ASP_DIA_Mapping_Registration` - It is of type enumeration and is used by the mapping users to subscribe to and unsubscribe from notifications. The following values are available:
  - `REGISTRATION`
  - `REGISTRATION_ACK`
  - `DEREGISTRATION`
  - `DEREGISTRATION_ACK`

To subscribe for notifications:

- The users must issue a REGISTER.
- The mapping component answers this with a `REGISTER_ACK` and immediately sends an `ASP_DIA_Mapping_Notification` as well, that informs the user whether the transport connection is up or down.

To unsubscribe from notifications:

- The users must issue a DEREGISTER.
- The mapping component answers with a `DEREGISTER_ACK`. After receiving this message the mapping user component might terminate.

## Demo Module

### Test Cases

The module `DIAMETER_Demo.ttcn` contains example testcases with their used templates, to show how the templates based on Diameter type definitions look like, and how to start and use the mapping module. The following testcases demonstrates client and server mode operation:

In case the transport layer is SCTP:

- `tc_DIAMETER_SCTP_Client_Demo()`
- `tc_DIAMETER_SCTP_Server_Demo()`

In case the transport layer is TCP:

- `tc_DIAMETER_TCP_Client_Demo()`
- `tc_DIAMETER_TCP_Server_Demo()`

### Configuration Files

There are example configuration files in the demo directory as well, that can be used when

executing the example test cases:

- *DIAMETER\_SCTP\_Client\_Demo.cfg*
- *DIAMETER\_SCTP\_Server\_Demo.cfg*
- *DIAMETER\_TCP\_Client\_Demo.cfg*
- *DIAMETER\_TCP\_Server\_Demo.cfg*

## Examples for Building the Project

There can be found an example *Makefile* for those who prefer command line compilation. Softlinks must be created before invoking the *Makefile*.

For GUI users there is a *DIAMETER\_Demo.prj* file as an example. Do not forget to generate softlinks for the files under the 'Misc Files' section.

## Script to Modify *Makefile*

Here is an example shell script to modify the generated *Makefile*. This script can be used by the GUI.

```
#!/bin/sh

sed -e '
s/TTCN3_MODULES =/TTCN3_MODULES = DIAMETER_Types.ttcn/g
/# Add your rules here if necessary./ {
a\
#
a\

a\
AWK=/usr/local/bin/gawk
a\

a\
DIAMETER_Types.ttcn: DiameterBaseTypes.ddf DiameterBaseAVPs.ddf AVP.awk
a\
    $(AWK) -f AVP.awk $(filter %.ddf,$^) > $@
a\

a\
#
a\
# End of additional rules for DPMG
}
' <$1 >$2
```

# Abbreviations

## **ASP**

Abstract Service Primitive

## **AVP**

Attribute Value Pair

## **DPMG**

Diameter Protocol Module Generator

## **GNU**

Gnu's Not Unix

## **GUI**

Graphical User Interface

## **PDU**

Protocol Data Unit

## **TTCN-3**

Testing and Test Control Notation version 3

# Terminology

## **DDF:**

Diameter Definitions File: TTCN-3 type definitions describing Diameter AVPs outside module

# References

[1] ETSI ES 201 873-1 v4.5.1

The Testing and Test Control Notation version 3. Part 1: Core Language

[2] Programmer's Technical Reference for TITAN TTCN-3 Test Executor

[3] DIAMETER Protocol Module Generator for TTCN-3 Toolset with TITAN, Product Revision Information

[4] DIAMETER Test Port for TTCN-3 Toolset with TITAN, PRI

[5] RFC 3588

Diameter Base Protocol

[6] The GNU Awk User's Guide,

<http://www.gnu.org/software/gawk/manual/gawk.html>

- [7] TCP Socket Test Port for TTCN-3 Toolset with TITAN, UG
  
- [8] RFC 4006  
Diameter Credit-Control Application
  
- [9] 3GPP TS 32.299 v6.7.0  
Diameter Charging Applications
  
- [10] 3GPP TS 32.299 v6.9.0  
Diameter Charging Applications
  
- [11] 3GPP TS 32.299 v6.11.0  
Diameter Charging Applications
  
- [12] Charging Interrogation Protocol
  
- [13] RFC 4005  
Diameter Network Access Server Application
  
- [14] Gy + Interface Description
  
- [15] Off-line Charging in MTAS
  
- [16] SRAP Interface Description
  
- [17] 3GPP TS 32.251 V8.1.0  
Packet Switched (PS) domain charging
  
- [18] 3GPP TS 32.260 V8.3.0  
IP Multimedia Subsystem (IMS) charging
  
- [19] 3GPP TS 32.299 V8.2.0  
Diameter charging applications
  
- [20] 3GPP TS 29.210 V6.7.0  
Charging rule provisioning over Gx interface
  
- [21] 3GPP TS 29.209 V6.7.0  
Policy control over Gq interface
  
- [22] 3GPP TS 29.229 V7.7.0  
Cx and Dx interfaces based on the Diameter protocol; Protocol details
  
- [23] 3GPP TS 29.229 V8.1.0  
Cx and Dx interfaces based on the Diameter protocol; Protocol details
  
- [24] Vodafone Gx+ Specification v1.3.1
  
- [25] 3GPP TS 29.210 V6.7.0  
Charging rule provisioning over Gx interface
  
- [26] 3GPP TS 32.299 v8.5.0

## Diameter Charging Applications

[27] 3GPP TS 32.299 v8.7.0

Diameter Charging Applications

[28] 3GPP TS 32.299 v9.0.0

Diameter Charging Applications

[29] 3GPP TS 29.229 V6.15.0

Cx and Dx interfaces based on the Diameter protocol; Protocol details

[30] 3GPP TS 29.229 V8.4.0

Cx and Dx interfaces based on the Diameter protocol; Protocol details

[31] 3GPP TS 29.229 V8.8.0

Cx and Dx interfaces based on the Diameter protocol; Protocol details

[32] 3GPP TS 29.061 V6.15.0

Interworking between the Public Land Mobile Network supporting packet based services and Packet Data Networks (PDN)

[33] 3GPP TS 29.061 V7.2.0

Interworking between the Public Land Mobile Network supporting packet based services and Packet Data Networks (PDN)

[34] 3GPP TS 29.061 V7.7.0

Interworking between the Public Land Mobile Network supporting packet based services and Packet Data Networks (PDN)

[35] 3GPP TS 29.061 V8.1.0

Interworking between the Public Land Mobile Network supporting packet based services and Packet Data Networks (PDN)

[36] 3GPP TS 29.061 V9.8.0

Interworking between the Public Land Mobile Network supporting packet based services and Packet Data Networks (PDN)

[37] ETSI 183.017 v2.3.1

DIAMETER protocol for session based policy set-up information exchange between the Application Function (AF) and the Service Policy Decision Function (SPDF)

[38] 3GPP TS 29.212 V7.4.0

Policy and Charging Control over Gx reference point

[39] 3GPP TS 29.212 V8.2.0

Policy and Charging Control over Gx reference point

[40] 3GPP TS 29.212 V8.3.0

Policy and Charging Control over Gx reference point

[41] 3GPP TS 29.212 V8.4.0

Policy and Charging Control over Gx reference point

[42] 3GPP TS 29.212 V9.1.0

Policy and Charging Control over Gx reference point

[43] 3GPP TS 29.212 V9.1.0

Policy and Charging Control over Gx reference point

[44] 3GPP TS 29.214 V8.3.0

Policy and Charging Control over Rx reference point

[45] 3GPP TS 29.329 v6.2.0

Sh Interface based on the Diameter PROTOCOL

[46] 3GPP TS 29.329 v7.5.0

Sh Interface based on the Diameter PROTOCOL

[47] 3GPP TS 29.329 v8.2.0

Sh Interface based on the Diameter PROTOCOL

[48] Service Contract, Diameter Cx Application

[49] Gx+ Interface Description

[50] Gy+ Interface Description

[51] Gy+ Interface Description

[52] Intelligent Packet Core Vodafone Diameter CCA Specification for the CLCI Version 2

[53] Verizon LTE Rf Interface Specification

<https://erilink.ericsson.se/eridoc/erl/objectId/09004cff87438f5d?docno=17/15519-FCP111391Uen&action=current&format=pdf>

[54] RFC 5090

RADIUS Extension for Digest Authentication

[55] ETSI ES 283 035 v1.2.1 (2007-06)

TISPAN; NASS; e2 interface based on the DIAMETER protocol

[56] ETSI ES 283 034 v2.2.0 (2008-05)

TISPAN; NASS; e4 interface based on the DIAMETER protocol

[57] SBG AF e2 Interface

[58] SBG AF e2 Extensions

[59] ETSI TS 183 066 V2.1.1 (2009-01)

Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN);

Network Attachment Sub-System (NASS);

a4 interface based on the DIAMETER protocol

[60] ETSI TS 183 020 V1.1.1 (2006-03)

Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN);

Network Attachment: Roaming in TISPAN

NGN Network Accesses;

Interface Protocol Definition

[61] ETSI TS 183 059-1 V2.1.1 (2009-08)

Telecommunications and Internet converged Services and Protocols for Advanced Networks (TISPAN);

Network Attachment Sub-System (NASS);

a2 interface based on the DIAMETER protocol

[62] 3GPP TS 29.212 V9.3.0

Policy and Charging Control over Gx reference point

[63] 3GPP TS 29.061 V9.3.0

Interworking between the Public Land Mobile Network (PLMN) supporting packet based services and Packet Data Networks (PDN)

[64] 3GPP TS 32.299 V9.4.0

Diameter Charging Applications

[65] 3GPP TS 32.299 V9.11.0

Diameter Charging Applications

[66] 3GPP TS 29.229 V9.2.0

Cx and Dx interfaces based on the Diameter protocol; Protocol details

[67] 3GPP TS 29.273 V8.4.0

Evolved Packet System (EPS); 3GPP EPS AAA interfaces

[68] Statement of Compliance 3GPP TS 29.273 3GPP EPS AAA interface

[69] 3GPP TS 29.273 V9.4.0

Evolved Packet System (EPS); 3GPP EPS AAA interfaces

[70] Statement of Compliance 3GPP TS 29.273 3GPP EPS AAA interface

[71] RFC5778

Diameter Mobile IPv6: Support for Home Agent to Diameter Server Interaction

[72] RFC5447

Diameter Mobile IPv6: Support for Network Access Server to Diameter Server

[73] 3GPP TS 29.061 V9.4.0

Interworking between the Public Land Mobile Network supporting packet based services and Packet Data Networks (PDN)(Release 9)

[74] Current Ericsson Diameter AVP Assignments Rev 1.111, 2010-01-19

<http://www.lmera.ericsson.se/~snmp/diameter-assignments.html>

- [75] 3GPP TS 29.272 V9.4.0 (2010-09)  
Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol(Release 9)
- [76] 3GPP TS 29.272 V9.5.0 (2010-12)  
Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol(Release 9)
- [77] 3GPP TS 29.212 V8.10.0 (2010-12)  
Policy and Charging Control over Gx reference point
- [78] 3GPP TS 29.212 V8.11.1 (2011-03)  
Policy and Charging Control over Gx reference point
- [79] 3GPP TS 29.272 V9.7.0 (2011-06)  
Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol(Release 9)
- [80] ETSI ES 283 026 V2.4.1 (2008-11)  
Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); Resource and Admission Control;  
Protocol for QoS reservation information exchange between the Service Policy Decision Function (SPDF) and the Access-Resource and Admission Control Function (A-RACF) in the Resource and Protocol specification
- [81] 3GPP TS 29.212 V9.7.0  
Policy and Charging Control over Gx reference point
- [82] 3GPP TS 29.329 v10.3.0  
Sh Interface based on the Diameter PROTOCOL
- [83] 3GPP TS 29.273 v9.4.0  
Evolved Packet System (EPS), 3GPP EPS AAA interfaces
- [84] 3GPP TS 29.234 v9.1.03  
GPP system to Wireless Local Area Network (WLAN) interworking
- [85] RFC 4072  
Diameter Extensible Authentication Protocol (EAP) Application
- [86] 3GPP TS 29.272 V10.3.0 (2011-06)  
Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol(Release 10)
- [87] 3GPP TS 29.061 V9.7.0 (2011-09)  
Interworking between the Public Land Mobile Network supporting packet based services and Packet Data Networks (PDN)(Release 9)
- [88] 3GPP2 X.S0057-0 Version 3.0  
E\_-UTRAN\_ – eHRPD Connectivity and Interworking Core Network Aspects

[89] Diameter Offline Charging in MTAS

[90] 3GPP TS 29.212 V11.4.0

Policy and Charging Control (PCC) over Gx/Sd reference point (Release 11)

[91] 3GPP TS 29.272 V10.6.0 (2012-03)

Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol(Release 10)

[92] 3GPP TS 29.273 v9.4.0

Evolved Packet System (EPS), 3GPP EPS AAA interfaces

[93] RFC 5779

Diameter Proxy Mobile IPv6: Mobile Access Gateway and Local Mobility Anchor Interaction with Diameter Server

[94] 3GPP TS 29.214 V10.7.0

Policy and Charging Control over Rx reference point

[95] RFC 4004

Diameter Mobile IPv4 Application

[96] 3GPP TS 29.272 V11.6.0 (2013-03)

3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol (Release 11)

[97] 3GPP TS 29.212 V10.10.0 (2013-03)

3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Policy and Charging Control (PCC) over Gx reference point (Release 10)

[98] 3GPP TS 29.219 V11.4.0 (2013-03)

3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Policy and Charging Control: Spending Limit Reporting over Sy reference point (Release 11)

[99] IS Verizon Ro Interface for Prompt and Collect in MTAS

[100] Ro interface enhancements based on Verizon's call flows and requirements

[101] 3GPP TS 29.329 V10.5.0 (2013-03)

Technical Specification 3rd Generation Partnership Project;  
Technical Specification Group Core Network and Terminals;  
Sh Interface based on the Diameter protocol;  
Protocol details (Release 10)

[102] 3GPP TS 29.214 V12.1.0

Policy and Charging Control over Rx reference point

[103] 3GPP TS 29.212 V12.5.2

3rd Generation Partnership Project;  
Technical Specification Group Core Network and Terminals;

Policy and Charging Control (PCC);  
Reference points (Release 12)

[104] 3GPP TS 32.299 V12.6.0  
Diameter Charging Applications

[105] 3GPP TS 32.299 V13.4.0  
Diameter Charging Applications

[106] 3GPP TS 29.229 V12.3.0  
Cx and Dx interfaces based on the Diameter protocol; Protocol details

[107] 3GPP TS 32.299 V11.15.0  
Diameter Charging Applications

[108] 3GPP TS 29.061 V11.9.0 (2014-12)  
Interworking between the Public Land Mobile Network (PLMN) supporting packet based services and Packet Data Networks (PDN)(Release 11)

[109] RFC 4818  
RADIUS Delegated-IPv6-Prefix Attribute

[110] 3GPP TS 29.272 V11.13.0  
3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol (Release 11)

[111] WP SKT-Zone

[112] 3GPP TS 29.172 V13.1.0 (2016-06)  
3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Location Services (LCS); Evolved Packet Core (EPC) LCS Protocol (ELP) between the Gateway Mobile Location Centre (GMLC) and the Mobile Management Entity (MME); SLg interface (Release 13)

[113] 3GPP TS 29.173 V13.0.0 (2015-12)  
3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Location Services (LCS); Diameter-based SLh interface for Control Plane LCS (Release 13)

[114] 3GPP TS 32.212 V13.7.0  
Policy and Charging Control (PCC); Reference points

[115] 3GPP TS 32.272 V13.7.0  
Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol

[116] RFC 7944  
Diameter Routing Message Priority

[117] 3GPP TS 29.212 V14.0.0  
3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Policy and Charging Control (PCC); Reference points (Release 14)

[118] 3GPP TS 32.299 v13.9.0  
Diameter Charging Applications

[119] 3GPP TS 29.061 V13.7.0 (2017-03)  
3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals;  
Interworking between the Public Land Mobile Network (PLMN) supporting packet based services  
and Packet Data Networks (PDN) (Release 13)

[120] 3GPP TS 29.336 V15.0.0 (2017-09)  
3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals;  
Home Subscriber Server (HSS) diameter interfaces for interworking with packet data networks and  
applications (Release 15)

[121] 3GPP TS 29.128 V15.0.0 (2017-09)  
3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals;  
Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) interfaces for  
interworking with packet data networks and applications (Release 15)

[122] 3GPP TS 29.338 V15.0.0 (2017-09)  
3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals;  
Diameter based protocols to support Short Message Service (SMS) capable Mobile Management  
Entities (MMEs) (Release 15)

[123] 3GPP TS 29.272 V15.1.0 (2017-09)  
3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals;  
Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node  
(SGSN) related interfaces based on Diameter protocol (Release 15)

[124] 3GPP TS 29.273 V15.0.0 (2017-09)  
3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals;  
Evolved Packet System (EPS); 3GPP EPS AAA interfaces (Release 15)

[125] RFC 7683  
Diameter Overload Indication Conveyance