|  |  |
| --- | --- |
|  |  |

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

Contents

[1 Introduction 3](#_Toc507670480)

[1.1 About this Document 3](#_Toc507670481)

[1.1.1 How to Read this Document 3](#_Toc507670482)

[1.1.2 Presumed Knowledge 3](#_Toc507670483)

[2 General overview 3](#_Toc507670484)

[2.1 Background 3](#_Toc507670485)

[2.2 Overview 4](#_Toc507670486)

[3 Functionality 4](#_Toc507670487)

[3.1 Naming Conventions 5](#_Toc507670488)

[3.1.1 Key to unique naming of identifiers 5](#_Toc507670489)

[3.1.2 How to model enumeration type AVPs in DDFs? 6](#_Toc507670490)

[3.2 Script operation 7](#_Toc507670491)

[3.2.1 Load and parse all input files 7](#_Toc507670492)

[3.2.2 Type identifiers 7](#_Toc507670493)

[3.2.3 Vendor\_Id 7](#_Toc507670494)

[3.2.4 AVP\_Code\_<Official-Vendor-Id> 8](#_Toc507670495)

[3.2.5 AVP\_Code 8](#_Toc507670496)

[3.2.6 AVP\_Header 8](#_Toc507670497)

[3.2.7 AVP\_Data 9](#_Toc507670498)

[3.2.8 AVP 9](#_Toc507670499)

[3.2.9 GenericAVP 9](#_Toc507670500)

[3.2.10 Command\_Code 10](#_Toc507670501)

[3.2.11 Output TTCN-3 module 10](#_Toc507670502)

[3.2.12 Output encoder/decoder 10](#_Toc507670503)

[3.2.13 Detailed VMP and RPET bits 10](#_Toc507670504)

[3.2.14 Bigint support for unsigned32 and 64 bit integer 10](#_Toc507670505)

[3.3 Backward incompatibilities 10](#_Toc507670506)

[3.3.1 CxDxInterface\_Ericsson\_1551\_FAY301\_0059\_PC26.ddf 10](#_Toc507670507)

[3.4 System Requirements 11](#_Toc507670508)

[3.5 Installation 11](#_Toc507670509)

[3.6 Generation of the DIAMETER\_Types.ttcn 11](#_Toc507670510)

[3.6.1 Supported options 12](#_Toc507670511)

[3.6.2 Makefile preparation 13](#_Toc507670512)

[3.7 Helper functions 13](#_Toc507670513)

[3.8 Encoding/Decoding Functions 14](#_Toc507670514)

[3.9 Error handling 14](#_Toc507670515)

[3.10 Limitations 15](#_Toc507670516)

[4 Protocol versions 15](#_Toc507670517)

[4.1 Product contents, structure 15](#_Toc507670518)

[4.2 Protocol version implemented 15](#_Toc507670519)

[4.3 Modifications/deviations related to the protocol specification 22](#_Toc507670520)

[4.3.1 Unimplemented Messages, Information Elements and Constants 22](#_Toc507670521)

[4.3.2 Protocol Modifications/Deviations 22](#_Toc507670522)

[5 Upgrading templates used by the DIAMETER test port 25](#_Toc507670523)

[6 Examples 26](#_Toc507670524)

[6.1 Mapping module 27](#_Toc507670525)

[6.1.1 Client mode 27](#_Toc507670526)

[6.1.2 Server mode 29](#_Toc507670527)

[6.1.3 ASPs of the DIAMETERmsg\_PT port 30](#_Toc507670528)

[6.2 Demo module 32](#_Toc507670529)

[6.2.1 Test cases 32](#_Toc507670530)

[6.2.2 Configuration files 32](#_Toc507670531)

[6.2.3 Examples for building the project 32](#_Toc507670532)

[6.2.4 Script to modify Makefile 33](#_Toc507670533)

[7 Abbreviations 33](#_Toc507670534)

[8 Terminology 33](#_Toc507670535)

[9 References 34](#_Toc507670536)

# Introduction

## 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. This document should be read together with Product Revision Information [3]

### Presumed Knowledge

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

Basic knowledge of the Diameter protocol [5] is valuable to use this protocol module.

# General overview

## Background

Former solution for testing Diameter applications is based on the Diameter Test Port [4]. 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 e.g. 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.



Figure 1 DPMG Architecture

# Functionality

The DIAMETER protocol module (e.g. 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 (Figure 1).

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** |
| BaseTypes\_IETF\_RFC3588 | Diameter Base Protocol [5] related type definitions |
| Base\_IETF\_RFC3588 | Diameter Base Protocol [5] 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

1. 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 …

1. The entire comment line with the information MUST stand in the same line (no line breaks when it splits to multiple lines).
2. 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.
3. 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.
4. 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)!
5. 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:
6. Hyphens and spaces must be replaced by a single underscore
7. 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 [5] – 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 <Application-Name>s: BASE, NASREQ  
   Example <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 (e.g. 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:

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 obsoleted 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 [5].

### 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 will contain 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) "LENGTHTO(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 (e.g. 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]. Please 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 (e.g. 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 [6] so it must be available on the system.

## Generation of the DIAMETER\_Types.ttcn

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. Note that 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*

Please note that 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 $(filter %.ddf,$^)

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. Please note that you should 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.
2. Create a script to modify the generated Makefile (see clause 5.2.4 for example). List all the DDF files you need when executing the AWK script.
3. Add script to modify the Makefile in the project properties.
4. 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 [7]. 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 (e.g. 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 value  
f\_DIAMETER\_Enc (in PDU\_DIAMETER pl\_pdu) octetstring;

f\_DIAMETER\_Dec (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, structure

The major parts of DPMG are:

1. AVP.sh sctipt: This is the front-end of the protocol module generator.
2. AVP.awk script: This is the most important part of the product.
3. 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.** |
| BaseTypes\_IETF\_RFC3588.ddf Base\_IETF\_RFC3588.ddf | [5] |
| CLCInterface\_Vodafone\_Rev2.ddf | [52] |
| ChargingApplications\_3GPP\_TS32299\_850.ddf | [26] |
| ChargingApplications\_3GPP\_TS32299\_870.ddf | [27] |
| ChargingApplications\_3GPP\_TS32299\_900.ddf | [28] |
| ChargingApplications\_3GPP\_TS32299\_940.ddf | [64] |
| ChargingApplications\_3GPP\_TS32299\_9b0.ddf | [65] |
| ChargingApplications\_3GPP\_TS32299\_a60.ddf | [65] |
| ChargingApplications\_3GPP\_TS32299\_c60.ddf | [104] |
| ChargingApplications\_3GPP\_TS32299\_d70.ddf |  |
| ChargingApplications\_3GPP\_TS32299\_be0.ddf | [106] |
| ChargingApplications\_3GPP\_TS32299\_d40.ddf | [105] |
| CreditControl\_IETF\_RFC4006.ddf | [8] |
| CxDxInterface\_3GPP\_TS29229\_6a0.ddf | [29] |
| CxDxInterface\_3GPP\_TS29229\_840.ddf | [30] |
| CxDxInterface\_3GPP\_TS29229\_880.ddf | [31] |
| CxDxInterface\_3GPP\_TS29229\_920.ddf | [64] |
| CxDxInterface\_3GPP\_TS29229\_c30.ddf | [105] |
| DigestAuthentication\_IETF\_RFC5090.ddf | [54] |
| GiInterface\_3GPP\_TS29061\_770.ddf | [34] |
| GiSGiInterface\_3GPP\_TS29061\_810.ddf | [35] |
| GiSGiInterface\_3GPP\_TS29061\_930.ddf | [63] |
| GiSGiInterface\_3GPP\_TS29061\_980.ddf | [36] |
| GiSGiInterface\_3GPP\_TS29061\_930\_QoS\_Detailed.ddf | [63] |
| GiSGiInterface\_3GPP\_TS29061\_b90.ddf | [107] |
| GiSGiInterface\_3GPP\_TS29061\_b90\_QoS\_Detailed.ddf | [107] |
| GiSGiInterface\_3GPP\_TS29061\_d70.ddf | [119] |
| GmbInterface\_3GPP\_TS29061\_6f0.ddf | [32] |
| GmbInterface\_3GPP\_TS29061\_720.ddf | [33] |
| GmbInterface\_3GPP\_TS29061\_810.ddf | [35] |
| GmbInterface\_3GPP\_TS29061\_930.ddf | [63] |
| GmbInterface\_3GPP\_TS29061\_970.ddf | [86] |
| GmbInterface\_3GPP\_TS29061\_980.ddf | [36] |
| GmbInterface\_3GPP\_TS29061\_d70.ddf | [119] |
| GqInterface\_PC\_3GPP\_TS29209\_670.ddf | [21] |
| GqInterface\_S3\_ETSI\_TS183017\_V231.ddf | [37] |
| GxInterface\_CRP\_3GPP\_TS29210\_670.ddf | [20] |
| GxInterface\_PCC\_3GPP\_TS29212\_740.ddf | [38] |
| GxInterface\_PCC\_3GPP\_TS29212\_820.ddf | [39] |
| GxInterface\_PCC\_3GPP\_TS29212\_830.ddf | [40] |
| GxInterface\_PCC\_3GPP\_TS29212\_840.ddf | [41] |
| GxInterface\_PCC\_3GPP\_TS29212\_910.ddf | [42] |
| GxInterface\_PCC\_3GPP\_TS29212\_930.ddf | [62] |
| GxInterface\_PCC\_3GPP\_TS29212\_970.ddf | [80] |
| GxInterface\_PCC\_3GPP\_TS29212\_9b0.ddf | [80] |
| GxInterface\_PCC\_3GPP\_TS29212\_d70.ddf | [114] |
| GxInterface\_PCC\_3GPP\_TS29212\_f10.ddf |  |
| NetworkAccessServer\_IETF\_RFC4005.ddf | [13] |
| RxInterface\_PCC\_3GPP\_TS29214\_830.ddf | [44] |
| RxInterface\_PCC\_3GPP\_TS29214\_990.ddf | [44] |
| RxInterface\_PCC\_3GPP\_TS29214\_a70.ddf | [94] |
| RxInterface\_PCC\_3GPP\_TS29214\_c10.ddf | [102] |
| RxInterface\_PCC\_3GPP\_TS29214\_f20.ddf |  |
| ShInterface\_3GPP\_TS29329\_620.ddf | [45] |
| ShInterface\_3GPP\_TS29329\_750.ddf | [46] |
| ShInterface\_3GPP\_TS29329\_820.ddf | [47] |
| ShInterface\_3GPP\_TS29329\_a30.ddf | [81] |
| ShInterface\_3GPP\_TS29329\_a50.ddf | [101] |
| SLgInterface\_3GPP\_TS29172\_d10.ddf | [112] |
| SLhInterface\_3GPP\_TS29173\_d00.ddf | [113] |
| Verizon\_Specific\_AVPs.ddf | [53] |
| e2Interface\_ETSI\_ES283035\_121.ddf | [55] |
| e4Interface\_ETSI\_ES283034\_220.ddf | [56] |
| TCOM\_Specific\_AVPs.ddf | [57][58] |
| a4Interface\_ETSI\_ES183066\_211.ddf | [59] |
| NGN\_NetworkAccesses\_ETSI\_ES183020\_111.ddf | [60] |
| a2Interface\_ETSI\_ES183059\_1\_211.ddf | [61] |
| AAAInterface\_3GPP\_TS29273\_840.ddf | [66][67] |
| AAAInterface\_3GPP\_TS29273\_940.ddf | [68][69] |
| AAAInterface\_3GPP\_TS29273\_b30.ddf | [92] |
| AAAInterface\_3GPP\_TS29273\_d60.ddf |  |
| AAAInterface\_3GPP\_TS29273\_f00.ddf | [124] |
| MobileIPv6\_NAS\_IETF\_RFC5447.ddf | [71] |
| MobileIPv6\_HA\_IETF\_RFC5778.ddf | [70] |
| MobileIPv4\_Application\_IETF\_RFC4004.ddf | [95] |
| GmbInterface\_3GPP\_TS29061\_930.ddf | [72] |
| Ericsson\_Specific\_AVPs.ddf | [73] |
| AAAInterface\_3GPP\_TS29272\_940.ddf | [74] |
| AAAInterface\_3GPP\_TS29272\_950.ddf | [74] |
| AAAInterface\_3GPP\_TS29272\_970.ddf | [78] |
| AAAInterface\_3GPP\_TS29272\_a30.ddf | [85] |
| AAAInterface\_3GPP\_TS29272\_a60.ddf | [91] |
| AAAInterface\_3GPP\_TS29272\_d70.ddf | [115] |
| AAAInterface\_3GPP\_TS29272\_f10.ddf | [123] |
| GxInterface\_PCC\_3GPP\_TS29212\_8a0.ddf | [76] |
| GxInterface\_PCC\_3GPP\_TS29212\_8b1.ddf | [77] |
| RqInterface\_ETSI\_ES283026\_241.ddf | [79] |
| Vimpelcom\_Specific.ddf |  |
| Vodafone\_Specific.ddf |  |
| ExtensibleAuthenticationProtocol\_IETF\_RFC4072.ddf | [83] |
| AAAInterface\_3GPP\_TS29273.ddf | [81] |
| WgInterface\_3GPP\_TS29234\_910.ddf | [82] |
| SGmbInterface\_3GPP\_TS29061\_980.ddf | [36] |
| SGmbInterface\_3GPP\_TS29061\_b90.ddf | [107] |
| SGmbInterface\_3GPP\_TS29061\_d70.ddf | [119] |
| GxaInterface\_3GPP2\_X\_S0057\_0\_300.ddf | [87] |
| Alcatel\_Lucent\_Specific\_AVPs.ddf | [89][99][100] |
| S9Interface\_3GPP\_TS29215\_b40.ddf | [90] |
| MobileIPv6\_HAAA\_IETF\_RFC5779.ddf | [93] |
| AAAInterface\_3GPP\_TS29272\_b60.ddf | [96] |
| GxInterface\_PCC\_3GPP\_TS29212\_aa0.ddf | [97] |
| SyInterface\_3GPP\_TS29219\_b30.ddf | [98] |
| Acision\_Specific.ddf |  |
| GxInterface\_PCC\_3GPP\_TS29212\_c52.ddf | [103] |
| DelegatedIPv6Prefix\_IETF\_RFC4818.ddf | [108] |
| AAAInterface\_3GPP\_TS29272\_bd0.ddf | [109] |
| SKT\_Specific\_AVPs.ddf | [110] |
| DiameterRoutingMessagePriority\_IETF\_RFC7944.ddf | [116] |
| GxInterface\_PCC\_3GPP\_TS29212\_e00.ddf | [117] |
| ChargingApplications\_3GPP\_TS32299\_d90.ddf | [118] |
| S6Interfaces\_3GPP\_TS29336\_f00.ddf | [120] |
| T6Interfaces\_3GPP\_TS29128\_f00.ddf | [121] |
| S6cInterface\_3GPP\_TS29338\_f00.ddf | [122] |
| SGdGddInterface\_3GPP\_TS29338\_f00.ddf | [122] |

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!

|  |  |
| --- | --- |
| **DDFs (in obsolete)** | **Refs.** |
| DiameterBaseAVPs.ddf  DiameterBaseTypes.ddf | [5] |
| 3GPPChargingApplicationAVPs.ddf | [19] |
| 3GPPCreditControlApplicationAVPs\_v6110.ddf | [11] |
| 3GPPCreditControlApplicationAVPs\_v670.ddf | [9] |
| 3GPPCreditControlApplicationAVPs\_v690.ddf | [10] |
| 3GPPShInterfaceAVPs\_v620.ddf | [45] |
| 3GPPShInterfaceAVPs\_v750.ddf | [46] |
| DiameterChargingApplicationAVPs.ddf | [19] |
| DiameterCreditControlApplicationAVPs.ddf | [8] |
| DiameterCreditControlApplicationAVPs\_aug05.ddf | [8] |
| DiameterMultimediaApplicationAVPs\_v770.ddf | [22] |
| DiameterMultimediaApplicationAVPs\_v810.ddf | [23] |
| DiameterNetworkAcessServerApplicationAVPs.ddf | [13] |
| DiameterOffLineCharging.ddf | [15] |
| EricssonChargingInterrogationProtocol\_to\_SDP\_IP.ddf | [10] |
| EricssonProprietaryCxDx.ddf | [48] |
| EricssonServiceChargingApplicationAVPs.ddf | [7] |
| GiSpecificAVPs.ddf | [34] |
| GmbSpecificAVPs.ddf | [33] |
| GqSpecificAVPs.ddf | [21] |
| GxSpecificAVPs.ddf | [20][38] |
| GyPlusSpecificAVPs.ddf | [14] |
| IMSSpecificAVPs.ddf | [18] |
| PsSpecificAVPs.ddf | [17] |
| SRAPSpecificAVPs.ddf | [16] |
| VodafoneSpecificAVPs.ddf | [24] |
| GxPlus\_Ericsson\_5\_1551\_AXB250\_10\_4RevF.ddf | [49] |
| GyPlus\_Ericsson\_6\_1551\_AXB250\_10\_4RevC.ddf | [50] |
| GyPlus\_Ericsson\_6\_1551\_AXB250\_10\_4RevK.ddf | [51] |
| CxDxInterface\_Ericsson\_1551\_FAY301\_0059\_PC26.ddf | [48] |

Please note, that 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 [8]

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 mustn’t be used together with the   
DiameterCreditControlApplication\_aug05.ddf!

#### DiameterNetworkAcessServerApplicationAVPs.ddf [13]

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 [18]

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 [19]

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 [20]

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 [20]

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 [21], [22]

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 [24]

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 [62]

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 [75]

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

* Trace-Depth (1462)

#### GmbInterface\_3GPP\_TS29061\_980.ddf [36]

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 chaged 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 [4]. 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.

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



Figure 2 Client mode mapping

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 (Figure 2). 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** |
| tsp\_hostname | charstring | Mandatory. Contains the IP address of the destination host in dot notation. |
| tsp\_portnumber | integer | Mandatory. Contains the port number of the destination host. |
| tsp\_reconnect | boolean | Optional, its default value is true. Enables reconnect mode . |
| tsp\_reconnect\_timeout | float | Optional, its default value is “2.0”. Specifies the time interval between two connection attempts in reconnect mode. |
| tsp\_connect\_timeout | 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 should not be used.

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

* use\_connection\_ASPs := "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 [10].

### Server mode



Figure 3 Server mode mapping

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 shall connect to the mapping component (Figure 3). 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** |
| tsp\_hostname | charstring | Mandatory. Contains the IP address of the listening interface in dot notation. |
| tsp\_portnumber | 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 that, 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**: Is the type for carrying notifications. The following fields are available:
  + *notification:* It is of type enumeartion 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 (see clause 2.3 for more details).

### 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

1. [2/198 17-CRL 113 200/6](https://github.com/eclipse/titan.core/blob/master/usrguide/referenceguide.doc) Uen  
   Programmer’s Technical Reference for TITAN TTCN–3 Test Executor
2. 109 21-CNL 113 462-53 Uen   
   DIAMETER Protocol Module Generator for TTCN-3 Toolset with TITAN, Product Revision Information
3. 109 21-CNL 113 310-9 Uen  
   DIAMETER Test Port for TTCN-3 Toolset with TITAN, PRI
4. RFC 3588  
   Diameter Base Protocol
5. The GNU Awk User's Guide, <http://www.gnu.org/software/gawk/manual/gawk.html>
6. 198 17-CNL 113 347 Uen  
   TCP Socket Test Port for TTCN-3 Toolset with TITAN, UG
7. RFC 4006  
   Diameter Credit-Control Application
8. 3GPP TS 32.299 v6.7.0  
   Diameter Charging Applications
9. 3GPP TS 32.299 v6.9.0  
   Diameter Charging Applications
10. 3GPP TS 32.299 v6.11.0  
    Diameter Charging Applications
11. 155 19-FAY 302 003/1 Uen Rev.PB1  
    Charging Interrogation Protocol
12. RFC 4005  
    Diameter Network Access Server Application
13. 6/1551-AXB 250 10/3 Uen Rev H   
    Gy + Interface Description
14. 2/155 19-CRA 119 0229/2 Uen Rev PB12  
    Off-line Charging in MTAS
15. 10/1551-AXB 250 10/3 Uen Rev J  
    SRAP Interface Description
16. 3GPP TS 32.251 V8.1.0  
    Packet Switched (PS) domain charging
17. 3GPP TS 32.260 V8.3.0  
    IP Multimedia Subsystem (IMS) charging
18. 3GPP TS 32.299 V8.2.0  
    Diameter charging applications
19. 3GPP TS 29.210 V6.7.0  
    Charging rule provisioning over Gx interface
20. 3GPP TS 29.209 V6.7.0  
    Policy control over Gq interface
21. 3GPP TS 29.229 V7.7.0  
    Cx and Dx interfaces based on the Diameter protocol; Protocol details
22. 3GPP TS 29.229 V8.1.0  
    Cx and Dx interfaces based on the Diameter protocol; Protocol details
23. Vodafone Gx+ Specification v1.3.1
24. 3GPP TS 29.210 V6.7.0  
    Charging rule provisioning over Gx interface
25. 3GPP TS 32.299 v8.5.0  
    Diameter Charging Applications
26. 3GPP TS 32.299 v8.7.0  
    Diameter Charging Applications
27. 3GPP TS 32.299 v9.0.0  
    Diameter Charging Applications
28. 3GPP TS 29.229 V6.15.0  
    Cx and Dx interfaces based on the Diameter protocol; Protocol details
29. 3GPP TS 29.229 V8.4.0  
    Cx and Dx interfaces based on the Diameter protocol; Protocol details
30. 3GPP TS 29.229 V8.8.0  
    Cx and Dx interfaces based on the Diameter protocol; Protocol details
31. 3GPP TS 29.061 V6.15.0  
    Interworking between the Public Land Mobile Network supporting packet based services and Packet Data Networks (PDN)
32. 3GPP TS 29.061 V7.2.0  
    Interworking between the Public Land Mobile Network supporting packet based services and Packet Data Networks (PDN)
33. 3GPP TS 29.061 V7.7.0  
    Interworking between the Public Land Mobile Network supporting packet based services and Packet Data Networks (PDN)
34. 3GPP TS 29.061 V8.1.0  
    Interworking between the Public Land Mobile Network supporting packet based services and Packet Data Networks (PDN)
35. 3GPP TS 29.061 V9.8.0  
    Interworking between the Public Land Mobile Network supporting packet based services and Packet Data Networks (PDN
36. 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)
37. 3GPP TS 29.212 V7.4.0  
    Policy and Charging Control over Gx reference point
38. 3GPP TS 29.212 V8.2.0  
    Policy and Charging Control over Gx reference point
39. 3GPP TS 29.212 V8.3.0  
    Policy and Charging Control over Gx reference point
40. 3GPP TS 29.212 V8.4.0  
    Policy and Charging Control over Gx reference point
41. 3GPP TS 29.212 V9.1.0  
    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 29.214 V8.3.0  
    Policy and Charging Control over Rx reference point
44. 3GPP TS 29.329 v6.2.0  
    Sh Interface based on the Diameter PROTOCOL
45. 3GPP TS 29.329 v7.5.0  
    Sh Interface based on the Diameter PROTOCOL
46. 3GPP TS 29.329 v8.2.0  
    Sh Interface based on the Diameter PROTOCOL
47. 1551-FAY 301 0059 Uen PC26  
    Service Contract, Diameter Cx Application
48. Ericsson 5/1551-AXB 250 10/4 Uen Rev F  
    Gx+ Interface Description
49. Ericsson 6/1551-AXB 250 10/4 Uen Rev C  
    Gy+ Interface Description
50. Ericsson 6/1551-AXB 250 10/4 Uen Rev K  
    Gy+ Interface Description
51. Intelligent Packet Core Vodafone Diameter CCA   
    Specification for the CLCI Version 2
52. Ericsson 31/155 19-FCP 111 391 Uen PC1 1  
    [Verizon LTE Rf Interface Specification](http://cdmweb.ericsson.se:7033/TeamCenter/controller/download_file_inline?file_name=Verizon+LTE+Rf+Interface+Specification.doc&file_handle_name=MTIObjectHandle-0002-1%7ER%7EtjviaDfprdwebKI------QYY%7EWordDoc%7EKI%7E%7E)
53. RFC 5090  
    RADIUS Extension for Digest Authentication
54. ETSI ES 283 035 v1.2.1 (2007-06)  
    TISPAN; NASS; e2 interface based on the DIAMETER protocol
55. ETSI ES 283 034 v2.2.0 (2008-05)  
    TISPAN; NASS; e4 interface based on the DIAMETER protocol
56. 10/155 19-HSD 101 96/1 Uen Rev G  
    SBG AF e2 Interface
57. 14/155 19-HSD 101 96/1 Uen Rev E  
    SBG AF e2 Extensions
58. 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
59. 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
60. 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
61. 3GPP TS 29.212 V9.3.0  
    Policy and Charging Control over Gx reference point
62. 3GPP TS 29.061 V9.3.0  
    Interworking between the Public Land Mobile Network (PLMN) supporting packet based services and Packet Data Networks (PDN)
63. 3GPP TS 32.299 V9.4.0  
    Diameter Charging Applications
64. 3GPP TS 32.299 V9.11.0  
    Diameter Charging Applications
65. 3GPP TS 29.229 V9.2.0  
    Cx and Dx interfaces based on the Diameter protocol; Protocol details
66. 3GPP TS 29.273 V8.4.0  
    Evolved Packet System (EPS); 3GPP EPS AAA interfaces
67. X/174 02-AXB 250 20 Uen PA2  
    Statement of Compliance  
    3GPP TS 29.273 3GPP EPS AAA interface
68. 3GPP TS 29.273 V9.4.0  
    Evolved Packet System (EPS); 3GPP EPS AAA interfaces
69. X/174 02-AXB 250 20 Uen PA4  
    Statement of Compliance  
    3GPP TS 29.273 3GPP EPS AAA interface
70. RFC5778  
    Diameter Mobile IPv6:  
    Support for Home Agent to Diameter Server Interaction
71. RFC5447  
    Diameter Mobile IPv6:  
    Support for Network Access Server to Diameter Server
72. 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)
73. Current Ericsson Diameter AVP Assignments Rev 1.111, 2010-01-19  
    <http://www.lmera.ericsson.se/~snmp/diameter-assignments.html>
74. 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)
75. 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)
76. 3GPP TS 29.212 V8.10.0 (2010-12)  
    Policy and Charging Control over Gx reference point
77. 3GPP TS 29.212 V8.11.1 (2011-03)  
    Policy and Charging Control over Gx reference point
78. 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)
79. 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
80. 3GPP TS 29.212 V9.7.0  
    Policy and Charging Control over Gx reference point
81. 3GPP TS 29.329 v10.3.0  
    Sh Interface based on the Diameter PROTOCOL
82. 3GPP TS 29.273 v9.4.0  
    Evolved Packet System (EPS), 3GPP EPS AAA interfaces
83. 3GPP TS 29.234 v9.1.0  
    3GPP system to Wireless Local Area Network (WLAN) interworking
84. RFC 4072  
    Diameter Extensible Authentication Protocol (EAP) Application
85. 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)
86. 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)
87. 3GPP2 X.S0057-0 Version 3.0  
    E*-UTRAN* – eHRPD Connectivity   
    and Interworking Core Network Aspects
88. 2/155 19-CRA 119 2107 Uen  
    Diameter Offline Charging in MTAS
89. 3GPP TS 29.212 V11.4.0  
    Policy and Charging Control (PCC) over Gx/Sd reference point   
    (Release 11)
90. 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)
91. 3GPP TS 29.273 v9.4.0  
    Evolved Packet System (EPS), 3GPP EPS AAA interfaces
92. RFC 5779  
    Diameter Proxy Mobile IPv6: Mobile Access Gateway and Local Mobility Anchor Interaction with Diameter Server
93. 3GPP TS 29.214 V10.7.0  
    Policy and Charging Control over Rx reference point
94. RFC 4004  
    Diameter Mobile IPv4 Application
95. 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)
96. 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)
97. 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)
98. 158/1594-FCP 101 8664 Uen  
    IS Verizon Ro Interface for Prompt and Collect in MTAS
99. 140/1594-FCP 101 8664 Uen  
    Ro interface enhancements based on Verizon's call flows and requirements
100. 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)
101. 3GPP TS 29.214 V12.1.0  
     Policy and Charging Control over Rx reference point
102. 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)
103. 3GPP TS 32.299 V12.6.0  
     Diameter Charging Applications
104. 3GPP TS 32.299 V13.4.0  
     Diameter Charging Applications
105. 3GPP TS 29.229 V12.3.0  
     Cx and Dx interfaces based on the Diameter protocol; Protocol details
106. 3GPP TS 32.299 V11.15.0  
     Diameter Charging Applications
107. 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)
108. RFC 4818  
     RADIUS Delegated-IPv6-Prefix Attribute
109. 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)
110. 109/0363-1/FCP111392-1  
     FA – WP SKT-Zone
111. 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)
112. 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)
113. 3GPP TS 32.212 V13.7.0  
     Policy and Charging Control (PCC); Reference points
114. 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
115. RFC 7944  
     Diameter Routing Message Priority
116. 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)
117. 3GPP TS 32.299 v13.9.0  
     Diameter Charging Applications
118. 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)
119. 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)
120. 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)
121. 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)
122. 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)
123. 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)