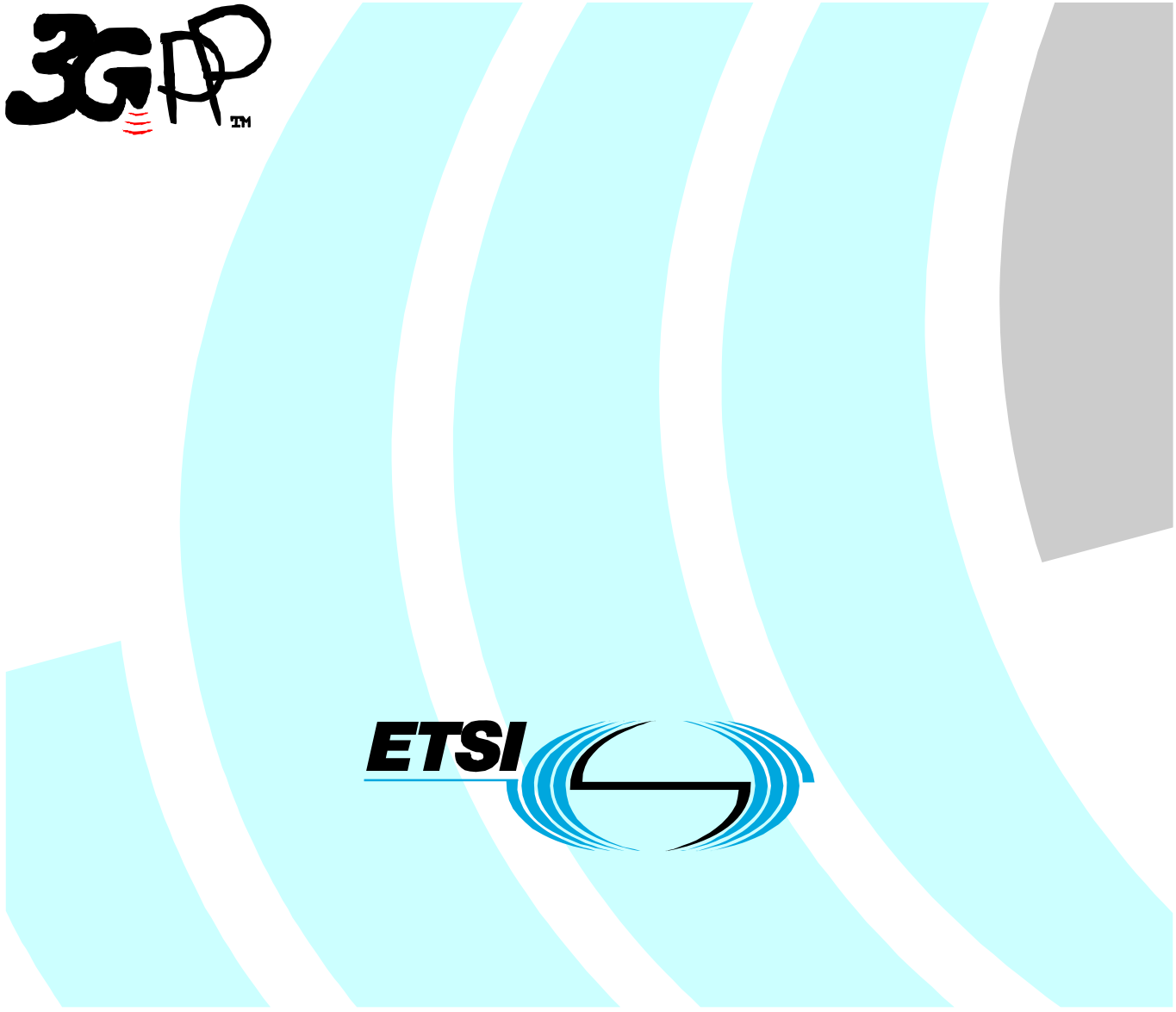
ETSI TS 134 229-3 V5.2.0 (2007-10)

Technical Specification

**Universal Mobile Telecommunications System (UMTS);
Internet Protocol (IP) multimedia call control protocol
based on Session Initiation Protocol (SIP) and
Session Description Protocol (SDP);
Part 3: Abstract test suite (ATS)
(3GPP TS 34.229-3 version 5.2.0 Release 5)**



Reference
RTS/TSGR-0534229-3v520

Keywords
UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is 3rd part of a multi-part conformance test specification for UE and is *valid for 3GPP Release 5*. The specification contains a TTCN design frame work and the detailed test specifications in TTCN for the UE conformance at the Gm reference point.

3GPP TS 34.229-1 [5] contains a conformance test description in prose.

3GPP TS 34.229-2 [6] contains a pro-forma for the UE Implementation Conformance Statement (ICS).

3GPP TS 34.229-3 the current document.

1 Scope

The present document specifies the protocol conformance testing in TTCN for the 3GPP User Equipment (UE) at the Gm interface.

The present document is the 3rd part of a multi-part test specification, 3GPP TS 34.229. The following TTCN test specification and design considerations can be found in the present document:

- the overall test suite structure;
- the testing architecture;
- the test methods and PCO definitions;
- the test configurations;
- the design principles, assumptions, and used interfaces to the TTCN tester (System Simulator);
- TTCN styles and conventions;
- the partial PIXIT proforma;
- the TTCNfiles for the mentioned protocols tests.

The Abstract Test Suites designed in the document are based on the test cases specified in prose (3GPP TS 34.229-1 [5]).

The present document is valid for UE implemented according 3GPP Release 5.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document in the same Release as the present document.
    - For a Release 5 UE, references to 3GPP documents are to version 5.x.y, when available.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 34.123-1: "User Equipment (UE) conformance specification; Part 1: Protocol conformance specification".

[3] 3GPP TS 34.123-2: "User Equipment (UE) conformance specification; Part 2: Implementation Conformance Statement (ICS) proforma specification".

[4] 3GPP TS 34.123-3: "User Equipment (UE) conformance specification; Part 3: Abstract Test Suites (ATS)".

[5] 3GPP TS 34.229-1: "Internet Protocol (IP) multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); User Equipment (UE) conformance specification; Part 1: Protocol conformance specification ".

[6] 3GPP TS 34.229-2: "Internet Protocol (IP) multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); User Equipment (UE) conformance specification; Part 2: Implementation Conformance Statement (ICS) proforma specification".

[7] 3GPP TS 34.108: "Common test environments for User Equipment (UE) conformance testing".

[8] ISO/IEC 9646-1: "Information technology - Open systems interconnection - Conformance testing methodology and framework - Part 1: General concepts".

[9] ISO/IEC 9646-7: "Information technology - Open systems interconnection - Conformance testing methodology and framework - Part 7: Implementation Conformance Statements".

[10] ETSI ETS 300 406 (1995): "Methods for testing and Specification (MTS); Protocol and profile conformance testing specifications; Standardization methodology".

[11] 3GPP TS 24.229: "IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3".

[12] ETSI ES 201 873: "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3".

[13] IETF RFC 3320: "Signalling Compression (SigComp)".

[14] IETF RFC 3485: "The Session Initiation Protocol (SIP) and Session Description Protocol (SDP) Static Dictionary for Signalling Compression (SigComp)".

[15] IETF RFC 3486: "Compressing the Session Initiation Protocol (SIP)".

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

[17] IETF RFC 4566: "SDP: Session Description Protocol".

[18] IETF RFC 1035: "Domain names - implementation and specification".

[19] IETF RFC 1533: "DHCP Options and BOOTP Vendor Extensions".

[20] IETF RFC 2131: "Dynamic Host Configuration Protocol".

[21] IETF RFC 3315: "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)".

[22] IETF RFC 3319: "Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers".

[23] IETF RFC 3361: "Dynamic Host Configuration Protocol (DHCP-for-IPv4) Option for Session Initiation Protocol (SIP) Servers".

[24] IETF RFC 3680: "A Session Initiation Protocol (SIP) Event Package for Registrations".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and 3GPP TS 34.229-1 [5] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and 3GPP TS 34.229-1 [5] apply.

4 Requirements on the TTCN development

A number of requirements are identified for the development and production of TTCN specification for 3GPP UE at the Gm reference point.

1. Top-down design, following 3GPP 34.229-1 [5], 3GPP TS 34.123-1 [2], 3GPP TS 34.108 [7].
2. A unique testing architecture and test method for testing all protocol layers of UE.
3. Uniform TTCN style and naming conventions.
4. Improve TTCN readability.
5. Using TTCN-3 (ES 201 873-1 [12]).
6. TTCN specification feasible, implementable and compilable.
7. Test cases shall be designed in a way for easily adaptable, upwards compatible with the evolution of the 3GPP core specifications and the future Releases.
8. The test declarations, data structures and data values shall be largely reusable.
9. Modularity and modular working method.
10. Minimizing the requirements of intelligence on the emulators of the lower testers.
11. Giving enough design freedom to the test equipment manufacturers.
12. Maximizing reuse of RFC BNF definitions from the relevant IETF core specifications.

In order to fulfil these requirements and to ensure the investment of the test equipment manufacturers having a stable testing architecture for a relatively long period, a unique testing architecture and test method are applied to the 3GPP UE protocol tests.

5 Test method and test model

5.1 Test method

5.2 IMS CC test model

The test model is shown in figure 2.

5.2.1 Ports interfacing to SS

In TTCN-3, ports are defined in all test components and in the Test System Interface. This is the equivalent of PCOs in TTCN-2. These ports then have to be mapped, or connected, to the SS at the start of each test.

5.2.1.1 Data ports

IMS_CC ATS in TTCN-3 simulates the SIP behaviour at the P_CSCF side. The scripts of SIP signalling in TTCN-3 communicate with the UE under test through four data ports and the emulations beneath. Each port shall be able to distinguish the use of one of the dual protocol stacks of IPv4 / IPv6.

The type of port (client or server) used to send or received a message will depend on the transport protocol selected for the testing, i.e. UDP or TCP.

- UDP case: The SS will send requests and responses to the UE from its client port. The SS will receive requests and responses from the UE on its server port.

- TCP case: The SS will receive requests from the UE and will send responses to those requests on its server port. The SS will send requests to the UE and will receive responses to those requests on its client port.

For requests originated in the UE, the transport protocol is selected by the UE. This information is extracted in the TTCN-3 and used in subsequent responses sent by the SS.

For requests originating in the SS, the UDP transport protocol is used.

If no security associations have been set up, the unprotected client and server ports will be used. The security ports shall be used by the TTCN-3 authors when a security association has been established.

5.2.1.2 Security Associations Setup

Four unidirectional SAs are established between the UE and the SS:

SA1: port_uc to port_ps
SA2: port_pc to port_us
SA3: port_ps to port_uc
SA4: port_us to port_pc

The first pair (SA1 and SA3) is for bidirectional traffic between port_uc and port_ps. The second pair (SA2 and SA4) is for bidirectional traffic between port_pc and port_us.

While TCP scenario will use all four SAs, in UDP, only two SAs are needed because there is no traffic from port_ps to port_uc nor from port_us to port_pc. Figure 1 shows one example of the use of ports and security association in UDP and TCP.

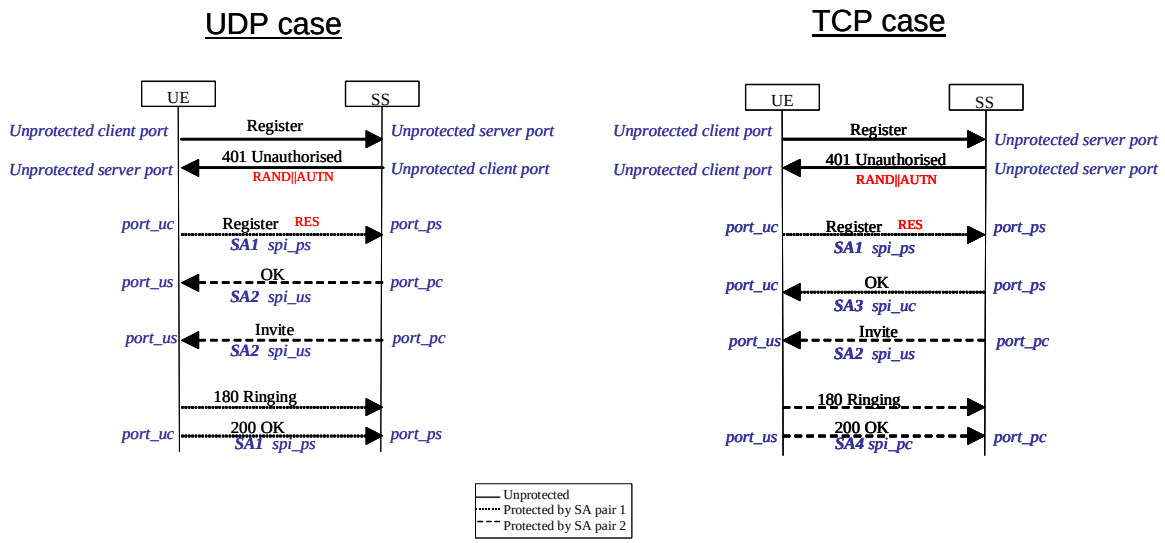


Figure 1: Use of port and SA in UDP and TCP

5.2.1.3 Control ports

IMS_CC ATS also controls the SS configuration and passes necessary parameters to the various emulation entities in the SS. This is done by ASPs through an **IP-CAN control port,** an **IP configuration port and a Signalling Compression control port**.

From the protocol stack point of view, SIP is an application layer protocol located above transport layer UDP which in turn uses the services provided by the IP/IPsec layer. The IP packages are transmitted via the connected IP-CAN bearer, the UTRAN bearer or the GERAN bearer. The emulations of these protocol layers in the SS shall be compliant with the relevant core specifications (3GPP and IETF).

The IP-CAN bearers are created, configured modified and released though the ASP at the IP-CAN control port. The TTCN-3 codes shall also be able to control the UDP/IP/IPsec configurations and provide necessary parameters through the control ASPs.

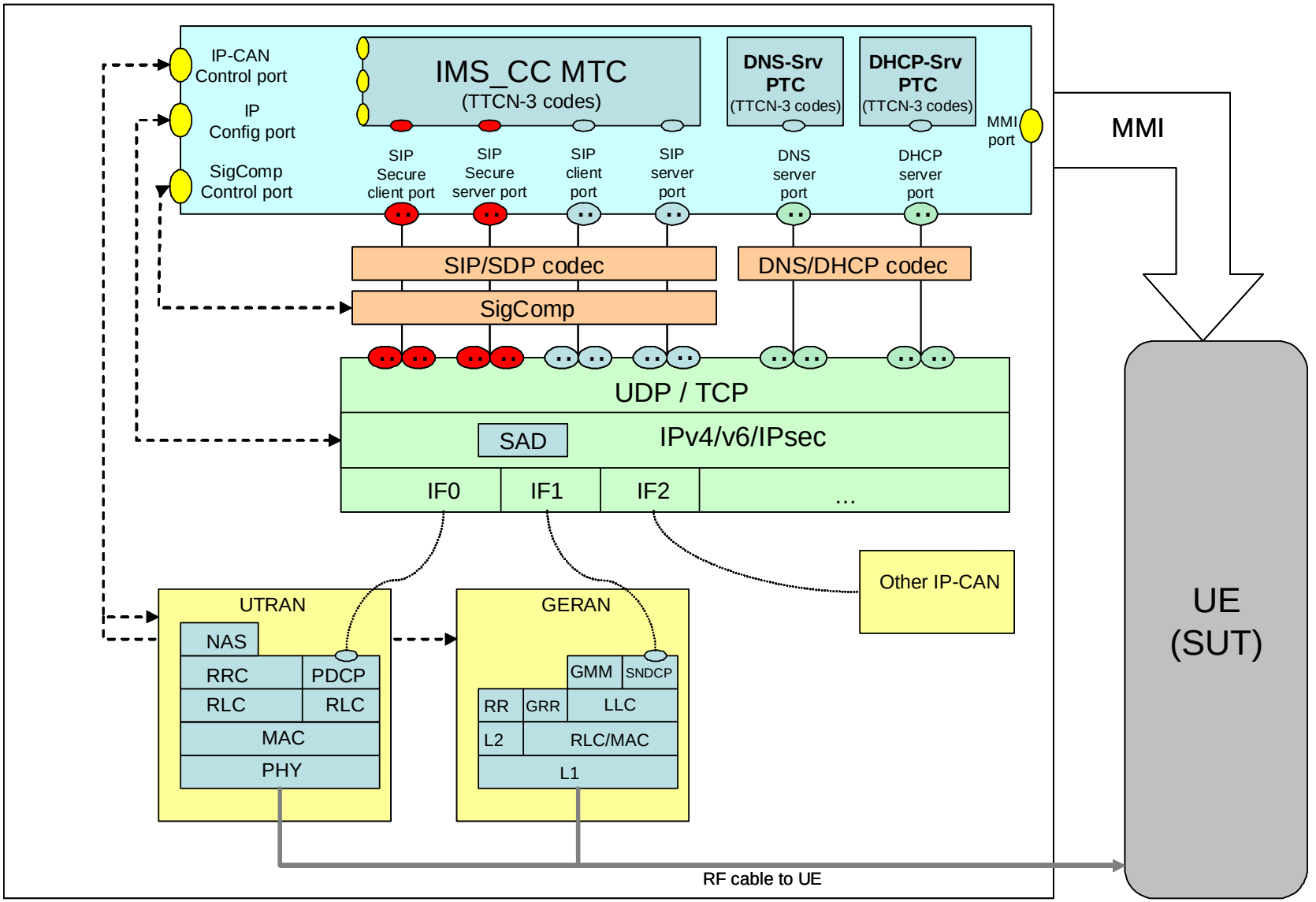


Figure 2: IMS CC test mode

5.2.2 SAD

Security Association Database (SAD) shall be made accessible by the IPsec entity and contain sets of parameters corresponding to each security association. During registration/authentication, the UE and the SS will negotiate these parameters for setting up a security association. As the negotiation is carried out on SIP level (through SIP message exchanges), the resulting security parameters are obtained and stored in IMS_CC ATS. A number of ASPs are defined to convey these parameters from TTCN-3 codes to SAD. ASPs manipulating the SAD are also defined.

5.2.3 Network interface

Similar to the majority of TCP/IP stack implementations, a network interface (IF0, IF1, IF2, etc.) structure is used to connect the IP-CAN bearer to IP protocol entity. When the ASP for setting up an IP-CAN bearer is called via the IP-CAN control port, the SS shall connect the established radio access bearer to the relevant IF structure, in order to provide the radio bearer connectivity to the IP/IPsec layer.

5.2.4 SigComp and related control port

SIP Compression is mandatory (clause 8 of 3GPP TS 24.229) and Signalling compression (RFC 3320, RFC 3485, RFC 3486) protocol is used for SIP compression. The SigComp entity in the model is used to carry out the compression/decompression functions. In the receiving direction of the SS, the SigComp entity will detect whether the incoming SIP message is compressed and, if so, decompress it. In the sending direction of the SS, the TTCN controls whether the outgoing SIP message is compressed through the SigComp control port. If while decompressing a message, decompression failure occurs, the message shall be discarded. The SigComp layer in the SS shall automatically find if a secure port or un-secure port is being used for transmission or reception of messages. If an un-secure port is used for transmission, then as per clause 8 of 3GPP TS 24.229, it shall not include state creation instructions. If the state creation command is received in a compressed message on an un-secured port (clause 8 of 3GPP TS 24.229), a decompression failure shall be generated.

5.2.5 SIP TTCN 3 Codec

SIP is a text-based protocol, the messages exchanged between the UE and the SS are character strings. In TTCN-3 ATS the messages are structured to take the advantage of TTCN-3 functionality, and to make the debugging and maintenance of the ATS easier. When the TTCN-3 ATS sends a message to the UE, the SIP TTCN-3 codec converts the structured message to the corresponding character string then transfers it to the UE. When the SS receives a message from the UE, the TTCN-3 codec converts the received character string to the structured message and passes it to the TTCN-3 ATS.

5.2.6 DHCP and DNS data ports

The DHCP port is used for receiving the DHCP requests from the UE under test, and sending corresponding responses to the UE. The DNS port is used for receiving domain name resolution requests from the UE and sending the results back to the UE. The TTCN which implements the required DHCP and DNS server functions (only the functions necessary for testing purposes, not full functionality) will receive and send on these ports.

The DHCP and DNS server functionalities in the default test configuration are implemented as Parallel Test Components (PTCs). For P-CSCF Discovery test cases (3GPP TS 34.229-1, clause 7), the PTCs are disabled and the DHCP and DNS ports are connected to the Main Test Component (MTC) so that the test script running on the MTC has full control of DHCP and DNS signalling.

5.3 Upper Tester (UT)

In order to support test automation and regression testing, an MMI port has been defined through which MMI commands (e.g. 'Please initiate a call') are sent to an external entity. Implementations can customize the external entity according to their needs. This port is enabled by setting PIXIT parameter px_TestAutomation to "true".

5.4 TTCN-3

TTCN is used as specification language. ES 201 873 [12] (TTCN-3) is applied to the notation.

6 ASP definitions

6.1 Control ASP

ASPs for configuring/controlling the SS are defined to operate in a pair of ASPs, Req (request) ASP and Cnf (Confirm) ASP of the blocking mode. The TTCN-3 execution after sending a Req ASP shall wait (be blocked) for the Cnf ASP.

Because the IMS Test Suite is radio access technology independent, few parameters are passed from the TTCN-3. Therefore the exact configuration procedures used are determined by the implementation.

The PIXIT px_RANTech (see below) is set by the operator and is passed through the TTCN to the SS. This is defined as an enumerated type and is used to specify which platform the test is to be run on (e.g. GERAN or UTRAN).

6.1.1 Cell Control

Name	CreateCellReq
Port	IPCANctl
Comment	ASP type for creating a cell

Parameter Name	Parameter Type	Comment
ranTech	RANTech	

Name	CreateCellCnf
Port	IPCANctl
Comment	ASP type which returns the result of the execution of CreateCellReq

Parameter Name	Parameter Type	Comment
status	Status	

Name	ReleaseCellReq
Port	IPCANctl
Comment	ASP type for releasing resources allocated to the cell

Parameter Name	Parameter Type	Comment

Name	ReleaseCellCnf
Port	IPCANctl
Comment	ASP type which returns the result of the execution of ReleaseCellReq

Parameter Name	Parameter Type	Comment
status	Status	

Name	RANTech
Type	enumerated
Parameters	GERAN, UTRAN_FDD, UTRAN_TDD, dummy1, dummy2
Comment	Indicates the radio access network technology used for transport of SIP signalling messages over the air interface

Name	Status
Type	enumerated
Parameters	success, failure, inconclusive
Comment	Indicates the status result of the requesting ASP

6.1.2 IdleUpdated

Name	IdleUpdatedReq
Port	IPCANctl
Comment	ASP type which requests the SS to bring the UE into an idle updated state and both GMM and/or MM registered

Parameter Name	Parameter Type	Comment

Name	IdleUpdatedCnf
Port	IPCANctl
Comment	ASP type which returns the result of the execution of IdleUpdatedReq

Parameter Name	Parameter Type	Comment
status	Status	

6.1.3 PDPContext

Name	ActivatePDPContextRequest_Req
Port	IPCANctl
Comment	ASP type which sets up a radio connection and waits for the Activate PDP Context Request and sends the Radio Bearer Setup message (if required). The ProtocolConfigurationOptions IE received in the ActivatePDPContextRequest is sent back in the Cnf. ActivatePDPContextAccept_Req must be called after this to complete the procedure

Parameter Name	Parameter Type	Comment
pdpContextId	integer	
bearerInfo	integer	

Name	ActivatePDPContextRequest_Cnf
Port	IPCANctl
Comment	ASP type which returns the result of the execution of ActivatePDPContextRequest_Req. The contents of the ProtocolConfigurationOptions IE received in the ActivatePDPContextRequest are included here

Parameter Name	Parameter Type	Comment
configOptList	ConfigOptList	
status	Status	

Name	ActivatePDPContextAccept_Req
Port	IPCANctl
Comment	ASP type which sends the Activate PDP Context Accept message with the ProtocolConfigurationOptions IE specified. ActivatePDPContextRequest_Req and Cnf must be called before this

Parameter Name	Parameter Type	Comment
pdpContextId	integer	
configOptList	ConfigOptList	

Name	ActivatePDPContextAccept_Cnf
Port	IPCANctl
Comment	ASP type which returns the result of the execution of ActivatePDPContextAccept_Req.

Parameter Name	Parameter Type	Comment
status	Status	

Name	ActivateSecondaryPDPContextReq
Port	IPCANctl
Comment	ASP type which informs the SS to expect the UE to request a secondary PDP context. Includes the bearer info to be configured for this secondary PDP context

Parameter Name	Parameter Type	Comment
pdpContextId	integer	
bearerInfo	integer	

Name	ActivateSecondaryPDPContextCnf
Port	IPCANctl
Comment	ASP type which returns the result of the execution of ActivateSecondaryPDPContextReq, when it is completed

Parameter Name	Parameter Type	Comment
status	Status	

Name	ModifyPDPContextReq
Port	IPCANctl
Comment	ASP type which informs the SS to expect the UE to request to modifiy an existing PDP context. Includes the bearer info for this to be modified to

Parameter Name	Parameter Type	Comment
pdpContextId	integer	
bearerInfo	integer	

Name	ModifyPDPContextCnf
Port	IPCANctl
Comment	ASP type which returns the result of the execution of ModifyPDPContextReq, when it is completed

Parameter Name	Parameter Type	Comment
status	Status	

Name	DeactivatePDPContextReq
Port	IPCANctl
Comment	ASP type which requests the SS deactivate the indicated PDP context. A value of pdpContextId = 0 indicates that all existing PDP contexts are to be deactivated.

Parameter Name	Parameter Type	Comment
pdpContextId	integer	
moInititiated	boolean	Flag indicating if the PDP context deactivation is initiated by the UE

Name	DeactivatePDPContextCnf
Port	IPCANctl
Comment	ASP type which returns the result of the execution of DeactivatePDPContextReq

Parameter Name	Parameter Type	Comment
status	Status	

Name	BearerInfo
Type	integer
Comment	References the RAB to be configured. This is RAN independent and can be added to/reduced as required

This is simply a list of RAB identifiers. It is expected, in the future, for these identifiers to equate to specific RAB requirements, for all available radio access technologies See clause 8.1 for more information.

Name	ConfigOptList
Type	set of ConfigOpt
Comment	Used to contain the protocol configuration options IE used in the PDP context messages

Name	ConfigOpt
Type	octetstring

Parameter Name	**Parameter Type**
ContainerId	octetstring [2]
ContainerLength	octetstring [1]
ContainerContents	octetstring optional

6.1.4 IP Configuration

Name	InstallKeyReq
Port	IPconf
Comment	ASP type which installs the keys into the IP layer in the SS

Parameter Name	**Parameter Type**	**Comment**
MD5_96Key	bitstring	length (128)
SHA_1_96Key	bitstring	length (160)
DES_EDE3_CBCKey	bitstring	length (192)
AES_CBCKey	bitstring	length (128)

Name	InstallKeyCnf
Port	IPconf
Comment	ASP type which returns the result of the execution of InstallKeyReq

Parameter Name	**Parameter Type**	**Comment**
status	Status	

Name	AssignIPaddrReq
Port	IPconf
Comment	ASP type which assigns the IP address to the IP layer in the SS

Parameter Name	**Parameter Type**	**Comment**
p_cscf_Addr	IPAddr	
dhcp_Addr	IPAddr	
dns_Addr	IPAddr	
ue_Addr	IPAddr	
peerUE_Addr	IPAddr	

Name	AssignIPaddrCnf
Port	IPconf
Comment	ASP type which returns the result of the execution of AssignIPaddrReq

Parameter Name	**Parameter Type**	**Comment**
status	Status	

Name	IPAddr
Type	charstring
Comment	in either colon separated or dotted decimal format

Name	ReleaseIPConfigurationReq
Port	IPconf
Comment	ASP type which releases the IMS IP layer configurations including Security Associations. This ASP is meant to be used when starting a new test case to make sure that the IP layer is in a well defined initial state irrespective of the execution of previous tests.

Parameter Name	**Parameter Type**	**Comment**
-	-	No parameters

Name	ReleaseIPConfigurationCnf
Port	IPconf
Comment	ASP type which returns the result of the execution of ReleaseIPConfigurationReq

Parameter Name	Parameter Type	Comment
status	Status	

Name	AddPCSCFaddrReq
Port	IPconf
Comment	ASP type which configures a new address of the P-CSCF component in the IP layer in the SS

Parameter Name	Parameter Type	Comment
p_cscf_Addr	IPAddr	New IP address of P-CSCF component to be simulated

Name	AddPCSCFaddrCnf
Port	IPconf
Comment	ASP type which returns the result of the execution of AddPCSCFaddrReq

Parameter Name	Parameter Type	Comment
status	Status	

Name	SignallingCompressionReq
Port	SigComp
Comment	ASP type which starts/stops signalling compression of messages

Parameter Name	Parameter Type	Comment
startCompression	boolean	

Name	SignallingCompressionCnf
Port	SigComp
Comment	ASP type which returns the result of the execution of SignallingCompressionReq

Parameter Name	Parameter Type	Comment
status	Status	

Name	RcvdCompartmentId
Port	SigComp
Comment	ASP type which feeds back the Compartment Id back to the Sigcomp layer, extracted from the last received message, used by SigComp layer to store any state appropriately.

Parameter Name	Parameter Type	Comment
compartmentId	charstring	Call-Id of the SIP message will be used as compartment Id

Name	GenerateSigCompDecompFailReq
Port	SigComp
Comment	ASP type which starts/stops inserting instructions resulting in decompression failure in compressed messages.

Parameter Name	Parameter Type	Comment
startError	boolean	TRUE: Start generation of errors. FALSE: Stop generation of errors.
mechanism	DecompFailureType	Optional: present when startError is TRUE, else absent

Name	GenerateSigCompDecompFailCnf
Port	SigComp
Comment	ASP type which returns the result of the execution of GenerateSigCompDecompFailReq

Parameter Name	Parameter Type	Comment
status	Status	

Name	DecompFailureType
Type	enumerated
Parameters	stateCreation,dummy1,dumm2,dummy3
Comment	Indicates the mechanism through which decompression failure errors shall be inserted during compressing message stateCreation: This type indicates, decompression failure shall be generated by inserting "State Creation" instructions in DL messages sent on unsecured SS Port (clause 8 of 3GPP TS 24.229)

6.1.5 SA Database

Name	SingleAddSADCnf
Port	IPconf
Comment	ASP type which returns the result of the execution of SingleAddSADReq

Parameter Name	Parameter Type	Comment
status	Status	

Name	DoubleAddSADReq
Port	IPconf
Comment	ASP type which sets two entries of SAD in the SS

Parameter Name	Parameter Type	Comment
sa1	SA	
sa2	SA	

Name	DoubleAddSADCnf
Port	IPconf
Comment	ASP type which returns the result of the execution of DoubleAddSADReq

Parameter Name	Parameter Type	Comment
status	Status	

Name	DelSADReq
Port	IPconf
Comment	ASP type which deletes the SAD entries

Parameter Name	Parameter Type	Comment
spi1	SPI	
spi2	SPI	optional
spi3	SPI	optional
spi4	SPI	optional
spi5	SPI	optional
spi6	SPI	optional
spi7	SPI	optional
spi8	SPI	optional
spi9	SPI	optional

Name	DelSADCnf
Port	IPconf
Comment	ASP type which returns the result of the execution of DelSADReq

Parameter Name	Parameter Type	Comment
status	Status	

Name	SA
Port	IPconf
Comment	ASP type which sets a single entry of parameters for a security association in the SS

Parameter Name	Parameter Type
spi	SPI
srcIPaddr	IPAddr
desIPaddr	IPAddr
srcUDPport	integer
desUDPport	integer
intAlgo	IntAlgo
ciphAlgo	CiphAlgo

Name	IntAlgo
Type	enumerated
Parameters	hmac_md5_96, hmac_sha_1_96
Comment	Integrity algorithms

Name	CiphAlgo
Type	enumerated
Parameters	des_ede3_cbc, aes_cbc, nociph
Comment	Ciphering algorithms, "nociph" means no ciphering

Name	SPI
Type	integer (0..4294967295)
Comment	security parameter index for IPsec

6.1.6 Emergency CS Call

Name	ExpectEmergencyCSCall
Port	IPCANctl
Comment	ASP type which informs the SS to expect the UE to request an emergency CS call

Parameter Name	Parameter Type	Comment

Name	EmergencyCSCallActive
Port	IPCANctl
Comment	ASP type which returns the result of the execution of ExpectEmergencyCSCall when it is in call active state

Parameter Name	Parameter Type	Comment
status	Status	

Name	ReleaseCSCallReq
Port	IPCANctl
Comment	ASP type which requests the SS to release the CS call previously established during ExpectEmergencyCSCall

Parameter Name	Parameter Type	Comment

Name	ReleaseCSCallCnf
Port	IPCANctl
Comment	ASP type which returns the result of the execution of ReleaseCSCallReq

Parameter Name	Parameter Type	Comment
status	Status	

6.2 IMS-CC Data ASP definitions

6.2.1 ASP_DataRequest

Name	ASP_DataRequest	
Port	DataPort	
Comment	ASP type for receiving/sending SIP Request Messages	
Parameter Name	**Parameter Type**	**Comment**
sigCompInfo	SigCompInfo	OPTIONAL. Information for/from SigComp layer. Absence means compression is/shall be not applied in received/send message.
portInfo	SSPortInfo	
msg	union {REGISTER_Request, INVITE_Request, OPTIONS_Request, BYE_Request, CANCEL_Request, ACK_Request, PRACK_Request, NOTIFY_Request, SUBSCRIBE_Request, PUBLISH_Request, UPDATE_Request, REFER_Request , MESSAGE_Request}	SIP message

6.2.2 ASP_DataResponse

Name	ASP_DataResponse	
Port	DataPort	
Comment	ASP type for receiving/sending SIP RESPONSE Message	
Parameter Name	**Parameter Type**	**Comment**
sigCompInfo	SigCompInfo	OPTIONAL. Information for/from SigComp layer. Absence means compression is/shall be not applied in received/send message.
portInfo	SSPortInfo	
msg	Response	SIP RESPONSE message

Name	SigCompInfo	
Type	Union	
Parameter Name	**Parameter Type**	**Comment**
compartmentId	charstring	Used for Sending messages from TTCN. To be used by SigComp Layer
isCompressed	Compressed	Used for received messages. If set, means received message was compressed

Name	Compressed
Type	record
Comment	Empty record used in SigCompInfo. Its presence means received message was compressed

Name	SSPortInfo	
Type	record	
Parameter Name	**Parameter Type**	**Comment**
ipAddr	IPAddr	IP address of simulated network node
transportProtocol	TransportProtocol	

Name	TransportProtocol
Type	enumerated
Parameters	UDP, TCP

6.3 Ut ASP definitions

Name	MMIMessage	
Port	MMIPort	
Comment	ASP type for sending messages to upper tester	
Parameter Name	**Parameter Type**	**Comment**
mmiMessage	charstring	Action required by upper tester

7 Codec definition

7.1 Introduction

SIP is a text-based protocol, thus the message exchange between the UE and the SS are pure character strings. In the TTCN-3 ATS the messages are structured and optimized to take the advantage of TTCN-3 functionality, and to make the debugging and maintenance of the ATS easier.

Every time the TTCN-3 ATS sends a message to the UE, the SIP TTCN-3 codec converts (encodes) the structured message given as a template to the corresponding character string before transferred to the UE.

When the SS receives a message from the UE, the TTCN-3 codec converts (decodes) the received character string to the structured message value and passes it to the TTCN-3 ATS.

7.2 TCI Interface Specification

TTCN-3 provides a reference test system implementation architecture in [ETSI ES 201 873-6] which is used here.

7.2.1 TCI - Required and Provided Interface Methods

A codec implementation for this ATS has to adhere to the TCI-CD provided and TCI-CD required interfaces as defined in ES 201 873-6, clause 7.3.2]. Within this context we recommend to use the TCI value interface ES 201 873-6, clause 7.2.2] with its several methods. In addition the codec has to follow the type mappings and instruction as defined in clause 8.3.

7.3 Requirements on abstract message syntax

7.3.1 Type definition - Syntax / Semantic aspects

All given defined BNF grammars (e.g. the ABNF of RFC 3261) are unique. Thus the syntax tree for each syntactically correct message derived with these grammars are unique too and the parts of a message can be uniquely identified (represented) by the terminal phrase belonging to a non terminal symbol and its derivation path in the syntax tree.

The syntax tree of all given messages can be used to uniquely identify and describe the parts of the messages. The leaves are the part of every message and the nodes from the root to the leaves represent the sequence of rules to be applied to derive that part

The IMS/SIP root message type is an ordered structured type, which is represented as a record type in TTCN-3. For each grammar rule of the ABNF a TTCN-3 record type is declared with the specific name of the rule. The following rules are applied to the fields within a record:

- A non-terminal symbol is declared as a record type for this symbol.
- The order of the symbols in the rule are represented by an equal order of the fields.
- Repetitions are declared as 'set of' or 'record of' types.
- Options are represented as optional record/set fields.
- Alternatives are declared as union types.

7.3.2 Deviations of the type definition semantic

- Most of the 'literals' of a message (for example: the string `"Via"` or `"v"` in the message header fields) are not represented.
- The TTCN-3 charstring type is used where we stop structuring even if the ABNF uses structured types. More details found in clause 8.3.3.
- Wherever possible parts are mapped to their best type representation, e.g. DIGIT based rules are mapped to integer type not to a charstring type.
- All of the following delimiters (including preceding or following whitespace) defined by the ABNF grammar to separate the parts of a message are not represented (see note).

```
STAR    =  SWS "*" SWS ; asterisk
SLASH   =  SWS "/" SWS ; slash
EQUAL   =  SWS "=" SWS ; equal
LPAREN  =  SWS "(" SWS ; left parenthesis
RPAREN  =  SWS ")" SWS ; right parenthesis
RAQUOT  =  ">" SWS ; right angle quote
LAQUOT  =  SWS "<"; left angle quote
COMMA   =  SWS "," SWS ; comma
SEMI    =  SWS ";" SWS ; semicolon
COLON   =  SWS ":" SWS ; colon
LDQUOT  =  SWS DQUOTE; open double quotation mark
RDQUOT  =  DQUOTE SWS ; close double quotation mark
HCOLON  =  *( SP / HTAB ) ":" SWS
SP      = single space
HTAB    = tab
SWS     = sep whitespace
```

NOTE: If they are present within a pure charstring they will be handled like a normal character and are still included.

- Messages which are not of interest to the test suite are left undecoded as a charstring and will not be further structured.

7.3.3 Additional requirements for codec implementations (SIP/IMS Message

The SIP/IMS codec is based on a normalized encoding which is always produced by an encoder. Decoder implementations, however, have to handle normalization before, or when constructing the structured message value, e.g. long versus compact form, whitespace compression, delimiter removal, same header grouping, etc. All these aspects will be handled in the next clause.

7.3.3.1 Differences between BNF - TTCN-3 Type Mapping

In normal cases the mapping is straight forward. Below you find the exceptions, including potential examples.

- The root message type is not a SIP-message but directly a Request or Response type which is represented as a TTCN-3 record. All Method - Message names (INVITE, BYE, ACK etc.) and all message header field names (To, From, CallID, CSeq, Via etc.) are mapped to an enumerated type in TTCN-3 to simplify the extension of new headers. During encoding, the long-form of these message header fields is always used. The respective field in the header type is restricted to values which are allowed.

BNF rules of RFC	TTCN-3 Type Mapping
SIP-message = Request / Response	type record REGISTER_Request {…}, type record INVITE_Request {…}, type record PRACK_Request {…}, type record NOTIFY_Request {…}, type record UPDATE_Request {…}, … type record Response {…}
Method = INVITEm / ACKm / OPTIONSm / BYEm / CANCELm / REGISTERm / …	type enumerated Method { ACK_E, BYE_E, CANCEL_E, INVITE_E, OPTIONS_E, REGISTER_E, …}

- The structure of the message header fields are mapped to a "set " type in TTCN-3, because the order of these header fields is not mandatory. There is an Unknown Header List given in the type system to decode unknown headers with ID and Value.

message-header = (	type **set** MessageHeader {
…	…
/ Contact	Contact contact optional,
/ Content-Disposition	ContentDisposition contentDisposition optional,
…	…
/ Via	Via via,
/ Warning	Warning warning optional,
/ WWW-Authenticate	WwwAuthenticate wwwAuthenticate optional,
/ extension-header) CRLF	UndefinedHeader_List undefinedHeader_List optional }

- The various parameter lists defined in the BNF are mapped and combined into three different TTCN-3 sets of generic-param types. These types differ only in their name: SemicolonParam_List, AmpersandParam_List, CommaParam_List to distinguish between the relevant separators.

uri-parameters = *(";" uri-parameter)	type set of GenericParam **SemicolonParam_List;**

Authentication-Info = "Authentication-Info" HCOLON ainfo
*(COMMA ainfo)
	type record AuthenticationInfo {
	FieldName fieldName(AUTHENTICATION_INFO_E),
	CommaParam_List ainfo
	}

ainfo = nextnonce / message-qop / response-auth / cnonce / nonce-count	type set of GenericParam **CommaParam_List**;
Headers = "?" header *("&" header)	type set of GenericParam **AmpersandParam_List**;

- Any more specific parameter rule (e.g. uri-param, user-param, lr-param , digest-cln, etc.) is simplified to the generic-param rule which will be mapped as a record structure of two charstrings (ID and paramValue). This is equivalent to a token with an optional generic value (token [EQUAL gen-value]).

```
digest-cln =        realm                          type record GenericParam {
                  / domain                           charstring id ,
                  / nonce                            charstring paramValue optional
                  / opaque                         }
                  / stale
                  / algorithm
                  / qop-options
                  / auth-param
```

- In addition to the pure charstring as a base type, the TTCN-3 type system provides base integer types which are unrestricted to the model e.g. the portField, CSeq number, maxForward digit.

```
user  =             1*( unreserved                 charstring
                        / escaped / user-unreserved
                      )
telephone-subscriber as defined in RC 2806
password =          *( unreserved                  charstring
                    / escaped
                    /"&"
                    / "="
                    / "+"
                    / "$"
                    / ","
                  )

Port =              1*DIGIT                        integer
Status-Code =     Informational                    integer
                  /   Redirection
                  /   Success
                  /   Client-Error
                  /   Server-Error
                  /   Global-Failure
                  /   extension-code
```

- Where the same header type can appear multiple times within a message, they will be decoded as a single header field, with multiple list elements. The order of appearance of the headers will be preserved within the header list value.

```
Contact =           ("Contact" / "m" ) HCOLON      type record Contact {
                        ( STAR / (contact-param      FieldName fieldName(CONTACT_E),
                        *(COMMA contact-param)        ContactBody contactBody
                    )                              }
                  )
contact-param =     (name-addr / addr-spec)         type record ContactAddress {
                  *(SEMI contact-params)             Addr_Union addressField,
                                                     SemicolonParam_List contactParams optional
                                                   }

                                                   type union ContactBody {
                                                    charstring wildcard,
                                                    ContactAddress_List contactAddresses
                                                   }

                                                   Used in

                                                   type set of ContactAddress ContactAddress_List;
```

- The BNF [clause 7.3.1 Header Field Format RFC 3261] specifies that several WWW or Proxy Authentication/Authorization headers should not be combined into a single header; however they will be decoded into such in the codec. If these need to be sent downlink then a new, 'raw' (pure charstring) message type will be introduced.

```
Authorization =      "Authorization" HCOLON          type record Authorization {
credentials                                            FieldName fieldName(AUTHORIZATION_E),
                                                       Credentials body
                                                     }
Credentials =        ("Digest" LWS digest-response)  type union Credentials {
                   / other-response                  CommaParam_List digestResponse,
                                                     OtherAuth otherResponse
                                                     }
```

- The different schemes (sip, sips, tel, fax, absoluteUri) in the SIP URI are all handled via the same type definition to simplify the decoding. This is because there is no difference between the URIs except the scheme.

```
Request-URI =    SIP-URI                          type record SipUrl  {
                 / SIPS-URI                         charstring scheme,
                 / absoluteURI                      UserInfo userInfo  optional,
                                                    HostPort hostPort,
with                                                SemicolonParam_List urlParameters optional,
                                                    AmpersandParam_List headers optional
SIP-URI  =       "sip:"                           }
                 [ userinfo ]
                 hostport
                 uri-parameters
                 [ headers ]

and

SIPS-URI =           "sips:"
                 [ userinfo ]
                 hostport
                 uri-parameters
                 [ headers ]

and

absoluteURI =    scheme ":" ( hier-part / opaque-part )
```

- Universal charstrings should be supported by the codec especially for the Display name in the URI.
- For downlink messages, if a message body is included, the TTCN will set the len field in the ContentLength header to the value -1. This value will be replaced by the codec with the actual length of the encoded message body (see clause 7.3.4)

7.3.4 Additional requirements for codec implementations (Message Body)

The message body is optional, but if it is included, will be encoded using either SDP or XML (see below).

The message body type consists of an optional charstring, containing the encoded message and a union of the different SDP and XML types.

```
type record MessageBody  {
  charstring encodedMsg optional,
  MsgBodyTypes msgBody
}

type union  MsgBodyTypes {
      reginfoElement regE,
      IMCN_Subsystem_XMLBody IMCNBody,
      SDP_Message sdpE
}
```

For uplink messages, if the received message contains a message body, the codec will provide the encoded charstring in encodedMsg and the decoded message in the appropriate choice of MsgBodyType.

For downlink messages, the charstring encodedMsg will always be set to omit. The codec will encode the msgBodyType according to the appropriate type definitions and will then include the length of the encoded message body in the content length header, replacing the value of -1 set in the TTCN.

7.3.5 Additional requirements for codec implementations (SDP Body)

The Session Description Protocol is defined in RFC 4566.

- The 'type' fields (such as 'v' and 'o' are not represented).
- For the defined attributes, the att-field is also not represented (e.g. 'curr' is not represented in SDP_attribute_curr).
- The Messages which are not of interest to a test suite are left undecoded as a charstring and will not be further structured.

7.3.5.1 Differences between BNF - SDP Type Mapping

In normal cases the mapping is straight forward. Below are the exceptions which differ.

- The numerical fields in the origin-field, the time-field and the timezone field have been defined as charstring because they may not fit into a 32-bit signed integer.

BNF Rules of RFC 4566

```
origin = username
         sess-id
         sess-version
         nettype
         addrtype
         unicast-address


time-fields = start-time
         stop-time
         repeat-fields
         [ zone-adjustments]
zone-adjustments = time
         typed-time
```

TTCN 3 Type Mapping

```
type record SDP_Origin {
                charstring username,
                charstring session_id,
                charstring session_version,
                charstring net_type,
                charstring addr_type,
                charstring addr
}
type record SDP_time_field {
                charstring start_time,
                charstring stop_time
}
type record SDP_timezone {
                charstring adjustment_time,
                SDP_typed_time offset
}
```

- The zone-adjustments field in the time-fields has been included as an additional field in the top-level message definition.

BNF Rules of RFC 4566	TTCN 3 Type Mapping
session-description = proto-version origin-field session-name-field information-field uri-field email-fields phone-fields connection-field bandwitdh-fields time-fields key-fields attribute-fields media-descriptions	type record SDP_Message { integer protocol_version, SDP_Origin origin, charstring session_name, charstring information optional, charstring uri optional, SDP_email_list emails optional, SDP_phone_list phone_numbers optional, SDP_connection connection optional, SDP_bandwidth_list bandwidth optional, SDP_time_list times, SDP_timezone_list timezone_adjustments optional, SDP_key key optional, SDP_attribute_list attributes optional, SDP_media_desc_list media_list optional

time-fields = start-time
stop-time
repeat-fields
[zone-adjustments]
	type record SDP_time {
	SDP_time_field time_field,
	SDP_repeat_list time_repeat optional
	}

- The mappings for the email-address, phone-number and connection-address fields have been simplified.

BNF Rules of RFC 4566	TTCN 3 Type Mapping

email-address = address-and-comment
/ dispname-and-address
/ addrspec
	type record SDP_contact {
	charstring addr_or_phone,
	charstring disp_name optional
	}

phone-number = email-safe
/ email-safe "<" phone ">"
/ phone
	type record SDP_contact {
	charstring addr_or_phone,
	charstring disp_name optional
	}

connection-address = multicast-address
/ unicast-address
	type record SDP_conn_addr {
	charstring addr,
	integer ttl optional,
	integer num_of_addr optional
	}

7.3.5.2 Defined attributes

The SDP_attribute type is defined as a union of the following attribute types. There is an unknown attribute given to decode undefined attributes with a name and value.

SDP Attribute	TTCN 3 Type Mapping

cat	type record SDP_attribute_cat {
	charstring attr_value
	}

charset	type record SDP_attribute_charset {
	charstring attr_value
	}

conf	type record SDP_attribute_curr {
	charstring preconditionType,
	charstring statusType,
	charstring direction
	}

curr	type record SDP_attribute_curr {
	charstring preconditionType,
	charstring statusType,
	charstring direction
	}

des	type record SDP_attribute_des { charstring preconditionType, charstring strength, charstring statusType, charstring direction

SDP Attribute	TTCN 3 Type Mapping
	}

fmtp	type record SDP_attribute_fmtp {
	charstring attr_value
	}

framerate	type record SDP_attribute_framerate {
	charstring attr_value
	}

inactive	type record SDP_attribute_inactive { }

keywds	type record SDP_attribute_keywds {
	charstring attr_value
	}

lang	type record SDP_attribute_lang {
	charstring attr_value
	}

orient	type record SDP_attribute_orient {
	charstring attr_value
	}

ptime	type record SDP_attribute_ptime {
	charstring attr_value
	}

quality	type record SDP_attribute_quality {
	charstring attr_value
	}

recvonly	type record SDP_attribute_recvonly { }

rtcp	type record SDP_attribute_rtcp {
	charstring attr_value
	}

rtpmap	type record SDP_attribute_rtpmap {
	charstring attr_value
	}

sdplang	type record SDP_attribute_sdplang {
	charstring attr_value
	}

sendrecv	type record SDP_attribute_sendrecv { }
sendonly	type record SDP_attribute_sendonly { }

tool	type record SDP_attribute_tool {
	charstring attr_value
	}

type	type record SDP_attribute_type {
	charstring attr_value
	}

unknown	type record SDP_attribute_tool {
	charstring name,
	charstring attr_value optional
	}

7.3.6 Additional requirements for codec implementations (DHCP/DNS)

The DHCP/DNS codec shall convert TTCN descriptions into/from octet streams as specified in the RFCs. The TTCN type definitions for DHCP/DNS types follow closely the data formats defined in the corresponding RFCs (RFC 1035, RFC 1533, RFC 2131, RFC 3315, RFC 3319 and RFC 3361).

The only special case to be considered is when a TTCN length field in a DHCP/DNS record is set to 0, in which case the encoder shall compute the proper length value during encoding. This agreement relieves the test case writer of complex length computations which are not relevant to the testcase.

7.3.7 Additional requirements for codec implementations (XML)

7.3.7.1 Registration Information

The used XML schema is taken directly from the RFC 3680.

The header taken from the XML Schema [RFC 3680, section 5.4] has to be generated in the Encoder automatically and will not be checked within the receive statement, thus it must not be decoded. This header is NOT declared in the type system definition in TTCN-3

```
<?xml version="1.0"?>
           <reginfo xmlns="urn:ietf:params:xml:ns:reginfo"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    version="0" state="full">
```

In normal cases the mapping is straight forward. All Sequences are defined as a set or record type. Examples of the Type Mapping are below:

XML Schema rule of RFC 3680

```
  <xs:element name="reginfo">
     <xs:complexType>
      <xs:sequence>
      <xs:attribute name="version"
type="xs:nonNegativeInteger"
  use="required"/>
      <xs:attribute name="state" use="required">
       <xs:simpleType>
      <xs:any namespace="##other"
processContents="lax" minOccurs="0"
  maxOccurs="unbounded"/>
      <xs:sequence>
       <xs:element ref="tns:registration"
minOccurs="0"
  maxOccurs="unbounded"/>
       <xs:any namespace="##other"
processContents="lax" minOccurs="0"
  maxOccurs="unbounded"/>
      </xs:sequence>
    <xs:element name="registration">
     <xs:complexType>
      <xs:sequence>
       <xs:element ref="tns:contact"
minOccurs="0" maxOccurs="unbounded"/>
       <xs:any namespace="##other"
processContents="lax" minOccurs="0"
  maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="aor" type="xs:anyURI"
use="required"/>
      <xs:attribute name="id" type="xs:string"
use="required"/>
      <xs:attribute name="state" use="required">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:enumeration value="init"/>
```

TTCN-3 Type Mapping

```
type set reginfoElement {
     reginfoSequence sequence,
     nonNegativeInteger version,
     reginfoAttribute state,
     Namespaces namespaces optional
}

type record reginfoSequence {
     Registrations registration,
     Any anyName optional
}

type set of registration Registrations;
type set registration {
     registrationSequence sequence,
     XSDAUX.anyURI aor,
     XSDAUX.string id,
     registration_stateAttribute state
}
```

```
        <xs:enumeration value="active"/>
        <xs:enumeration value="terminated"/>
       </xs:restriction>

      </xs:simpleType>
     </xs:attribute>
    </xs:complexType>
   </xs:element>
```

7.3.7.2 3GPP IM CN subsystem

The used XML schema is taken directly from 3GPP TS 24.229-1 [5], clause 7.6.

XML Schema rule of 3GPP TS 24.229, clause 7.6

```
<!ELEMENT ims-3gpp (
  alternative-service?, service-info?)>

<!ATTLIST ims-3gpp version CDATA #REQUIRED>
<!-- service-info element: The transparent data
received from HSS for AS -->
<!ELEMENT service-info        (#CDATA)>

<!-- alternative-service: alternative-service
used in emergency sessions -->
<!ELEMENT alternative-service (
   type, reason)>

<!ELEMENT type          (emergency)>
<!ELEMENT reason        (#PCDATA)>
```

TTCN-3 Type Mapping

```
type record IMCN_Subsystem_XMLBody {
    AlternativeService alternativeService
optional,
    charstring serviceInfo optional,
    integer version
}

type record AlternativeService {
    charstring typeName ("emergency"),
    charstring reason
}
```

7.4 Textual Codec Requirements (Details)

7.4.1 Encoder

The encoding is straight forward. The TCI Interface method **encode(in Value value)** which returns the TriMessageType from the provided (TciCDProvided) must be implemented. Selection of the relevant String field name should be used to generate an ASCII Byte Stream which provides the complete message.

Some hints for the implementation:

- Value interface:
  - The usage of the TCI Value API is recommended here.
- Whitespace/delimiter handling:
  - Should be included by the Encoder. There is no information given in the type system about whitespaces and delimiter.
- Long vs. Compact format:
  - Only the long format must be supported for the message header name.

7.4.2 Decoder

For the decoder the TCI Interface method **Value decode (in TriMessageType message, in Type hyp)** which returns the message Value must be implemented. Within this operation a parser must be instantiated which constructs the structured message values from the text message.

Some hints for the implementation:

- Value interface:
  - The usage of the TCI Value API is recommended here.
- Whitespace/delimiter handling:
  - Should be ignored by the Decoder. Just the values without spaces and delimiters should be handled by the decoder and represented in a template structure afterwards.
- Different formats:
  - The decoder must be able to handle all header codings, e.g. v, VIA, via, vIa, etc.
  - The long and the short format must be supported for the message header name.
- Error handling:
  - All errors should be logged in addition to the TTCN-3 logging. If the message is not decodable it should return NULL, as specified in the TCI standard.

8 Design consideration

8.1 Bearer Configurations for IMS Testing

8.1.1 Bearer Information for UTRAN

BearerInfo	RANTech = UTRAN_FDD	Description
1	34.108, clause 6.10.2.4.1.56	To be used for IMS Signalling only
2	34.108, clause 6.10.2.4.6.6	Not supported in Rel-5
3	34.108, clause 6.10.2.4.6.7	Not supported in Rel-5
4	25.993, clause 7.1.122	Only supported in Rel-5
5	25.993, clause 7.1.124	Not supported in Rel-5

8.1.2 Bearer Information for GERAN

No specific bearer information has yet been defined. The QoS to be used is therefore dependant on the media applications supported by the UE.

8.2 Security

TBD.

8.3 Test Suite Operations

Table 1: TSO definitions

TSO Name	Description
o_Bitstring2Base64	**Type of the result:** charstring **Parameters:** bitstring p_Bitstring **Description** Returns the Base 64 encoded value of p_Bitstring
o_GetItemFromCommaList	**Type of the result:** charstring **Parameters:** charstring p_CommaList, integer p_ItemIndex, integer p_NumberOfItems **Description** To get item number p_ItemIndex from a list of items separated by commas. The returned item must not have any white spaces at the beginning Used with PIXIT for MT call test case
o_IPv4Addr2Octetstring	**Type of the result:** octetstring **Parameters:** IPAddr ipAddr **Description** converts an IPv4 Address (in dotted separated decimal text format) into an octetstring (32-bit address, according to RFC 1035, section 3.4.1)
o_IPv6Addr2Octetstring	**Type of the result:** octetstring **Parameters:** IPAddr ipAddr **Description** converts an IPv6 Address (in text format, colon separated hexadecimal format) into an octetstring (128-bit address, according to RFC 3513)
o_isIPv4Addr	**Type of the result:** boolean **Parameters:** IPAddr ipAddr **Description** checks whether the IP Address in text format (dotted separated decimal) corresponds to an IPv4 address
o_isIPv6Addr	**Type of the result:** boolean **Parameters:** IPAddr ipAddr **Description** checks that the IP Address in text format (colon separated hexadecimal format) corresponds to an IPv6 address
o_MD5	**Type of the result:** charstring **Parameters:** charstring p_Data **Description** calculates the MD5 Message-Digest Algorithm according to RFC 1321
o_PutInLowercase	**Type of the result:** charstring **Parameters:** charstring par_string **Description** returns the equivalent string in lower case

8.4 AT commands

No AT commands have yet been defined for IMS operations

Annex A (normative): Abstract Test Suites (ATS)

This annex contains the approved ATSs.

The ATSs have been produced using the Testing and Test Control Notation version 3 (TTCN3) according to ES 201 873 [12].

A.1 Version of specifications

Table A.1 shows the version of the test specifications which the delivered ATSs are referred to.

Table A.1: Versions of the test and Core specifications

Core specifications	3GPP TS 24.229 [11]
Test specifications	3GPP TS 34.229-1 [5] 3GPP TS 34.229-2 [6] 3GPP TS 34.123-3 [2]

A.2 IMS-CC ATS

Table A.2: IMS-CC TTCN test cases

Test case	Description
8.6	Initial registration for combined IMS security and early IMS security against a network with early IMS support only

The ATS is contained in an ASCII file (IMS_CC.ttcn) which accompanies the present document.

A.2.3 Optional IP-CAN TTCN 2++ interface

FFS.

Annex B (normative): Partial IXIT proforma

B.0 Introduction

This partial IXIT proforma contained in the present document is provided for completion, when the related Abstract Test Suite is to be used against the Implementation Under Test (IUT).

Text in *italics* is comments for guidance for the production of a IXIT, and is not to be included in the actual IXIT.

The completed partial IXIT will normally be used in conjunction with the completed ICS, as it adds precision to the information provided by the ICS.

B.1 Parameter values

Table B.1: PIXIT

Parameter name	Description	Type	Default value	Supported value
px_AssociatedTelUri	Arbitrary TEL URI for the user	charstring	tel:+358-555-1234567	
px_AuthAMF	Authentication Management Field (16 bits).	bitstring (16)	'0000000000000000'B	The value shall be different from '1111 1111 1111 1111'B (AMFresynch)
px_AuthK	Authentication Key (128 bits)	bitstring (128)	'01011110010010101011001101100011000100100010011011101101010010101011011101000000100010111001100111110000110000100110100011000101001'B	
px_AuthN	Length of Extended value min 31, max 127 (TS 34.108 cl. 8.1.2)	integer	127	
px_AuthRAND	Authentication / Random challenge (128 bits)	bitstring (128)	'01010101…01'B	
px_BearerInfo1	Initial Bearer to be used	integer	1	
px_BearerInfo2	Bearer to be used for Secondary PDP Context	integer	2	
px_CalleeUri	URI of Callee, send in INVITE	charstring	"sip:User-B@3gpp.org"	
px_CalleeContactUri	URI to be used to contact Callee	charstring	"sip:User-B@3gpp.org"	
px_CellId	Utran cell Id	charstring	"0010100010000001'	See TS 24.229 clause 7.2A.4.3
px_CiphAlgo_Def	Ciphering Algorithm	CiphAlgo	nociph	enumerated type: des_ede3_cbc, aes_cbc or nociph
px_DHCPServer_IPAddr	IP address of DHCP server (in v4 or v6 format)	IPAddr	"10.122.11.33"	
px_DNS_DomainName	DNS server fully qualified domain name	charstring	"dnsserver.3gpp.	

Parameter name	Description	Type	Default value	Supported value
	(FQDN)		org"	
px_DNSServer_IPAddr	IP address of DNS server (in v4 or v6 format)	IPAddr	"10.122.11.33"	
px_HomeDomainName	Home Domain Name when using ISIM or the home domain name derived from px_IMSI when using USIM (preceded by "sip:")	charstring	"sip:3gpp.org"	
px_InviteToTag	Value of the tag in the To header related to Invite	charstring	"abc-InviteToTag"	
px_IPSecAlgorithm	Integrity Algorithm	IntAlgo	hmac_md5_96	enumerated type; hmac_md5_96, hmac_sha_1_96
px_Opaque	String of data, specified by the server, which should be returned by the client unchanged in the Authorization header of subsequent requests with URIs in the same protection space.	charstring	"5ccc069c403eb af9f0171e9517f4 0e41"	
px_P_CSCF_DomainName	P-CSCF fully qualified domain name (FQDN)	charstring	"pcscf.3gpp.org"	
px_P_CSCF_DomainName _2	Additional P-CSCF FQDN (Full Qualified Domain Name) for special tests	charstring	"pcscf2.3gpp.org "	
px_P_CSCF_DomainName _3	Additional P-CSCF FQDN (Full Qualified Domain Name) for special tests	charstring	"pcscf3.3gpp.org "	
px_P_CSCF_IPAddr	IP address of P-CSCF (in v4 or v6 format)	IPAddr	"10.122.11.33"	
px_P_CSCF_IPAddr_2	Additional P-CSCF IPaddress for special tests (in v4 or v6 format)	IPAddr	"10.122.11.34"	
px_P_CSCF_IPAddr_3	Additional P-CSCF IPaddress for special tests (in v4 or v6 format)	IPAddr	"10.122.11.35"	
px_Pcscf	P-CSCF fully qualified domain name that resolves to the IP address of SS	charstring	"pcscf.3gpp.org"	
px_PeerUE_IPAddr	IP address of peer UE (in v4 or v6 format)	IPAddr	"10.122.11.55	
px_Port_pc	Protected Client port at the SS (simulated P-CSCF)	integer	5061	
px_Port_ps	Protected Server port at the SS (simulated P-CSCF)	integer	5062	
px_Port_ps_NoSec	Unprotected Server port at the SS (simulated P-CSCF)	integer	5060	
px_Private_UserId	Private User Identity when using ISIM or private user identity derived from px_IMSI when using USIM or SIM	charstring	"privateuser@3gpp.org"	
px_Public_UserId	Public User Identity when using ISIM or public user identity derived from px_IMSI when using USIM or SIM	charstring	"sip:localuser@3gpp.org"	
px_RANTech	RAN Technology	RANTech	UTRAN_FDD	enumerated type: GERAN, UTRAN_FDD or UTRAN_TDD
px_RegisterExpiration	Value (in seconds) of the 'expires' parameter in the Contact header	charstring	"600"	
px_RSeqNumFor183	Value in the RSeq header in 183 Session in Progress (value between 1 and 2**32 – 1)	integer	1	
px_Scscf	S-CSCF fully qualified domain name that does not resolve to the IP address of SS	charstring	"scscf@3gpp.org"	
px_SS_SipUri	SIP URI with IP Address or FQDN of SS (simulated P-CSCF)	charstring	"sip:pcscf.3gpp.org"	
px_ToTagRegister	Value of the tag in the To header	charstring	"abc-ToTag"	
px_ToTagSubscribeDialog	Value of the tag in the To header related to Subscribe	charstring	"abc-SubscribeToTag	

Parameter name	Description	Type	Default value	Supported value
			''	
px_UE_IPAddr	IP address assigned to UE (in v4 or v6 format)	IPAddr	"10.122.11.145"	
px_UE_SipUri	SIP URI with IP Address or FQDN of UE		'sip: 10.122.11.145'	
px_UeWithSIM	UE has a SIM inserted	boolean	false	
px_TestAutomation	If set, MMI commands are sent to the MMI port instead to a pop-up window	boolean	false	

B.1.1 SDP parameters for MT call test case

This clause contains parameters to describe one to three media that the SS will propose to the UE in the INVITE Request. This information shall be compatible with the UE's capabilities.

Table B.2: SDP parameters for MT call

Parameter name	Description	Type	Default value	Supported value
px_NumberOfMedia	Number of media description	integer	1	1, 2, 3
For each media description, the following parameters shall be supplied:				
px_Media	Media type	charstring	'audio'	audio, video, text, application, message
px_MediaPort	Transport port to which the media stream is sent	integer	49230	Integer within the range 49152 - 65535
px_Proto	Transport protocol	charstring	'RTP/AVP'	UDP, RTP/AVP, RTP/SAVP, TCP, RTP/AVPF, TCP/TLS
px_FmtNumber	Number of Media format description	integer	3	
px_FmtValues	Value of each media format description (in a comma separated list)	charstring	'96, 97, 98'	
px_Bandwidth	Bandwidth value for b=AS (only if RTP/RTCP is used)	integer	75	
px_RS_Bandwidth	Bandwidth value for b=RS (only if RTP/RTCP is used)	integer	75	
px_RR_Bandwidth	Bandwidth value for b=RR (only if RTP/RTCP is used)	integer	75	
px_AttribNumber	Number of attribute ("a=") lines (excluding 'curr' and 'des' lines)	integer	4	
px_AttribValues	Value of each of the attribute lines, excluding 'curr' and 'des' lines (in a comma separated list).	charstring	'rtpmap:96 L8/8000, rtpmap:97 L16/8000, rtpmap:98 L16/11025/2, maxptime:80'	
px_LocalDir	Direction tag for desired local resource	charstring	'sendrecv'	sendrecv, send, recv
px_RemoteDir	Direction tag for desired remote resource	charstring	'sendrecv'	sendrecv, send, recv

B.2 MMI questions

Table B.3 requests additional information needed for the execution of the MMI commands used in the ATS.

Table B.3: MMI questions

Required information for MMI question
Please REGISTER
Please make a Call
Please release the Call
Please switch off the UE
Please switch on the UE
Please configure UE to initiate a Dedicated PDP Context
Please configure UE to initiate P-CSCF Discovery via PCO
Please configure UE to initiate P-CSCF Discovery via DHCP
Please de-REGISTER

Annex C (informative): Additional information to IXIT

Additional information may be provided when completing the IXIT questions listed in annex A.

C.1 Identification Summary

Table C.1 is completed by the test laboratory. The item "Contract References" is optional.

Table C.1: Identification Summary

IXIT Reference Number	
Test Laboratory Name	
Date of Issue	
Issued to (name of client)	
Contract References	

C.2 Abstract Test Suite Summary

In table C.2 the test laboratory provides the version number of the protocol specification and the version number of ATS which are used in the conformance testing.

Table C.2: ATS Summary

Protocol Specification	3GPP TS 24.229
Version of Protocol Specification	
Test Specification in prose	3GPP TS 34.229-1
Version of TSS & TP Specification	
ATS Specification	3GPP TS 34.229-3
Version of ATS Specification	
Abstract Test Method	Distributed Test Method

C.3 Test Laboratory

C.3.1 Test Laboratory Identification

The test laboratory provides the following information.

Table C.3: Test Laboratory Identification

Name of Test Laboratory	
Postal Address	
Office address	
e-mail address	
Telephone Number	
FAX Number	

C.3.2 Accreditation status of the test service

The test laboratory provides the following information.

Table C.4: Accreditation status of the test service

Accreditation status	
Accreditation Reference	

C.3.3 Manager of Test Laboratory

The test laboratory provides the information about the manager of test laboratory in table C.5.

Table C.5: Manager of Test Laboratory

Name of Manager of Test Laboratory	
e-mail address	
Telephone Number	
FAX Number	
E-mail Address	

C.3.4 Contact person of Test Laboratory

The test laboratory provides the information about the contact person of test laboratory in table C.6.

Table C.6: Contact person of Test Laboratory

Name of Contact of Test Laboratory	
e-mail address	
Telephone Number	
FAX Number	
E-mail Address	

C.3.5 Means of Testing

In table C.7, the test laboratory provides a statement of conformance of the Means Of Testing (MOT) to the reference standardized ATS, and identifies all restrictions for the test execution required by the MOT beyond those stated in the reference standardized ATS.

Table C.7: Means of Testing

Means of Testing

C.3.6 Instructions for Completion

In table C.8, the test laboratory provides any specific instructions necessary for completion and return of the proforma from the client.

Table C.8: Instruction for Completion

Instructions for Completion

C.4 Client

C.4.1 Client Identification

The client provides the identification in table C.9.

Table C.9: Client Identification

Name of Client	
Postal Address	
Office Address	
Telephone Number	
FAX Number	

C.4.2 Client Test Manager

In table C.10 the client provides information about the test manager.

Table C.10: Client Test Manager

Name of Client Test Manager	
Telephone Number	
FAX Number	
E-mail Address	

C.4.3 Client Contact person

In table C.11 the client provides information about the test contact person.

Table C.11: Client Contact person

Name of Client contact person	
Telephone Number	
FAX Number	
E-mail Address	

C.4.4 Test Facilities Required

In table C.12, the client records the particular facilities required for testing, if a range of facilities is provided by the test laboratory.

Table C.12: Test Facilities Required

Test Facilities Required

C.5 System Under Test

C.5.1 SUT Information

The client provides information about the SUT in table C.13.

Table C.13: SUT Information

System Name	
System Version	
SCS Reference	
Machine Configuration	
Operating System Identification	
IUT Identification	
ICS Reference for the IUT	

C.5.2 Limitations of the SUT

In table C.14, the client provides information explaining if any of the abstract tests cannot be executed.

Table C.14: Limitation of the SUT

Limitations of the SUT

C.5.3 Environmental Conditions

In table C.15 the client provides information about any tighter environmental conditions for the correct operation of the SUT.

Table C.15: Environmental Conditions

Environmental Conditions

C.6 Ancillary Protocols

This clause is completed by the client in conjunction with the test laboratory.

In the following tables, the client identifies relevant information concerning each ancillary protocol in the SUT other than the IUT itself. One table for one ancillary protocol.

Based on the MOT the test laboratory should create question proforma for each ancillary protocol in the blank space following each table. The information required is dependent on the MOT and the SUT, and covers all the addressing, parameter values, timer values and facilities (relevant to ENs) as defined by the ICS for the ancillary protocol.

C.6.1 Ancillary Protocols 1

Table C.16: Ancillary Protocol 1

Protocol Name	
Version number	
ICS Reference (optional)	
IXIT Reference (optional)	
PCTR Reference (optional)	

C.6.2 Ancillary Protocols 2

Table C.17: Ancillary Protocol 2

Protocol Name	
Version number	
ICS Reference (optional)	
IXIT Reference (optional)	
PCTR Reference (optional)	

Annex D (informative): PCTR Proforma

PROTOCOL

Conformance Test Report

(PCTR)

Universal Mobile Telecommunication System, UMTS,
User Equipment-Network Access

Layer 3 Signalling Functions

Test Candidate	
Name :	SUT name
Model :	model
H/W version :	hw
S/W version :	sw
Serial No. :	serienr

Client	
Name :	
Street / No. :	
Postal Code / City:	
Country :	

Annex E (informative): TTCN3 style guide for 3GPP IMS ATS

E.1 General rules for 3GPP ATSs

A detailed guide on 3GPP ATS style can be found in TS 34.123-3 [4], Annex E. Although the guidelines in TS 34.123-3 [4] were written for TTCN-2 ATSs, most of them are applicable to ATSs written in TTCN-3 and were considered when designing the IMS ATS in TTCN-3 whenever it was possible.

E.2 3GPP IMS ATS implementation guidelines

The content of this clause is specific for IMS ATS written in TTCN-3.

E.2.1 Grouping of similar objects

In order to aid readability and to add logical structure to the test suite, definitions shall be collected in named groups by using the TTCN-3 keyword '`group`'.

EXAMPLE: Header field types are grouped under the group name HeaderFieldTypes.

```
group HeaderFieldTypes
{
    type record Accept {
        FieldName fieldName(ACCEPT_E),
        AcceptBody_List acceptArgs optional
    }

    type record AcceptEncoding {
        FieldName fieldName(ACCEPT_ENCODING_E),
        ContentCoding_List contentCoding optional
    }
….
}
```

E.2.2 'Visible' test case description

A short description of the test cases shall be made available for System Simulator manufacturers use. This shall be done by using the keywords '`with`' and '`extension`' at the end of the test case.

Please Note: This field is for information purposes in the SS only

EXAMPLE: Test case description for test case 8.1.

```
testcase TC_8_1() runs on IMSComponent system SystemInterfaces
      {
        TGuard.start;
        ts_InitPorts ();
        v_Default := activate(ts_DefaultDef());
        ts_ConfigureIPAddr (px_SS_IPAddr);
        ts_Preamble(px_BearerInfo, px_SS_IPAddr, px_UE_IPAddr);
        ts_Register_Authentication();
        ts_Register_SubscribeNotify ();
        ts_Postamble(px_BearerInfo);
      }
        with {extension " Description: Test to verify that the UE can correctly register to IMS
services when equipped with UICC that contains either both ISIM and USIM applications or only USIM
application but not ISIM.  The process consists of sending initial registration to S-SCSCF via the
P-CSCF discovered, authenticating the user and finally subscribing the registration event package
for the registered default public user identity."}// end testcase TC_8_1
```

E.2.3 Naming conventions

The following prefixes shall be used when creating new objects in TTCN-3.

Table E.1: Prefixes used for TTCN-3 objects

TTCN object	Case of first character	Prefix	Comment
TTCN Module	Upper	IMS_CC	
External function	Upper	o_	Note 1
Function parameters	Upper	p_	
Functions	Upper	ts_	
Test Case Selection Expression			
Constant	Upper	<NAME IN CAPITALS>	
Variable	Upper	v_	Note 2
General Variable	Upper	gv_	Note 3
Port Types	Upper	-	
Port Names	Lower	-	
Timer	Upper	T	
General templates	Upper	t_	
Templates for Config ASPs	Upper	c_	
Templates for Header types	Upper	h[r|s]_	Note 4
Templates for Method types	Upper	m[b|r|s]	Note 4
Templates for XML types	Upper	x[r|s]_	Note 4
Templates for SDP types	Upper	d[b|r|s]_	
Test Suite Parameter (PICS)	Upper	pc_	
Test Suite Parameter (PIXIT)	Upper	px_	
Test Case	Upper	TC_	Note 5

NOTE 1: External functions are the equivalent to test suite operations in TTCN-2.
NOTE 2: These are local variables, only visible in the functions where they are defined.
NOTE 3: General variables are those defined within the TTCN-3 components. They are visible to all the functions run in the component.
NOTE 4: Prefix for templates can be followed by the following indicators:
- 'b' shall be included in base templates. Normally, templates without the 'b' indicator are modified templates from a parent (or base) template.
- 'r' shall be present to indicate that the object is only used in receive statements (i.e. the template may contain wildcards).
- 's' shall be present to indicate that the object is only used in send statements.

NOTE 5: Test case names will correspond to the clause in the prose that specifies the test purpose. E.g. TC_8_1. An additional digit may be specified if more than one test case is used to achieve the test purpose. If an additional digit is required, this probably means that the test prose are not well defined.

Annex F (informative): BNF Message Definitions

This is a list of all the BNF definitions required for the ATS, compiled from all necessary RFCs.

F.1 RFC 3261

```
25  Augmented BNF for the SIP Protocol

   All of the mechanisms specified in this document are described in
   both prose and an augmented Backus-Naur Form (BNF) defined in RFC
   2234 [10].  Section 6.1 of RFC 2234 defines a set of core rules that
   are used by this specification, and not repeated here.  Implementers
   need to be familiar with the notation and content of RFC 2234 in
   order to understand this specification.  Certain basic rules are in
   uppercase, such as SP, LWS, HTAB, CRLF, DIGIT, ALPHA, etc.  Angle
   brackets are used within definitions to clarify the use of rule
   names.

   The use of square brackets is redundant syntactically.  It is used as
   a semantic hint that the specific parameter is optional to use.

25.1 Basic Rules

   The following rules are used throughout this specification to
   describe basic parsing constructs.  The US-ASCII coded character set
   is defined by ANSI X3.4-1986.

      alphanum  =  ALPHA / DIGIT

   Several rules are incorporated from RFC 2396 [5] but are updated to
   make them compliant with RFC 2234 [10].  These include:

      reserved    =  ";" / "/" / "?" / ":" / "@" / "&" / "=" / "+"
                     / "$" / ","
      unreserved  =  alphanum / mark
      mark        =  "-" / "_" / "." / "!" / "~" / "*" / "'"
                     / "(" / ")"
      escaped     =  "%" HEXDIG HEXDIG

   SIP header field values can be folded onto multiple lines if the
   continuation line begins with a space or horizontal tab.  All linear
   white space, including folding, has the same semantics as SP.  A
   recipient MAY replace any linear white space with a single SP before
   interpreting the field value or forwarding the message downstream.
   This is intended to behave exactly as HTTP/1.1 as described in RFC
   2616 [8].  The SWS construct is used when linear white space is
   optional, generally between tokens and separators.

      LWS  =  [*WSP CRLF] 1*WSP ; linear whitespace
      SWS  =  [LWS] ; sep whitespace

   To separate the header name from the rest of value, a colon is used,
   which, by the above rule, allows whitespace before, but no line
   break, and whitespace after, including a linebreak.  The HCOLON
   defines this construct.

      HCOLON  =  *( SP / HTAB ) ":" SWS

   The TEXT-UTF8 rule is only used for descriptive field contents and
   values that are not intended to be interpreted by the message parser.
   Words of *TEXT-UTF8 contain characters from the UTF-8 charset (RFC
   2279 [7]).  The TEXT-UTF8-TRIM rule is used for descriptive field
   contents that are n t quoted strings, where leading and trailing LWS
   is not meaningful.  In this regard, SIP differs from HTTP, which uses
   the ISO 8859-1 character set.

      TEXT-UTF8-TRIM  =  1*TEXT-UTF8char *(*LWS TEXT-UTF8char)
      TEXT-UTF8char   =  %x21-7E / UTF8-NONASCII
      UTF8-NONASCII   =  %xC0-DF 1UTF8-CONT
```

```
                    /  %xE0-EF 2UTF8-CONT
                    /  %xF0-F7 3UTF8-CONT
                    /  %xF8-Fb 4UTF8-CONT
                    /  %xFC-FD 5UTF8-CONT
   UTF8-CONT        =  %x80-BF
```

A CRLF is allowed in the definition of TEXT-UTF8-TRIM only as part of a header field continuation. It is expected that the folding LWS will be replaced with a single SP before interpretation of the TEXT-UTF8-TRIM value.

Hexadecimal numeric characters are used in several protocol elements. Some elements (authentication) force hex alphas to be lower case.

```
   LHEX  =  DIGIT / %x61-66 ;lowercase a-f
```

Many SIP header field values consist of words separated by LWS or special characters. Unless otherwise stated, tokens are case-insensitive. These special characters MUST be in a quoted string to be used within a parameter value. The word construct is used in Call-ID to allow most separators to be used.

```
   token       =  1*(alphanum / "-" / "." / "!" / "%" / "*"
                  / "_" / "+" / "`" / "'" / "~" )
   separators  =  "(" / ")" / "<" / ">" / "@" /
                  "," / ";" / ":" / "\" / DQUOTE /
                  "/" / "[" / "]" / "?" / "=" /
                  "{" / "}" / SP / HTAB
   word        =  1*(alphanum / "-" / "." / "!" / "%" / "*" /
                  "_" / "+" / "`" / "'" / "~" /
                  "(" / ")" / "<" / ">" /
                  ":" / "\" / DQUOTE /
                  "/" / "[" / "]" / "?" /
                  "{" / "}" )
```

When tokens are used or separators are used between elements, whitespace is often allowed before or after these characters:

```
   STAR    =  SWS "*" SWS ; asterisk
   SLASH   =  SWS "/" SWS ; slash
   EQUAL   =  SWS "=" SWS ; equal
   LPAREN  =  SWS "(" SWS ; left parenthesis
   RPAREN  =  SWS ")" SWS ; right parenthesis
   RAQUOT  =  ">" SWS ; right angle quote
   LAQUOT  =  SWS "<"; left angle quote
   COMMA   =  SWS "," SWS ; comma
   SEMI    =  SWS ";" SWS ; semicolon
   COLON   =  SWS ":" SWS ; colon
   LDQUOT  =  SWS DQUOTE; open double quotation mark
   RDQUOT  =  DQUOTE SWS ; close double quotation mark
```

Comments can be included in some SIP header fields by surrounding the comment text with parentheses. Comments are only allowed in fields containing "comment" as part of their field value definition. In all other fields, parentheses are considered part of the field value.

```
   comment  =  LPAREN *(ctext / quoted-pair / comment) RPAREN
   ctext    =  %x21-27 / %x2A-5B / %x5D-7E / UTF8-NONASCII
               / LWS
```

ctext includes all chars except left and right parens and backslash. A string of text is parsed as a single word if it is quoted using double-quote marks. In quoted strings, quotation marks (") and backslashes (\) need to be escaped.

```
   quoted-string  =  SWS DQUOTE *(qdtext / quoted-pair ) DQUOTE
   qdtext         =  LWS / %x21 / %x23-5B / %x5D-7E
                     / UTF8-NONASCII
```

The backslash character ("\") MAY be used as a single-character quoting mechanism only within quoted-string and comment constructs. Unlike HTTP/1.1, the characters CR and LF cannot be escaped by this mechanism to avoid conflict with line folding and header separation.

```
quoted-pair  =  "\" (%x00-09 / %x0B-0C
                / %x0E-7F)

SIP-URI          =  "sip:" [ userinfo ] hostport
```

```
                    uri-parameters [ headers ]
SIPS-URI         =  "sips:" [ userinfo ] hostport
                    uri-parameters [ headers ]
userinfo         =  ( user / telephone-subscriber ) [ ":" password ] "@"
user             =  1*( unreserved / escaped / user-unreserved )
user-unreserved  =  "&" / "=" / "+" / "$" / "," / ";" / "?" / "/"
password         =  *( unreserved / escaped /
                    "&" / "=" / "+" / "$" / "," )
hostport         =  host [ ":" port ]
host             =  hostname / IPv4address / IPv6reference
hostname         =  *( domainlabel "." ) toplabel [ "." ]
domainlabel      =  alphanum
                    / alphanum *( alphanum / "-" ) alphanum
toplabel         =  ALPHA / ALPHA *( alphanum / "-" ) alphanum

IPv4address    =  1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT
IPv6reference  =  "[" IPv6address "]"
IPv6address    =  hexpart [ ":" IPv4address ]
hexpart        =  hexseq / hexseq "::" [ hexseq ] / "::" [ hexseq ]
hexseq         =  hex4 *( ":" hex4)
hex4           =  1*4HEXDIG
port           =  1*DIGIT
```

The BNF for telephone-subscriber can be found in RFC 2806 [9]. Note, however, that any characters allowed there that are not allowed in the user part of the SIP URI MUST be escaped.

```
uri-parameters    =  *( ";" uri-parameter)
uri-parameter     =  transport-param / user-param / method-param
                     / ttl-param / maddr-param / lr-param / other-param
transport-param   =  "transport="
                     ( "udp" / "tcp" / "sctp" / "tls"
                     / other-transport)
other-transport   =  token
user-param        =  "user=" ( "phone" / "ip" / other-user)
other-user        =  token
method-param      =  "method=" Method
ttl-param         =  "ttl=" ttl
maddr-param       =  "maddr=" host
lr-param          =  "lr"
other-param       =  pname [ "=" pvalue ]
pname             =  1*paramchar
pvalue            =  1*paramchar
paramchar         =  param-unreserved / unreserved / escaped
param-unreserved  =  "[" / "]" / "/" / ":" / "&" / "+" / "$"

headers         =  "?" header *( "&" header )
header          =  hname "=" hvalue
hname           =  1*( hnv-unreserved / unreserved / escaped )
hvalue          =  *( hnv-unreserved / unreserved / escaped )
hnv-unreserved  =  "[" / "]" / "/" / "?" / ":" / "+" / "$"

SIP-message    =  Request / Response
Request        =  Request-Line
                  *( message-header )
                  CRLF
                  [ message-body ]
Request-Line   =  Method SP Request-URI SP SIP-Version CRLF
Request-URI    =  SIP-URI / SIPS-URI / absoluteURI
absoluteURI    =  scheme ":" ( hier-part / opaque-part )
hier-part      =  ( net-path / abs-path ) [ "?" query ]
net-path       =  "//" authority [ abs-path ]
abs-path       =  "/" path-segments

opaque-part    =  uric-no-slash *uric
uric           =  reserved / unreserved / escaped
uric-no-slash  =  unreserved / escaped / ";" / "?" / ":" / "@"
                  / "&" / "=" / "+" / "$" / ","
path-segments  =  segment *( "/" segment )
segment        =  *pchar *( ";" param )
param          =  *pchar
pchar          =  unreserved / escaped /
                  ":" / "@" / "&" / "=" / "+" / "$" / ","
scheme         =  ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )
authority      =  srvr / reg-name
srvr           =  [ [ userinfo "@" ] hostport ]
reg-name       =  1*( unreserved / escaped / "$" / ","
                  / ";" / ":" / "@" / "&" / "=" / "+" )
```

```
query          =  *uric
SIP-Version    =  "SIP" "/" 1*DIGIT "." 1*DIGIT

message-header  =  (Accept
                /  Accept-Encoding
                /  Accept-Language
                /  Alert-Info
                /  Allow
                /  Authentication-Info
                /  Authorization
                /  Call-ID
                /  Call-Info
                /  Contact
                /  Content-Disposition
                /  Content-Encoding
                /  Content-Language
                /  Content-Length
                /  Content-Type
                /  CSeq
                /  Date
                /  Error-Info
                /  Expires
                /  From
                /  In-Reply-To
                /  Max-Forwards
                /  MIME-Version
                /  Min-Expires
                /  Organization
                /  Priority
                /  Proxy-Authenticate
                /  Proxy-Authorization
                /  Proxy-Require
                /  Record-Route
                /  Reply-To
                /  Require
                /  Retry-After
                /  Route
                /  Server
                /  Subject
                /  Supported
                /  Timestamp
                /  To
                /  Unsupported
                /  User-Agent
                /  Via
                /  Warning
                /  WWW-Authenticate
                /  extension-header) CRLF

INVITEm           =  %x49.4E.56.49.54.45 ; INVITE in caps
ACKm              =  %x41.43.4B ; ACK in caps
OPTIONSm          =  %x4F.50.54.49.4F.4E.53 ; OPTIONS in caps
BYEm              =  %x42.59.45 ; BYE in caps
CANCELm           =  %x43.41.4E.43.45.4C ; CANCEL in caps
REGISTERm         =  %x52.45.47.49.53.54.45.52 ; REGISTER in caps
Method            =  INVITEm / ACKm / OPTIONSm / BYEm
                     / CANCELm / REGISTERm
                     / extension-method
extension-method  =  token
Response          =  Status-Line
                     *( message-header )
                     CRLF
                     [ message-body ]

Status-Line     =  SIP-Version SP Status-Code SP Reason-Phrase CRLF
Status-Code     =  Informational
                /  Redirection
                /  Success
                /  Client-Error
                /  Server-Error
                /  Global-Failure
                /  extension-code
extension-code  =  3DIGIT
Reason-Phrase   =  *(reserved / unreserved / escaped
                   / UTF8-NONASCII / UTF8-CONT / SP / HTAB)

Informational  =  "100"  ;  Trying
               /  "180"  ;  Ringing
```

```
              /   "181"  ;  Call Is Being Forwarded
              /   "182"  ;  Queued
              /   "183"  ;  Session Progress

Success  =  "200"  ;  OK

Redirection  =  "300"  ;  Multiple Choices
             /   "301"  ;  Moved Permanently
             /   "302"  ;  Moved Temporarily
             /   "305"  ;  Use Proxy
             /   "380"  ;  Alternative Service

Client-Error  =  "400"  ;  Bad Request
              /   "401"  ;  Unauthorized
              /   "402"  ;  Payment Required
              /   "403"  ;  Forbidden
              /   "404"  ;  Not Found
              /   "405"  ;  Method Not Allowed
              /   "406"  ;  Not Acceptable
              /   "407"  ;  Proxy Authentication Required
              /   "408"  ;  Request Timeout
              /   "410"  ;  Gone
              /   "413"  ;  Request Entity Too Large
              /   "414"  ;  Request-URI Too Large
              /   "415"  ;  Unsupported Media Type
              /   "416"  ;  Unsupported URI Scheme
              /   "420"  ;  Bad Extension
              /   "421"  ;  Extension Required
              /   "423"  ;  Interval Too Brief
              /   "480"  ;  Temporarily not available
              /   "481"  ;  Call Leg/Transaction Does Not Exist
              /   "482"  ;  Loop Detected
              /   "483"  ;  Too Many Hops
              /   "484"  ;  Address Incomplete
              /   "485"  ;  Ambiguous
              /   "486"  ;  Busy Here
              /   "487"  ;  Request Terminated
              /   "488"  ;  Not Acceptable Here
              /   "491"  ;  Request Pending
              /   "493"  ;  Undecipherable

Server-Error  =  "500"  ;  Internal Server Error
              /   "501"  ;  Not Implemented
              /   "502"  ;  Bad Gateway
              /   "503"  ;  Service Unavailable
              /   "504"  ;  Server Time-out
              /   "505"  ;  SIP Version not supported
              /   "513"  ;  Message Too Large

Global-Failure  =  "600"  ;  Busy Everywhere
                /   "603"  ;  Decline
                /   "604"  ;  Does not exist anywhere
                /   "606"  ;  Not Acceptable

Accept         =  "Accept" HCOLON
                   [ accept-range *(COMMA accept-range) ]
accept-range   =  media-range *(SEMI accept-param)
media-range    =  ( "*/*"
                  / ( m-type SLASH "*" )
                  / ( m-type SLASH m-subtype )
                  ) *( SEMI m-parameter )
accept-param   =  ("q" EQUAL qvalue) / generic-param
qvalue         =  ( "0" [ "." 0*3DIGIT ] )
                  / ( "1" [ "." 0*3("0") ] )
generic-param  =  token [ EQUAL gen-value ]
gen-value      =  token / host / quoted-string

Accept-Encoding  =  "Accept-Encoding" HCOLON
                    [ encoding *(COMMA encoding) ]
encoding         =  codings *(SEMI accept-param)
codings          =  content-coding / "*"
content-coding   =  token

Accept-Language  =  "Accept-Language" HCOLON
                    [ language *(COMMA language) ]
language         =  language-range *(SEMI accept-param)
language-range   =  ( ( 1*8ALPHA *( "-" 1*8ALPHA ) ) / "*" )
```

```
Alert-Info   =  "Alert-Info" HCOLON alert-param *(COMMA alert-param)
alert-param  =  LAQUOT absoluteURI RAQUOT *( SEMI generic-param )

Allow  =  "Allow" HCOLON [Method *(COMMA Method)]

Authorization     =  "Authorization" HCOLON credentials
credentials       =  ("Digest" LWS digest-response)
                     / other-response
digest-response   =  dig-resp *(COMMA dig-resp)
dig-resp          =  username / realm / nonce / digest-uri
                      / dresponse / algorithm / cnonce
                      / opaque / message-qop
                      / nonce-count / auth-param
username          =  "username" EQUAL username-value
username-value    =  quoted-string
digest-uri        =  "uri" EQUAL LDQUOT digest-uri-value RDQUOT
digest-uri-value  =  rquest-uri ; Equal to request-uri as specified
                     by HTTP/1.1
message-qop       =  "qop" EQUAL qop-value

cnonce            =  "cnonce" EQUAL cnonce-value
cnonce-value      =  nonce-value
nonce-count       =  "nc" EQUAL nc-value
nc-value          =  8LHEX
dresponse         =  "response" EQUAL request-digest
request-digest    =  LDQUOT 32LHEX RDQUOT
auth-param        =  auth-param-name EQUAL
                     ( token / quoted-string )
auth-param-name   =  token
other-response    =  auth-scheme LWS auth-param
                     *(COMMA auth-param)
auth-scheme       =  token

Authentication-Info  =  "Authentication-Info" HCOLON ainfo
                        *(COMMA ainfo)
ainfo                =  nextnonce / message-qop
                         / response-auth / cnonce
                         / nonce-count
nextnonce            =  "nextnonce" EQUAL nonce-value
response-auth        =  "rspauth" EQUAL response-digest
response-digest      =  LDQUOT *LHEX RDQUOT

Call-ID  =  ( "Call-ID" / "i" ) HCOLON callid
callid   =  word [ "@" word ]

Call-Info   =  "Call-Info" HCOLON info *(COMMA info)
info        =  LAQUOT absoluteURI RAQUOT *( SEMI info-param)
info-param  =  ( "purpose" EQUAL ( "icon" / "info"
               / "card" / token ) ) / generic-param

Contact        =  ("Contact" / "m" ) HCOLON
                  ( STAR / (contact-param *(COMMA contact-param)))
contact-param  =  (name-addr / addr-spec) *(SEMI contact-params)
name-addr      =  [ display-name ] LAQUOT addr-spec RAQUOT
addr-spec      =  SIP-URI / SIPS-URI / absoluteURI
display-name   =  *(token LWS)/ quoted-string

contact-params    =  c-p-q / c-p-expires / feature-param
(taken from RFC 3840)            / contact-extension

 c-p-q              =  "q" EQUAL qvalue
c-p-expires        =  "expires" EQUAL delta-seconds
contact-extension  =  generic-param
delta-seconds      =  1*DIGIT

Content-Disposition   =  "Content-Disposition" HCOLON
                         disp-type *( SEMI disp-param )
disp-type             =  "render" / "session" / "icon" / "alert"
                         / disp-extension-token

disp-param            =  handling-param / generic-param
handling-param        =  "handling" EQUAL
                         ( "optional" / "required"
                         / other-handling )
other-handling        =  token
disp-extension-token  =  token

Content-Encoding  =  ( "Content-Encoding" / "e" ) HCOLON
```

```
                    content-coding *(COMMA content-coding)

Content-Language  =  "Content-Language" HCOLON
                     language-tag *(COMMA language-tag)
language-tag      =  primary-tag *( "-" subtag )
primary-tag       =  1*8ALPHA
subtag            =  1*8ALPHA

Content-Length  =  ( "Content-Length" / "l" ) HCOLON 1*DIGIT
Content-Type     =  ( "Content-Type" / "c" ) HCOLON media-type
media-type       =  m-type SLASH m-subtype *(SEMI m-parameter)
m-type           =  discrete-type / composite-type
discrete-type    =  "text" / "image" / "audio" / "video"
                    / "application" / extension-token
composite-type   =  "message" / "multipart" / extension-token
extension-token  =  ietf-token / x-token
ietf-token       =  token
x-token          =  "x-" token
m-subtype        =  extension-token / iana-token
iana-token       =  token
m-parameter      =  m-attribute EQUAL m-value
m-attribute      =  token
m-value          =  token / quoted-string

CSeq  =  "CSeq" HCOLON 1*DIGIT LWS Method

Date          =  "Date" HCOLON SIP-date
SIP-date      =  rfc1123-date
rfc1123-date  =  wkday "," SP date1 SP time SP "GMT"
date1         =  2DIGIT SP month SP 4DIGIT
                 ; day month year (e.g. 02 Jun 1982)
time          =  2DIGIT ":" 2DIGIT ":" 2DIGIT
                 ; 00:00:00 - 23:59:59
wkday         =  "Mon" / "Tue" / "Wed"
                 / "Thu" / "Fri" / "Sat" / "Sun"
month         =  "Jan" / "Feb" / "Mar" / "Apr"
                 / "May" / "Jun" / "Jul" / "Aug"
                 / "Sep" / "Oct" / "Nov" / "Dec"

Error-Info  =  "Error-Info" HCOLON error-uri *(COMMA error-uri)
error-uri   =  LAQUOT absoluteURI RAQUOT *( SEMI generic-param )

Expires     =  "Expires" HCOLON delta-seconds
From        =  ( "From" / "f" ) HCOLON from-spec
from-spec   =  ( name-addr / addr-spec )
               *( SEMI from-param )
from-param  =  tag-param / generic-param
tag-param   =  "tag" EQUAL token

In-Reply-To  =  "In-Reply-To" HCOLON callid *(COMMA callid)

Max-Forwards  =  "Max-Forwards" HCOLON 1*DIGIT

MIME-Version  =  "MIME-Version" HCOLON 1*DIGIT "." 1*DIGIT

Min-Expires  =  "Min-Expires" HCOLON delta-seconds

Organization  =  "Organization" HCOLON [TEXT-UTF8-TRIM]

Priority        =  "Priority" HCOLON priority-value
priority-value  =  "emergency" / "urgent" / "normal"
                   / "non-urgent" / other-priority
other-priority  =  token

Proxy-Authenticate  =  "Proxy-Authenticate" HCOLON challenge
challenge           =  ("Digest" LWS digest-cln *(COMMA digest-cln))
                       / other-challenge
other-challenge     =  auth-scheme LWS auth-param
                       *(COMMA auth-param)
digest-cln          =  realm / domain / nonce
                        / opaque / stale / algorithm
                        / qop-options / auth-param
realm               =  "realm" EQUAL realm-value
realm-value         =  quoted-string
domain              =  "domain" EQUAL LDQUOT URI
                       *( 1*SP URI ) RDQUOT
URI                 =  absoluteURI / abs-path
nonce               =  "nonce" EQUAL nonce-value
```

```
nonce-value         =  quoted-string
opaque              =  "opaque" EQUAL quoted-string
stale               =  "stale" EQUAL ( "true" / "false" )
algorithm           =  "algorithm" EQUAL ( "MD5" / "MD5-sess"
                       / token )
qop-options         =  "qop" EQUAL LDQUOT qop-value
                       *("," qop-value) RDQUOT
qop-value           =  "auth" / "auth-int" / token

Proxy-Authorization  =  "Proxy-Authorization" HCOLON credentials

Proxy-Require  =  "Proxy-Require" HCOLON option-tag
                  *(COMMA option-tag)
option-tag     =  token

Record-Route  =  "Record-Route" HCOLON rec-route *(COMMA rec-route)
rec-route     =  name-addr *( SEMI rr-param )
rr-param      =  generic-param

Reply-To      =  "Reply-To" HCOLON rplyto-spec
rplyto-spec   =  ( name-addr / addr-spec )
                 *( SEMI rplyto-param )
rplyto-param  =  generic-param
Require       =  "Require" HCOLON option-tag *(COMMA option-tag)

Retry-After  =  "Retry-After" HCOLON delta-seconds
                [ comment ] *( SEMI retry-param )

retry-param  =  ("duration" EQUAL delta-seconds)
                / generic-param

Route        =  "Route" HCOLON route-param *(COMMA route-param)
route-param  =  name-addr *( SEMI rr-param )

Server           =  "Server" HCOLON server-val *(LWS server-val)
server-val       =  product / comment
product          =  token [SLASH product-version]
product-version  =  token

Subject  =  ( "Subject" / "s" ) HCOLON [TEXT-UTF8-TRIM]

Supported  =  ( "Supported" / "k" ) HCOLON
              [option-tag *(COMMA option-tag)]

Timestamp  =  "Timestamp" HCOLON 1*(DIGIT)
               [ "." *(DIGIT) ] [ LWS delay ]
delay      =  *(DIGIT) [ "." *(DIGIT) ]

To        =  ( "To" / "t" ) HCOLON ( name-addr
             / addr-spec ) *( SEMI to-param )
to-param  =  tag-param / generic-param

Unsupported  =  "Unsupported" HCOLON option-tag *(COMMA option-tag)
User-Agent   =  "User-Agent" HCOLON server-val *(LWS server-val)

Via               =  ( "Via" / "v" ) HCOLON via-parm *(COMMA via-parm)
via-parm          =  sent-protocol LWS sent-by *( SEMI via-params )
via-params        =  via-ttl / via-maddr
                     / via-received / via-branch
                     / via-extension
via-ttl           =  "ttl" EQUAL ttl
via-maddr         =  "maddr" EQUAL host
via-received      =  "received" EQUAL (IPv4address / IPv6address)
via-branch        =  "branch" EQUAL token
via-extension     =  generic-param
sent-protocol     =  protocol-name SLASH protocol-version
                     SLASH transport
protocol-name     =  "SIP" / token
protocol-version  =  token
transport         =  "UDP" / "TCP" / "TLS" / "SCTP"
                     / other-transport
sent-by           =  host [ COLON port ]
ttl               =  1*3DIGIT ; 0 to 255

Warning        =  "Warning" HCOLON warning-value *(COMMA warning-value)
warning-value  =  warn-code SP warn-agent SP warn-text
warn-code      =  3DIGIT
warn-agent     =  hostport / pseudonym
```

```
                ;  the name or pseudonym of the server adding
                ;  the Warning header, for use in debugging
warn-text      =  quoted-string
pseudonym      =  token

WWW-Authenticate  =  "WWW-Authenticate" HCOLON challenge

extension-header  =  header-name HCOLON header-value
header-name       =  token
header-value      =  *(TEXT-UTF8char / UTF8-CONT / LWS)
message-body  =  *OCTET
```

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Accept	R		-	o	-	o	m*	o
Accept	2xx		-	-	-	o	m*	o
Accept	415		-	c	-	c	c	c
Accept-Encoding	R		-	o	-	o	o	o
Accept-Encoding	2xx		-	-	-	o	m*	o
Accept-Encoding	415		-	c	-	c	c	c
Accept-Language	R		-	o	-	o	o	o
Accept-Language	2xx		-	-	-	o	m*	o
Accept-Language	415		-	c	-	c	c	c
Alert-Info	R	ar	-	-	-	o	-	-
Alert-Info	180	ar	-	-	-	o	-	-
Allow	R		-	o	-	o	o	o
Allow	2xx		-	o	-	m*	m*	o
Allow	r		-	o	-	o	o	o
Allow	405		-	m	-	m	m	m
Authentication-Info	2xx		-	o	-	o	o	o
Authorization	R		o	o	o	o	o	o
Call-ID	c	r	m	m	m	m	m	m
Call-Info		ar	-	-	-	o	o	o
Contact	R		o	-	-	m	o	o
Contact	1xx		-	-	-	o	-	-
Contact	2xx		-	-	-	m	o	o
Contact	3xx	d	-	o	-	o	o	o
Contact	485		-	o	-	o	o	o
Content-Disposition			o	o	-	o	o	o
Content-Encoding			o	o	-	o	o	o
Content-Language			o	o	-	o	o	o
Content-Length		ar	t	t	t	t	t	t
Content-Type			*	*	-	*	*	*
CSeq	c	r	m	m	m	m	m	m
Date		a	o	o	o	o	o	o
Error-Info	300-699	a	-	o	o	o	o	o
Expires			-	-	-	o	-	o
From	c	r	m	m	m	m	m	m
In-Reply-To	R		-	-	-	o	-	-
Max-Forwards	R	amr	m	m	m	m	m	m
Min-Expires	423		-	-	-	-	-	m
MIME-Version			o	o	-	o	o	o
Organization		ar	-	-	-	o	o	o

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Priority	R	ar	-	-	-	o	-	-
Proxy-Authenticate	407	ar	-	m	-	m	m	m
Proxy-Authenticate	401	ar	-	o	o	o	o	o
Proxy-Authorization	R	dr	o	o	-	o	o	o
Proxy-Require	R	ar	-	o	-	o	o	o
Record-Route	R	ar	o	o	o	o	o	-
Record-Route	2xx,18x	mr	-	o	o	o	o	-
Reply-To			-	-	-	o	-	-
Require		ar	-	c	-	c	c	c
Retry-After	404,413,480,486		-	o	o	o	o	o
	500,503		-	o	o	o	o	o
	600,603		-	o	o	o	o	o
Route	R	adr	c	c	c	c	c	c
Server	r		-	o	o	o	o	o
Subject	R		-	-	-	o	-	-
Supported	R		-	o	o	m*	o	o
Supported	2xx		-	o	o	m*	m*	o
Timestamp			o	o	o	o	o	o
To	c(1)	r	m	m	m	m	m	m
Unsupported	420		-	m	-	m	m	m
User-Agent			o	o	o	o	o	o
Via	R	amr	m	m	m	m	m	m

Via	rc	dr	m	m	m	m	m	m
Warning	r		-	o	o	o	o	o
WWW-Authenticate	401	ar	-	m	-	m	m	m
WWW-Authenticate	407	ar	-	o	-	o	o	o

F.2 RFC 3262

```
PRACKm        =  %x50.52.41.43.4B ; PRACK in caps
Method        =  INVITEm / ACKm / OPTIONSm / BYEm
                 / CANCELm / REGISTERm / PRACKm
                 / extension-method
RAck          =  "RAck" HCOLON response-num LWS CSeq-num LWS Method
response-num  =  1*DIGIT
CSeq-num      =  1*DIGIT
RSeq          =  "RSeq" HCOLON response-num
```

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG	PRA
RAck	R		-	-	-	-	-	-	m
RSeq	1xx		-	-	-	o	-	-	-

Header field	where	PRACK
Accept	R	o
Accept	2xx	-
Accept	415	c
Accept-Encoding	R	o
Accept-Encoding	2xx	-
Accept-Encoding	415	c
Accept-Language	R	o
Accept-Language	2xx	-
Accept-Language	415	c
Alert-Info	R	-
Alert-Info	180	-
Allow	R	o
Allow	2xx	o
Allow	r	o
Allow	405	m
Authentication-Info	2xx	o
Authorization	R	o
Call-ID	c	m
Call-Info		-
Contact	R	-
Contact	1xx	-
Contact	2xx	-
Contact	3xx	o
Contact	485	o
Content-Disposition		o
Content-Encoding		o
Content-Language		o
Content-Length		t
Content-Type		*
CSeq	c	m
Date		o
Error-Info	300-699	o
Expires		-
From	c	m
In-Reply-To	R	-
Max-Forwards	R	m
Min-Expires	423	-
MIME-Version		o
Organization		-
Priority	R	-
Proxy-Authenticate	407	m
Proxy-Authenticate	401	o
Proxy-Authorization	R	o
Proxy-Require	R	o
Record-Route	R	o
Record-Route	2xx,18x	o
Reply-To		-
Require		c
Retry-After	404,413,480,486	o
	500,503	o
	600,603	o

Route	R	c
Server	r	o
Subject	R	-
Supported	R	o
Supported	2xx	o
Timestamp		o
To	c	m
Unsupported	420	m
User-Agent		o
Via	c	m
Warning	r	o
WWW-Authenticate	401	m

F.3 RFC 3265

6.4. Response Codes

This document registers two new response codes. These response codes are defined by the following information, which is to be added to the method and response-code sub-registry under http://www.iana.org/assignments/sip-parameters.

Response Code Number: 202
Default Reason Phrase: Accepted

Response Code Number: 489
Default Reason Phrase: Bad Event

7.1. New Methods

This document describes two new SIP methods: SUBSCRIBE and NOTIFY.

SUBSCRIBE and NOTIFY methods:

Header	Where	SUB	NOT
Accept	R	o	o
Accept	2xx	-	-
Accept	415	o	o
Accept-Encoding	R	o	o
Accept-Encoding	2xx	-	-
Accept-Encoding	415	o	o
Accept-Language	R	o	o
Accept-Language	2xx	-	-
Accept-Language	415	o	o
Alert-Info	R	-	-
Alert-Info	180	-	-
Allow	R	o	o
Allow	2xx	o	o
Allow	r	o	o
Allow	405	m	m
Authentication-Info	2xx	o	o
Authorization	R	o	o
Call-ID	c	m	m
Contact	R	m	m
Contact	1xx	o	o
Contact	2xx	m	o
Contact	3xx	m	m
Contact	485	o	o
Content-Disposition		o	o
Content-Encoding		o	o
Content-Language		o	o
Content-Length		t	t
Content-Type		*	*
CSeq	c	m	m
Date		o	o
Error-Info	300-699	o	o
Expires		o	-
Expires	2xx	m	-
From	c	m	m
In-Reply-To	R	-	-
Max-Forwards	R	m	m
Min-Expires	423	m	-
MIME-Version		o	o
Organization		o	-

Priority	R	o	-
Proxy-Authenticate	407	m	m
Proxy-Authorization	R	o	o
Proxy-Require	R	o	o
RAck	R	-	-
Record-Route	R	o	o
Record-Route	2xx,401,484	o	o
Reply-To		-	-
Require		o	o
Retry-After	404,413,480,486	o	o
Retry-After	500,503	o	o
Retry-After	600,603	o	o
Route	R	c	c
RSeq	1xx	o	o
Server	r	o	o
Subject	R	-	-
Supported	R	o	o
Supported	2xx	o	o
Timestamp		o	o
To	c(1)	m	m
Unsupported	420	o	o
User-Agent		o	o
Via	c	m	m
Warning	R	-	o
Warning	r	o	o
WWW-Authenticate	401	m	m

7.4. Augmented BNF Definitions

The Augmented BNF definitions for the various new and modified syntax elements follows. The notation is as used in SIP [1], and any elements not defined in this section are as defined in SIP and the documents to which it refers.

```
SUBSCRIBEm        = %x53.55.42.53.43.52.49.42.45 ; SUBSCRIBE in caps
NOTIFYm           = %x4E.4F.54.49.46.59 ; NOTIFY in caps
extension-method  = SUBSCRIBEm / NOTIFYm / token

Event             =  ( "Event" / "o" ) HCOLON event-type
                     *( SEMI event-param )
event-type        =  event-package *( "." event-template )
event-package     =  token-nodot
event-template    =  token-nodot
token-nodot       =  1*( alphanum / "-"  / "!" / "%" / "*"
                       / "_" / "+" / "`" / "'" / "~" )
event-param       =  generic-param / ( "id" EQUAL token )

Allow-Events =  ( "Allow-Events" / "u" ) HCOLON event-type
               *(COMMA event-type)

Subscription-State   = "Subscription-State" HCOLON substate-value
                       *( SEMI subexp-params )
substate-value       = "active" / "pending" / "terminated"
                       / extension-substate
extension-substate   = token
subexp-params        =   ("reason" EQUAL event-reason-value)
                       / ("expires" EQUAL delta-seconds)
                       / ("retry-after" EQUAL delta-seconds)
                       / generic-param
event-reason-value   =   "deactivated"
                       / "probation"
                       / "rejected"
                       / "timeout"
                       / "giveup"
                       / "noresource"
                       / event-reason-extension
event-reason-extension = token
```

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG	PRA	SUB	NOT
Allow-Events	R	o	o	-	o	o	o	o	o	o	
Allow-Events	2xx	-	o	-	o	o	o	o	o	o	
Allow-Events	489	-	-	-	-	-	-	-	m	m	
Event	R	-	-	-	-	-	-	-	m	m	
Subscription-State	R	-	-	-	-	-	-	-	-	m	

F.4 RFC 3311

UPDATE method:

Header field	where	proxy	UPDATE
Accept	R		o
Accept	2xx		o
Accept	415		c
Accept-Encoding	R		o
Accept-Encoding	2xx		o
Accept-Encoding	415		c
Accept-Language	R		o
Accept-Language	2xx		o
Accept-Language	415		c
Alert-Info			-
Allow	R		o
Allow	2xx		o
Allow	r		o
Allow	405		m
Allow-Events	(1)		-
Authentication-Info	2xx		o
Authorization	R		o
Call-ID	c	r	m
Call-Info		ar	o
Contact	R		m
Contact	1xx		o
Contact	2xx		m
Contact	3xx	d	o
Contact	485		o
Content-Disposition			o
Content-Encoding			o
Content-Language			o
Content-Length		ar	t
Content-Type			*
CSeq	c	r	m
Date		a	o
Error-Info	300-699	a	o
Event	(1)		-
Expires			-
From	c	r	m
In-Reply-To			-
Max-Forwards	R	amr	m
Min-Expires			-
MIME-Version			o
Organization		ar	o
Priority			-
Proxy-Authenticate	407	ar	m
Proxy-Authenticate	401	ar	o
Proxy-Authorization	R	dr	o
Proxy-Require	R	ar	o
RAck	R		-
Record-Route	R	ar	o
Record-Route	2xx,18x	mr	o
Reply-To			-
Require		ar	c
Retry-After	404,413,480,486		o
	500,503		o
	600,603		o
Route	R	adr	c
RSeq	-		-
Server	r		o
Subject	-		-
Subscription-State	(1)		-
Supported	R		o
Supported	2xx		o
Timestamp			o
To	c	r	m
Unsupported	420		m
User-Agent			o
Via	R	amr	m
Via	rc	dr	m
Warning	r		o
WWW-Authenticate	401	ar	m
WWW-Authenticate	407	ar	o

F.5 RFC 3313

```
P-Media-Authorization   = "P-Media-Authorization" HCOLON
                           P-Media-Authorization-Token
                           *(COMMA P-Media-Authorization-Token)

P-Media-Authorization-Token = 1*HEXDIG

                         Where  proxy  ACK  BYE  CAN  INV  OPT  REG
P-Media-Authorization      R      ad    o    -    -    o    -    -
P-Media-Authorization     2xx     ad    -    -    -    o    -    -
P-Media-Authorization   101-199   ad    -    -    -    o    -    -

                         Where  proxy  INF  PRA  UPD  SUB  NOT
P-Media-Authorization      R      ad    -    o    o    -    -
P-Media-Authorization     2xx     ad    -    o    o    -    -
```

F.6 RFC 3323

```
Privacy-hdr  =  "Privacy" HCOLON priv-value *(";" priv-value)
priv-value   =   "header" / "session" / "user" / "none" / "critical"
                / token

Header field          where   proxy ACK BYE CAN INV OPT REG
_____________________________________________________________
Privacy                       amrd  o   o   o   o   o   o

Header field                       SUB NOT PRK IFO UPD MSG
_____________________________________________________________
Privacy                             o   o   o   o   o   o
```

F.7 RFC 3325

9.1 The P-Asserted-Identity Header

The P-Asserted-Identity header field is used among trusted SIP entities (typically intermediaries) to carry the identity of the user sending a SIP message as it was verified by authentication.

```
PAssertedID = "P-Asserted-Identity" HCOLON PAssertedID-value
              *(COMMA PAssertedID-value)
PAssertedID-value = name-addr / addr-spec
```

A P-Asserted-Identity header field value MUST consist of exactly one name-addr or addr-spec. There may be one or two P-Asserted-Identity values. If there is one value, it MUST be a sip, sips, or tel URI. If there are two values, one value MUST be a sip or sips URI and the other MUST be a tel URI. It is worth noting that proxies can (and will) add and remove this header field.

9.2 The P-Preferred-Identity Header

The P-Preferred-Identity header field is used from a user agent to a trusted proxy to carry the identity the user sending the SIP message wishes to be used for the P-Asserted-Header field value that the trusted element will insert.

```
PPreferredID = "P-Preferred-Identity" HCOLON PPreferredID-value
               *(COMMA PPreferredID-value)
PPreferredID-value = name-addr / addr-spec
```

A P-Preferred-Identity header field value MUST consist of exactly one name-addr or addr-spec. There may be one or two P-Preferred-Identity values. If there is one value, it MUST be a sip, sips, or tel URI. If there are two values, one value MUST be a sip or sips URI and the other MUST be a tel URI. It is worth noting that proxies can (and will) remove this header field.

9.3 The "id" Privacy Type

This specification adds a new privacy type ("priv-value") to the Privacy header, defined in [2]. The presence of this privacy type in a Privacy header field indicates that the user would like the Network Asserted Identity to be kept private with respect to SIP entities outside the Trust Domain with which the user authenticated. Note that a user requesting multiple types of privacy MUST include all of the requested privacy types in its Privacy header field value.

priv-value = "id"

Example:

Privacy: id

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
P-Asserted-Identity		adr	-	o	-	o	o	-

	SUB	NOT	REF	INF	UPD	PRA
	o	o	o	-	-	-

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
P-Preferred-Identity		adr	-	o	-	o	o	-

	SUB	NOT	REF	INF	UPD	PRA
	o	o	o	-	-	-

F.8 RFC 3326

```
Reason            =  "Reason" HCOLON reason-value *(COMMA reason-value)
 reason-value      =  protocol *(SEMI reason-params)
 protocol          =  "SIP" / "Q.850" / token
 reason-params     =  protocol-cause / reason-text
                      / reason-extension
 protocol-cause    =  "cause" EQUAL cause
 cause             =  1*DIGIT
 reason-text       =  "text" EQUAL quoted-string
 reason-extension  =  generic-param
```

The following values for the protocol field have been defined:

SIP: The cause parameter contains a SIP status code.

Q.850: The cause parameter contains an ITU-T Q.850 cause value in decimal representation.

Examples are:

Reason: SIP ;cause=200 ;text="Call completed elsewhere"
Reason: Q.850 ;cause=16 ;text="Terminated"
Reason: SIP ;cause=600 ;text="Busy Everywhere"
Reason: SIP ;cause=580 ;text="Precondition Failure"

F.9 RFC 3327

Path = "Path" HCOLON path-value *(COMMA path-value)

path-value = name-addr *(SEMI rr-param)

Note that the Path header field values conform to the syntax of a Route element as defined in [1]. As suggested therein, such values MUST include the loose-routing indicator parameter ";lr" for full compliance with [1].

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG

Path	R	ar	-	-	-	-	-	o
Path	2xx		-	-	-	-	-	o

F.10 RFC 3329

```
security-client  = "Security-Client" HCOLON
                         sec-mechanism *(COMMA sec-mechanism)
security-server  = "Security-Server" HCOLON
                         sec-mechanism *(COMMA sec-mechanism)
security-verify  = "Security-Verify" HCOLON
                         sec-mechanism *(COMMA sec-mechanism)

sec-mechanism    = mechanism-name *(SEMI mech-parameters)
mechanism-name   = ( "digest" / "tls" / "ipsec-ike" /
                           "ipsec-man" / token )
mech-parameters  = ( preference / digest-algorithm /
                            digest-qop / digest-verify / extension )
preference       = "q" EQUAL qvalue
qvalue           = ( "0" [ "." 0*3DIGIT ] )
                          / ( "1" [ "." 0*3("0") ] )
digest-algorithm = "d-alg" EQUAL token
digest-qop       = "d-qop" EQUAL token
digest-verify    = "d-ver" EQUAL LDQUOT 32LHEX RDQUOT
extension        = generic-param
```

Note that qvalue is already defined in the SIP BNF [1]. We have copied its definitions here for completeness.

The parameters described by the BNF above have the following semantics:

Mechanism-name
This token identifies the security mechanism supported by the client, when it appears in a Security-Client header field; or by the server, when it appears in a Security-Server or in a Security-Verify header field. The mechanism-name tokens are registered with the IANA. This specification defines four values:

* "tls" for TLS [3].

* "digest" for HTTP Digest [4].

* "ipsec-ike" for IPsec with IKE [2].

* "ipsec-man" for manually keyed IPsec without IKE.

Preference
The "q" value indicates a relative preference for the particular mechanism. The higher the value the more preferred the mechanism is. All the security mechanisms MUST have different "q" values. It is an error to provide two mechanisms with the same "q" value.

Digest-algorithm
This optional parameter is defined here only for HTTP Digest [4] in order to prevent the bidding-down attack for the HTTP Digest algorithm parameter. The content of the field may have same values as defined in [4] for the "algorithm" field.

Digest-qop
This optional parameter is defined here only for HTTP Digest [4] in order to prevent the bidding-down attack for the HTTP Digest qop parameter. The content of the field may have same values as defined in [4] for the "qop" field.

Digest-verify
This optional parameter is defined here only for HTTP Digest [4] in order to prevent the bidding-down attack for the SIP security mechanism agreement (this document). The content of the field is counted exactly the same way as "request-digest" in [4] except that the Security-Server header field is included in the A2 parameter. If the "qop" directive's value is "auth" or is unspecified, then A2 is:

A2 = Method ":" digest-uri-value ":" security-server

```
      If the "qop" value is "auth-int", then A2 is:

      A2 = Method ":" digest-uri-value ":" H(entity-body) ":"
      security-server

   All linear white spaces in the Security-Server header field
   MUST be replaced by a single SP before calculating or
   interpreting the digest-verify parameter.  Method, digest-uri-
   value, entity-body, and any other HTTP Digest parameter are as
   specified in [4].
```

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Security-Client	R	ard	-	o	-	o	o	o
Security-Server	421,494		-	o	-	o	o	o
Security-Verify	R	ard	-	o	-	o	o	o

Header field	where	proxy	SUB	NOT	PRK	IFO	UPD	MSG
Security-Client	R	ard	o	o	-	o	o	o
Security-Server	421,494		o	o	-	o	o	o
Security-Verify	R	ard	o	o	-	o	o	o

F.11 RFC 3428

MESSAGE method:

Header Field	where	proxy	MESSAGE
Accept	R		-
Accept	2xx		-
Accept	415		m*
Accept-Encoding	R		-
Accept-Encoding	2xx		-
Accept-Encoding	415		m*
Accept-Language	R		-
Accept-Language	2xx		-
Accept-Language	415		m*
Alert-Info	R		-
Alert-Info	180		-
Allow	R		o
Allow	2xx		o
Allow	r		o
Allow	405		m
Authentication-Info	2xx		o
Authorization	R		o
Call-ID	c	r	m
Call-Info		ar	o
Contact	R		-
Contact	1xx		-
Contact	2xx		-
Contact	3xx		o
Contact	485		o
Content-Disposition			o
Content-Encoding			o
Content-Language			o
Content-Length		ar	t
Content-Type			*
CSeq	c	r	m
Date		a	o
Error-Info	300-699	a	o
Expires			o
From	c	r	m
In-Reply-To	R		o
Max-Forwards	R	amr	m
Organization		ar	o
Priority	R	ar	o
Proxy-Authenticate	407	ar	m
Proxy-Authenticate	401	ar	o
Proxy-Authorization	R	dr	o
Proxy-Require	R	ar	o
Record-Route		ar	-
Reply-To			o
Require		ar	c

```
Retry-After   404,413,480,486         o
                  500,503             o
                  600,603             o
Route                R     adr        o
Server               r                o
Subject              R                o
Timestamp                             o
To                 c(1)     r         m
Unsupported         420               o
User-Agent                            o
Via                  R     amr        m
Via                 rc     dr         m
Warning              r                o
WWW-Authenticate    401    ar         m
WWW-Authenticate    407    ar         o

(1): copied  with  possible addition of tag
```

F.12 RFC 3455

```
5.1 P-Associated-URI header syntax

   The syntax of the P-Associated-URI header is described as follows:

      P-Associated-URI       = "P-Associated-URI" HCOLON
                               (p-aso-uri-spec)
                               *(COMMA p-aso-uri-spec)
      p-aso-uri-spec         = name-addr *(SEMI ai-param)
      ai-param               = generic-param

5.2 P-Called-Party-ID header syntax

   The syntax of the P-Called-Party-ID header is described as follows:

      P-Called-Party-ID      = "P-Called-Party-ID" HCOLON
                               called-pty-id-spec
      called-pty-id-spec     = name-addr *(SEMI cpid-param)
      cpid-param             = generic-param

5.3 P-Visited-Network-ID header syntax

   The syntax of the P-Visited-Network-ID header is described as
   follows:

      P-Visited-Network-ID   = "P-Visited-Network-ID" HCOLON
                                vnetwork-spec
                               *(COMMA vnetwork-spec)
      vnetwork-spec          = (token / quoted-string)
                                *(SEMI vnetwork-param)
      vnetwork-param         = generic-param

5.4 P-Access-Network-Info header syntax

   The syntax of the P-Access-Network-Info header is described as
   follows:

      P-Access-Network-Info  = "P-Access-Network-Info" HCOLON
                               access-net-spec
      access-net-spec        = access-type *(SEMI access-info)
      access-type            = "IEEE-802.11a" / "IEEE-802.11b" /
                               "3GPP-GERAN" / "3GPP-UTRAN-FDD" /
                               "3GPP-UTRAN-TDD" /
                               "3GPP-CDMA2000" / token
      access-info            = cgi-3gpp / utran-cell-id-3gpp /
                               extension-access-info
      extension-access-info  = gen-value
      cgi-3gpp               = "cgi-3gpp" EQUAL
                               (token / quoted-string)
      utran-cell-id-3gpp     = "utran-cell-id-3gpp" EQUAL
                               (token / quoted-string)

   The access-info may contain additional information relating to the
   access network.  The values for "cgi-3gpp" and "utran-cell-id-3gpp"
   are defined in 3GPP TS 24.229 [15].

5.5 P-Charging-Function-Addresses header syntax
```

```
   The syntax for the P-Charging-Function-Addresses header is described
   as follows:

      P-Charging-Addr        = "P-Charging-Function-Addresses" HCOLON
                               charge-addr-params
                               *(SEMI charge-addr-params)
      charge-addr-params     = ccf / ecf / generic-param
      ccf                    = "ccf" EQUAL gen-value
      ecf                    = "ecf" EQUAL gen-value

5.6 P-Charging-Vector header syntax

      The syntax for the P-Charging-Vector header is described as
      follows:

      P-Charging-Vector     = "P-Charging-Vector" HCOLON icid-value
                              *(SEMI charge-params)
      charge-params         = icid-gen-addr / orig-ioi /
                              term-ioi / generic-param
      icid-value            = "icid-value" EQUAL gen-value
      icid-gen-addr         = "icid-generated-at" EQUAL host
      orig-ioi              = "orig-ioi" EQUAL gen-value
      term-ioi              = "term-ioi" EQUAL gen-value

   Header field          where  proxy  ACK BYE CAN INV OPT REG
   ____________________________________________________________
   P-Associated-URI       2xx           -   -   -   -   -   o
   P-Called-Party-ID       R     amr    -   -   -   o   o   -
   P-Visited-Network-ID    R     ad     -   -   -   o   o   o
   P-Access-Network-Info         dr     -   o   -   o   o   o
   P-Charging-Vector             admr   -   o   -   o   o   o
   P-Charging-Function-          adr    -   o   -   o   o   o
        Addresses

   Header field                    SUB NOT PRA INF UPD MSG REF
   ____________________________________________________________
   P-Associated-URI                 -   -   -   -   -   -   -
   P-Called-Party-ID                o   -   -   -   -   o   o
   P-Visited-Network-ID             o   -   -   -   -   o   o
   P-Access-Network-Info            o   o   o   o   o   o   o
   P-Charging-Vector                o   o   o   o   o   o   o
   P-Charging-Function-             o   o   o   o   o   o   o
     Addresses
```

See also 3GPP TS 24.229, clause 7.2a.5.2 for the syntax of extenstions to the P-Charging-Vector header field.

F.13 RFC 3515

REFER method:

Header	Where	REFER
Accept	R	o
Accept	2xx	-
Accept	415	c
Accept-Encoding	R	o
Accept-Encoding	2xx	-
Accept-Encoding	415	c
Accept-Language	R	o
Accept-Language	2xx	-
Accept-Language	415	c
Alert-Info		-
Allow	Rr	o
Allow	405	m
Authentication-Info	2xx	o
Authorization	R	o
Call-ID	c	m
Call-Info		-
Contact	R	m
Contact	1xx	-
Contact	2xx	m
Contact	3-6xx	o
Content-Disposition		o
Content-Encoding		o
Content-Language		o

Content-Length		o
Content-Type		*
CSeq	c	m
Date		o
Error-Info	3-6xx	o
Expires	R	o
From	c	m
In-Reply-To		-
Max-Forwards	R	m
Min-Expires		-
MIME-Version		o
Organization		o
Priority	R	-
Proxy-Authenticate	401	o
Proxy-Authenticate	407	m
Proxy-Authorization	R	o
Proxy-Require	R	o
Record-Route	R	o
Record-Route	2xx,18x	o
Reply-To		-
Require		c
Retry-After	404,413,480,486	o
Retry-After	500,503	o
Retry-After	600,603	o
Route	R	c
Server	r	o
Subject	R	-
Supported	R,2xx	o
Timestamp		o
To	c(1)	m
Unsupported	420	o
User-Agent		o
Via	c(2)	m
Warning	r	o
WWW-Authenticate	401	m
WWW-Authenticate	407	o

Refer-To is a request header field (request-header) as defined by [1]. It only appears in a REFER request. It provides a URL to reference.

Refer-To = ("Refer-To" / "r") HCOLON (name-addr / addr-spec) * (SEMI generic-param)

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Refer-To	R		-	-	-	-	-	-

F.14 RFC 3608

Service-Route = "Service-Route" HCOLON sr-value *(COMMA sr-value)

sr-value = name-addr *(SEMI rr-param)

Note that the Service-Route header field values MUST conform to the syntax of a Route element as defined in [3]. As suggested therein, such values MUST include the loose-routing indicator parameter ";lr" for full compliance with [3].

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG	PRA
Service-Route	2xx	ar	-	-	-	-	-	o	-

F.15 RFC 3840

```
feature-param    =  enc-feature-tag [EQUAL LDQUOT (tag-value-list
                    / string-value ) RDQUOT]
enc-feature-tag  =  base-tags / other-tags
base-tags        =  "audio" / "automata" /
                    "class" / "duplex" / "data" /
                    "control" / "mobility" / "description" /
                    "events" / "priority" / "methods" /
                    "schemes" / "application" / "video" /
                    "language" / "type" / "isfocus" /
```

```
                    "actor" / "text" / "extensions"
other-tags      =  "+" ftag-name
ftag-name       =  ALPHA *( ALPHA / DIGIT / "!" / "'" /
                   "." / "-" / "%" )
tag-value-list  =  tag-value *("," tag-value)
tag-value       =  ["!"] (token-nobang / boolean / numeric)
token-nobang    =  1*(alphanum / "-" / "." / "%" / "*"
                   / "_" / "+" / "`" / "'" / "~" )
boolean         =  "TRUE" / "FALSE"
numeric         =  "#" numeric-relation number
numeric-relation  =  ">=" / "<=" / "=" / (number ":")
number          =  [ "+" / "-" ] 1*DIGIT ["." 0*DIGIT]
string-value    =  "<" *(qdtext-no-abkt / quoted-pair ) ">"
qdtext-no-abkt  =  LWS / %x21 / %x23-3B / %x3D
                        / %x3F-5B / %x5D-7E / UTF8-NONASCII
```

F.16 RFC 3841

```
Request-Disposition   =   ( "Request-Disposition" / "d" ) HCOLON
                          directive *(COMMA directive)
directive             =   proxy-directive / cancel-directive /
                          fork-directive / recurse-directive /
                          parallel-directive / queue-directive
proxy-directive       =  "proxy" / "redirect"
cancel-directive      =  "cancel" / "no-cancel"
fork-directive        =  "fork" / "no-fork"
recurse-directive     =  "recurse" / "no-recurse"
parallel-directive    =  "parallel" / "sequential"
queue-directive       =  "queue" / "no-queue"

Accept-Contact  =  ("Accept-Contact" / "a") HCOLON ac-value
                   *(COMMA ac-value)
Reject-Contact  =  ("Reject-Contact" / "j") HCOLON rc-value
                   *(COMMA rc-value)
ac-value        =  "*" *(SEMI ac-params)
rc-value        =  "*" *(SEMI rc-params)
ac-params       =  feature-param / req-param
                      / explicit-param / generic-param
                    ;;feature param from RFC 3840
                    ;;generic-param from RFC 3261
rc-params       =  feature-param / generic-param
req-param       =  "require"
explicit-param  =  "explicit"
```

Despite the BNF, there MUST NOT be more than one req-param or explicit-param in an ac-params. Furthermore, there can only be one instance of any feature tag in feature-param.

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Accept-Contact	R	ar	o	o	o	o	o	-
Reject-Contact	R	ar	o	o	o	o	o	-
Request-Disposition	R	ar	o	o	o	o	o	o

Figure 2: Accept-Contact, Reject-Contact, and Request-Disposition header fields

Header field	where	proxy	PRA	UPD	SUB	NOT	INF	MSG	REF
Accept-Contact	R	ar	o	o	o	o	o	o	o
Reject-Contact	R	ar	o	o	o	o	o	o	o
Request-Disposition	R	ar	o	o	o	o	o	o	o

F.17 RFC 3891

```
   Replaces        = "Replaces" HCOLON callid *(SEMI replaces-param)
   replaces-param  = to-tag / from-tag / early-flag / generic-param
   to-tag          = "to-tag" EQUAL token
   from-tag        = "from-tag" EQUAL token
   early-flag      = "early-only"
```

A Replaces header field MUST contain exactly one to-tag and exactly one from-tag, as they are required for unique dialog matching. For compatibility with dialogs initiated by RFC 2543 [9] compliant UAs, a

tag of zero matches both tags of zero and null. A Replaces header field MAY contain the early-flag.

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG	MSG
Replaces	R		-	-	-	o	-	-	-

Header field	where	SUB	NOT	REF	INF	UPD	PRA	PUB
Replaces	R	-	-	-	-	-	-	-

F.18 RFC 3892

```
   Referred-By  =  ("Referred-By" / "b") HCOLON referrer-uri
                  *( SEMI (referredby-id-param / generic-param) )

   referrer-uri = ( name-addr / addr-spec )

   referredby-id-param = "cid" EQUAL sip-clean-msg-id

   sip-clean-msg-id = LDQUOT dot-atom "@" (dot-atom / host) RDQUOT

   dot-atom = atom *( "." atom )

   atom     = 1*( alphanum / "-" / "!" / "%" / "*" /
                      "_" / "+" / "'" / "`" / "~"   )
```

Since the Content-ID appears as a SIP header parameter value which must conform to the expansion of the gen-value defined in [5], this grammar produces values in the intersection of the expansions of gen-value and msg-id from [9]. The double-quotes surrounding the sip-clean-msg-id MUST be replaced with left and right angle brackets to derive the Content-ID used in the message's MIME body. For example,

```
   Referred-By: sip:r@ref.example;cid="2UWQFN309shb3@ref.example"
      indicates the token is in the body part containing

        Content-ID: <2UWQFN309shb3@ref.example>
```

If the referrer-uri contains a comma, question mark, or semicolon, (for example, if it contains URI parameters) the URI MUST be enclosed in angle brackets (< and >). Any URI parameters are contained within these brackets. If the URI is not enclosed in angle brackets, any semicolon-delimited parameters are header-parameters, not URI parameters.

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Referred-By	R		-	o	-	o	o	o

F.19 RFC 3903

```
   PUBLISHm           = %x50.55.42.4C.49.53.48 ; PUBLISH in caps.
   extension-method   = PUBLISHm / token
   SIP-ETag           = "SIP-ETag" HCOLON entity-tag
   SIP-If-Match       = "SIP-If-Match" HCOLON entity-tag
   entity-tag         = token
```

Header Field	where	PUBLISH
Accept	R	o
Accept	2xx	-
Accept	415	m*
Accept-Encoding	R	o
Accept-Encoding	2xx	-
Accept-Encoding	415	m*
Accept-Language	R	o
Accept-Language	2xx	-
Accept-Language	415	m*
Alert-Info		-
Allow	R	o

Allow	r	o
Allow	405	m
Allow-Events	R	o
Allow-Events	489	m
Authentication-Info	2xx	o
Authorization	R	o
Call-ID	c	m
Call-Info		o
Contact	R	-
Contact	1xx	-
Contact	2xx	-
Contact	3xx	o
Contact	485	o
Content-Disposition		o
Content-Encoding		o
Content-Language		o
Content-Length		t
Content-Type		*
CSeq	c	m
Date		o
Event	R	m
Error-Info	300-699	o
Expires		o
Expires	2xx	m
From	c	m
In-Reply-To	R	-
Max-Forwards	R	m
Min-Expires	423	m
MIME-Version		o
Organization		o
Priority	R	o
Proxy-Authenticate	407	m
Proxy-Authenticate	401	o
Proxy-Authorization	R	o
Proxy-Require	R	o
Record-Route		-
Reply-To		-
Require		o
Retry-After	404,413,480,486	o
Retry-After	500,503	o
Retry-After	600,603	o
Route	R	c
Server	r	o
Subject	R	o
Supported	R	o
Supported	2xx	o
Timestamp		o
To	c(1)	m
Unsupported	420	o
User-Agent		o
Via	R	m
Via	rc	m
Warning	r	o
WWW-Authenticate	401	m
WWW-Authenticate	407	o

Header Field	where	proxy	ACK	BYE	CAN	INF	INV
SIP-ETag	2xx		-	-	-	-	-
SIP-If-Match	R		-	-	-	-	-

Header Field	where	proxy	NOT	OPT	PRA	REG	SUB
SIP-ETag	2xx		-	-	-	-	-
SIP-If-Match	R		-	-	-	-	-

Header Field	where	proxy	UPD	MSG	REF	PUBLISH
SIP-ETag	2xx		-	-	-	m
SIP-If-Match	R		-	-	-	o

F.20 RFC 3911

```
   Join            = "Join" HCOLON callid *(SEMI join-param)
   join-param      = to-tag / from-tag / generic-param
   to-tag          = "to-tag" EQUAL token
   from-tag        = "from-tag" EQUAL token
```

A Join header MUST contain exactly one to-tag and exactly one from-tag, as they are required for unique dialog matching. For compatibility with dialogs initiated by RFC 2543 [11] compliant UAs, a to-tag of zero matches both a to-tag value of zero and a null to-tag. Likewise, a from-tag of zero matches both a to-tag value of zero and a null from-tag.

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG	MSG
Join	R		-	-	-	o	-	-	-

	SUB	NOT	REF	INF	UPD	PRA	PUB
Join	-	-	-	-	-	-	-

F.21 RFC 4028

```
Min-SE  =  "Min-SE" HCOLON delta-seconds *(SEMI generic-param)

Session-Expires  = ("Session-Expires" / "x") HCOLON delta-seconds
                    *(SEMI se-params)
se-params        = refresher-param / generic-param
refresher-param  = "refresher" EQUAL  ("uas" / "uac")
```

Header	where	proxy	ACK	BYE	CAN	INV	OPT	REG	PRA	UPD	SUB	NOT
Session-Expires	R	amr	-	-	-	o	-	-	-	o	-	-
Session-Expires	2xx	ar	-	-	-	o	-	-	-	o	-	-
Min-SE	R	amr	-	-	-	o	-	-	-	o	-	-
Min-SE	422		-	-	-	m	-	-	-	m	-	-

Annex G (informative): DHCP and DNS Message Definitions

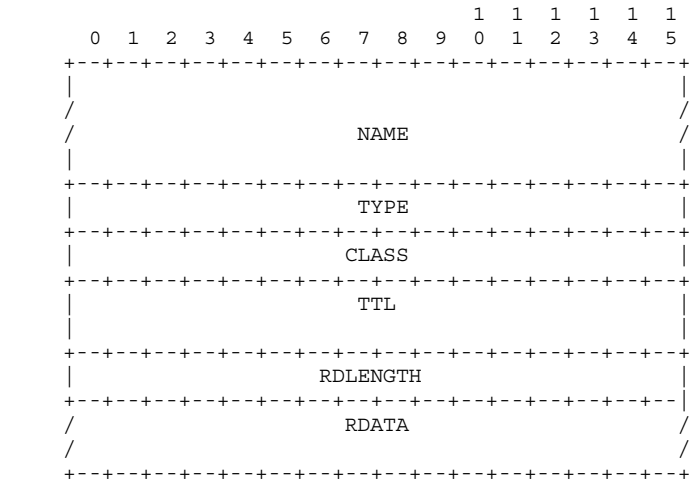
This is a list of the DNS and DHCP (v4 and v6) definitions compiled from all necessary RFCs.

G.1 RFC 1035

3.2. RR definitions

3.2.1. Format

All RRs have the same top level format shown below:



where:

NAME — an owner name, i.e., the name of the node to which this resource record pertains.

TYPE — two octets containing one of the RR TYPE codes.

CLASS — two octets containing one of the RR CLASS codes.

TTL — a 32 bit signed integer that specifies the time interval that the resource record may be cached before the source of the information should again be consulted. Zero values are interpreted to mean that the RR can only be used for the transaction in progress, and should not be cached. For example, SOA records are always distributed with a zero TTL to prohibit caching. Zero values can also be used for extremely volatile data.

RDLENGTH — an unsigned 16 bit integer that specifies the length in octets of the RDATA field.

RDATA — a variable length string of octets that describes the resource. The format of this information varies according to the TYPE and CLASS of the resource record.

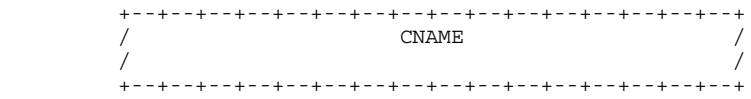
3.3. Standard RRs

The following RR definitions are expected to occur, at least potentially, in all classes. In particular, NS, SOA, CNAME, and PTR will be used in all classes, and have the same format in all classes. Because their RDATA format is known, all domain names in the RDATA section of these RRs may be compressed.

<domain-name> is a domain name represented as a series of labels, and terminated by a label with zero length. <character-string> is a single

length octet followed by that number of characters. <character-string> is treated as binary information, and can be up to 256 characters in length (including the length octet).

3.3.1. CNAME RDATA format

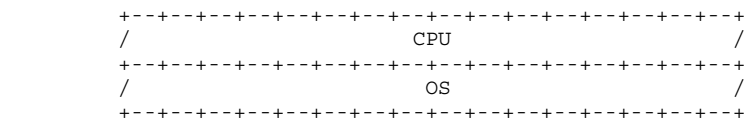


where:

CNAME A <domain-name> which specifies the canonical or primary name for the owner. The owner name is an alias.

CNAME RRs cause no additional section processing, but name servers may choose to restart the query at the canonical name in certain cases. See the description of name server logic in [RFC-1034] for details.

3.3.2. HINFO RDATA format



where:

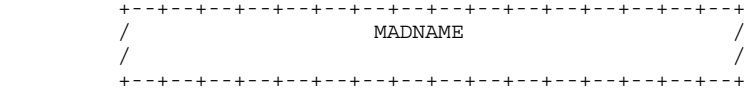
CPU A <character-string> which specifies the CPU type.

OS A <character-string> which specifies the operating system type.

Standard values for CPU and OS can be found in [RFC-1010].

HINFO records are used to acquire general information about a host. The main use is for protocols such as FTP that can use special procedures when talking between machines or operating systems of the same type.

3.3.3. MB RDATA format (EXPERIMENTAL)

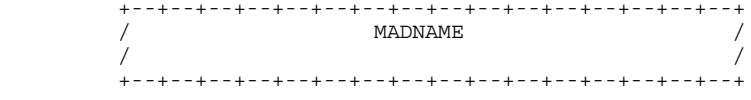


where:

MADNAME A <domain-name> which specifies a host which has the specified mailbox.

MB records cause additional section processing which looks up an A type RRs corresponding to MADNAME.

3.3.4. MD RDATA format (Obsolete)



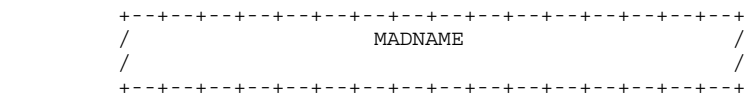
where:

MADNAME A <domain-name> which specifies a host which has a mail agent for the domain which should be able to deliver mail for the domain.

MD records cause additional section processing which looks up an A type record corresponding to MADNAME.

MD is obsolete. See the definition of MX and [RFC-974] for details of the new scheme. The recommended policy for dealing with MD RRs found in a master file is to reject them, or to convert them to MX RRs with a preference of 0.

3.3.5. MF RDATA format (Obsolete)



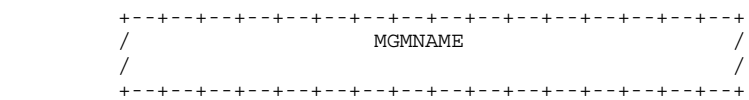
where:

MADNAME A <domain-name> which specifies a host which has a mail agent for the domain which will accept mail for forwarding to the domain.

MF records cause additional section processing which looks up an A type record corresponding to MADNAME.

MF is obsolete. See the definition of MX and [RFC-974] for details ofw the new scheme. The recommended policy for dealing with MD RRs found in a master file is to reject them, or to convert them to MX RRs with a preference of 10.

3.3.6. MG RDATA format (EXPERIMENTAL)

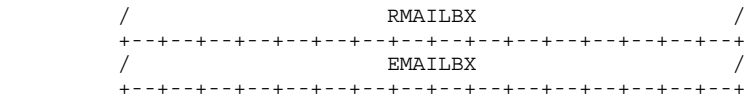


where:

MGMNAME A <domain-name> which specifies a mailbox which is a member of the mail group specified by the domain name.

MG records cause no additional section processing.

3.3.7. MINFO RDATA format (EXPERIMENTAL)



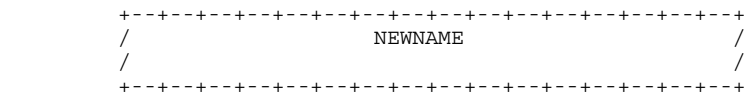
where:

RMAILBX A <domain-name> which specifies a mailbox which is responsible for the mailing list or mailbox. If this domain name names the root, the owner of the MINFO RR is responsible for itself. Note that many existing mailing lists use a mailbox X-request for the RMAILBX field of mailing list X, e.g., Msgroup-request for Msgroup. This field provides a more general mechanism.

EMAILBX A <domain-name> which specifies a mailbox which is to receive error messages related to the mailing list or mailbox specified by the owner of the MINFO RR (similar to the ERRORS-TO: field which has been proposed). If this domain name names the root, errors should be returned to the sender of the message.

MINFO records cause no additional section processing. Although these records can be associated with a simple mailbox, they are usually used with a mailing list.

3.3.8. MR RDATA format (EXPERIMENTAL)



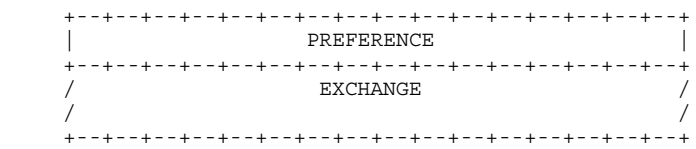
where:

NEWNAME A <domain-name> which specifies a mailbox which is the proper rename of the specified mailbox.

MR records cause no additional section processing. The main use for MR is as a forwarding entry for a user who has moved to a different

mailbox.

3.3.9. MX RDATA format



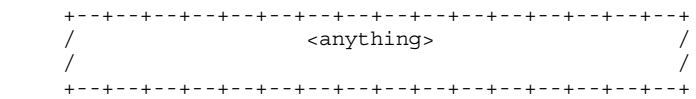
where:

PREFERENCE A 16 bit integer which specifies the preference given to this RR among others at the same owner. Lower values are preferred.

EXCHANGE A <domain-name> which specifies a host willing to act as a mail exchange for the owner name.

MX records cause type A additional section processing for the host specified by EXCHANGE. The use of MX RRs is explained in detail in [RFC-974].

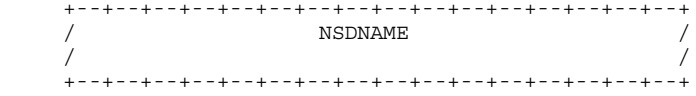
3.3.10. NULL RDATA format (EXPERIMENTAL)



Anything at all may be in the RDATA field so long as it is 65535 octets or less.

NULL records cause no additional section processing. NULL RRs are not allowed in master files. NULLs are used as placeholders in some experimental extensions of the DNS.

3.3.11. NS RDATA format



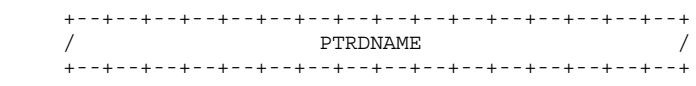
where:

NSDNAME A <domain-name> which specifies a host which should be authoritative for the specified class and domain.

NS records cause both the usual additional section processing to locate a type A record, and, when used in a referral, a special search of the zone in which they reside for glue information.

The NS RR states that the named host should be expected to have a zone starting at owner name of the specified class. Note that the class may not indicate the protocol family which should be used to communicate with the host, although it is typically a strong hint. For example, hosts which are name servers for either Internet (IN) or Hesiod (HS) class information are normally queried using IN class protocols.

3.3.12. PTR RDATA format

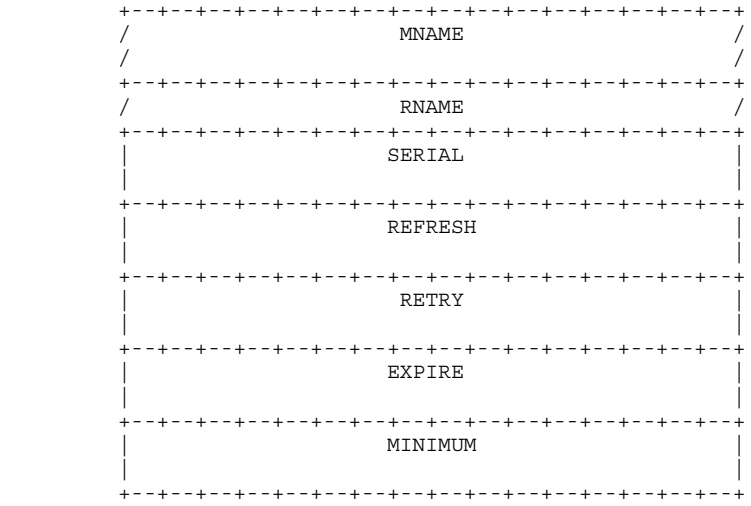


where:

PTRDNAME A <domain-name> which points to some location in the domain name space.

PTR records cause no additional section processing. These RRs are used in special domains to point to some other location in the domain space. These records are simple data, and don't imply any special processing similar to that performed by CNAME, which identifies aliases. See the description of the IN-ADDR.ARPA domain for an example.

3.3.13. SOA RDATA format



where:

MNAME The <domain-name> of the name server that was the original or primary source of data for this zone.

RNAME A <domain-name> which specifies the mailbox of the person responsible for this zone.

SERIAL The unsigned 32 bit version number of the original copy of the zone. Zone transfers preserve this value. This value wraps and should be compared using sequence space arithmetic.

REFRESH A 32 bit time interval before the zone should be refreshed.

RETRY A 32 bit time interval that should elapse before a failed refresh should be retried.

EXPIRE A 32 bit time value that specifies the upper limit on the time interval that can elapse before the zone is no longer authoritative.

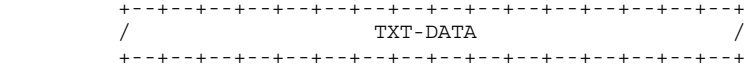
MINIMUM The unsigned 32 bit minimum TTL field that should be exported with any RR from this zone.

SOA records cause no additional section processing.

All times are in units of seconds.

Most of these fields are pertinent only for name server maintenance operations. However, MINIMUM is used in all query operations that retrieve RRs from a zone. Whenever a RR is sent in a response to a query, the TTL field is set to the maximum of the TTL field from the RR and the MINIMUM field in the appropriate SOA. Thus MINIMUM is a lower bound on the TTL field for all RRs in a zone. Note that this use of MINIMUM should occur when the RRs are copied into the response and not when the zone is loaded from a master file or via a zone transfer. The reason for this provison is to allow future dynamic update facilities to change the SOA RR with known semantics.

3.3.14. TXT RDATA format



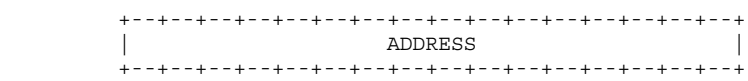
where:

TXT-DATA One or more <character-string>s.

TXT RRs are used to hold descriptive text. The semantics of the text depends on the domain where it is found.

3.4. Internet specific RRs

3.4.1. A RDATA format



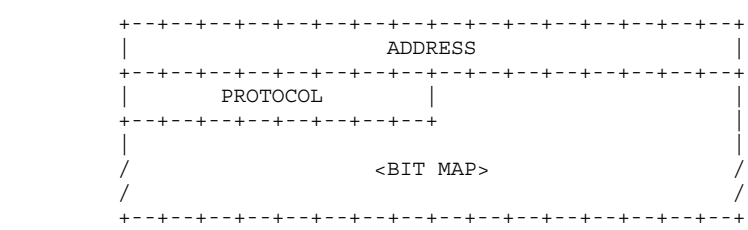
where:

ADDRESS A 32 bit Internet address.

Hosts that have multiple Internet addresses will have multiple A records.

A records cause no additional section processing. The RDATA section of an A line in a master file is an Internet address expressed as four decimal numbers separated by dots without any imbedded spaces (e.g., "10.2.0.52" or "192.0.5.6").

3.4.2. WKS RDATA format



where:

ADDRESS An 32 bit Internet address

PROTOCOL An 8 bit IP protocol number

<BIT MAP> A variable length bit map. The bit map must be a multiple of 8 bits long.

The WKS record is used to describe the well known services supported by a particular protocol on a particular internet address. The PROTOCOL field specifies an IP protocol number, and the bit map has one bit per port of the specified protocol. The first bit corresponds to port 0, the second to port 1, etc. If the bit map does not include a bit for a protocol of interest, that bit is assumed zero. The appropriate values and mnemonics for ports and protocols are specified in [RFC-1010].

For example, if PROTOCOL=TCP (6), the 26th bit corresponds to TCP port 25 (SMTP). If this bit is set, a SMTP server should be listening on TCP port 25; if zero, SMTP service is not supported on the specified address.

The purpose of WKS RRs is to provide availability information for servers for TCP and UDP. If a server supports both TCP and UDP, or has multiple Internet addresses, then multiple WKS RRs are used.

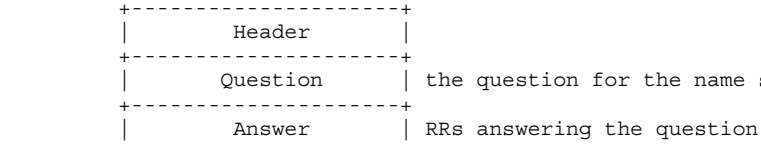
WKS RRs cause no additional section processing.

In master files, both ports and protocols are expressed using mnemonics or decimal numbers.

4. MESSAGES

4.1. Format

All communications inside of the domain protocol are carried in a single format called a message. The top level format of message is divided into 5 sections (some of which are empty in certain cases) shown below:



```
    +---------------------+
    |      Authority      | RRs pointing toward an authority
    +---------------------+
    |      Additional     | RRs holding additional information
    +---------------------+
```

The header section is always present. The header includes fields that specify which of the remaining sections are present, and also specify whether the message is a query or a response, a standard query or some other opcode, etc.

The names of the sections after the header are derived from their use in standard queries. The question section contains fields that describe a question to a name server. These fields are a query type (QTYPE), a query class (QCLASS), and a query domain name (QNAME). The last three sections have the same format: a possibly empty list of concatenated resource records (RRs). The answer section contains RRs that answer the question; the authority section contains RRs that point toward an authoritative name server; the additional records section contains RRs which relate to the query, but are not strictly answers for the question.

4.1.1. Header section format

The header contains the following fields:

```
                                    1  1  1  1  1  1
      0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                      ID                       |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    QDCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ANCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    NSCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ARCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

where:

ID — A 16 bit identifier assigned by the program that generates any kind of query. This identifier is copied the corresponding reply and can be used by the requester to match up replies to outstanding queries.

QR — A one bit field that specifies whether this message is a query (0), or a response (1).

OPCODE — A four bit field that specifies kind of query in this message. This value is set by the originator of a query and copied into the response. The values are:

0	a standard query (QUERY)
1	an inverse query (IQUERY)
2	a server status request (STATUS)
3-15	reserved for future use

AA — Authoritative Answer - this bit is valid in responses, and specifies that the responding name server is an authority for the domain name in question section.

Note that the contents of the answer section may have multiple owner names because of aliases. The AA bit corresponds to the name which matches the query name, or the first owner name in the answer section.

TC — TrunCation - specifies that this message was truncated due to length greater than that permitted on the transmission channel.

RD — Recursion Desired - this bit may be set in a query and

is copied into the response. If RD is set, it directs the name server to pursue the query recursively. Recursive query support is optional.

RA — Recursion Available - this be is set or cleared in a response, and denotes whether recursive query support is available in the name server.

Z — Reserved for future use. Must be zero in all queries and responses.

RCODE — Response code - this 4 bit field is set as part of responses. The values have the following interpretation:

0	No error condition
1	Format error - The name server was unable to interpret the query.
2	Server failure - The name server was unable to process this query due to a problem with the name server.
3	Name Error - Meaningful only for responses from an authoritative name server, this code signifies that the domain name referenced in the query does not exist.
4	Not Implemented - The name server does not support the requested kind of query.
5	Refused - The name server refuses to perform the specified operation for policy reasons. For example, a name server may not wish to provide the information to the particular requester, or a name server may not wish to perform a particular operation (e.g., zone transfer) for particular data.
6-15	Reserved for future use.

QDCOUNT — an unsigned 16 bit integer specifying the number of entries in the question section.

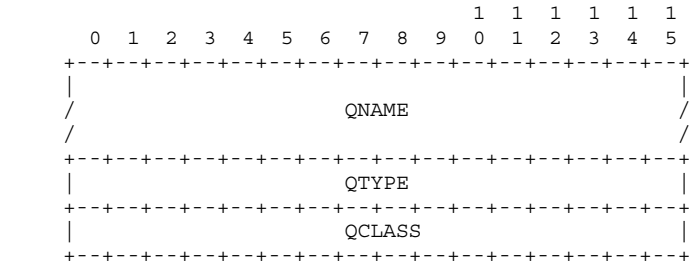
ANCOUNT — an unsigned 16 bit integer specifying the number of resource records in the answer section.

NSCOUNT — an unsigned 16 bit integer specifying the number of name server resource records in the authority records section.

ARCOUNT — an unsigned 16 bit integer specifying the number of resource records in the additional records section.

4.1.2. Question section format

The question section is used to carry the "question" in most queries, i.e., the parameters that define what is being asked. The section contains QDCOUNT (usually 1) entries, each of the following format:



where:

QNAME — a domain name represented as a sequence of labels, where

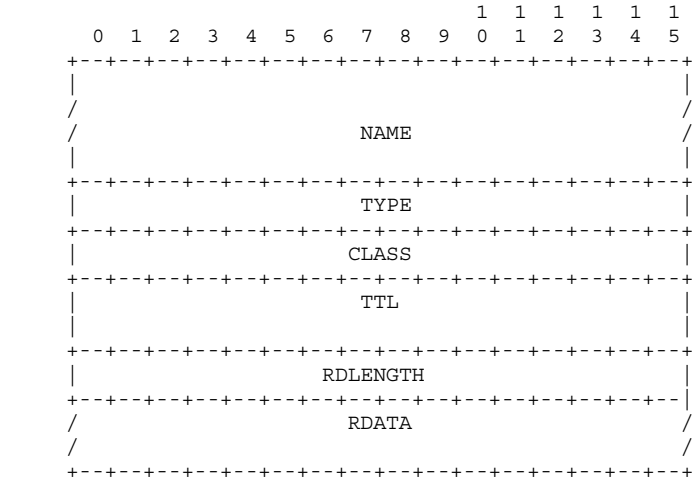
each label consists of a length octet followed by that number of octets. The domain name terminates with the zero length octet for the null label of the root. Note that this field may be an odd number of octets; no padding is used.

QTYPE a two octet code which specifies the type of the query. The values for this field include all codes valid for a TYPE field, together with some more general codes which can match more than one type of RR.

QCLASS a two octet code that specifies the class of the query. For example, the QCLASS field is IN for the Internet.

4.1.3. Resource record format

The answer, authority, and additional sections all share the same format: a variable number of resource records, where the number of records is specified in the corresponding count field in the header. Each resource record has the following format:



where:

NAME a domain name to which this resource record pertains.

TYPE two octets containing one of the RR type codes. This field specifies the meaning of the data in the RDATA field.

CLASS two octets which specify the class of the data in the RDATA field.

TTL a 32 bit unsigned integer that specifies the time interval (in seconds) that the resource record may be cached before it should be discarded. Zero values are interpreted to mean that the RR can only be used for the transaction in progress, and should not be cached.

RDLENGTH an unsigned 16 bit integer that specifies the length in octets of the RDATA field.

RDATA a variable length string of octets that describes the resource. The format of this information varies according to the TYPE and CLASS of the resource record. For example, the if the TYPE is A and the CLASS is IN, the RDATA field is a 4 octet ARPA Internet address.

G.2 RFC 1533

3.1. Pad Option

The pad option can be used to cause subsequent fields to align on word boundaries.

The code for the pad option is 0, and its length is 1 octet.

```
 Code
+-----+
|  0  |
+-----+
```

3.2. End Option

The end option marks the end of valid information in the vendor field. Subsequent octets should be filled with pad options.

The code for the end option is 255, and its length is 1 octet.

```
 Code
+-----+
| 255 |
+-----+
```

3.3. Subnet Mask

The subnet mask option specifies the client's subnet mask as per RFC 950 [5].

If both the subnet mask and the router option are specified in a DHCP reply, the subnet mask option MUST be first.

The code for the subnet mask option is 1, and its length is 4 octets.

```
 Code   Len        Subnet Mask
+-----+-----+-----+-----+-----+-----+
|  1  |  4  |  m1 |  m2 |  m3 |  m4 |
+-----+-----+-----+-----+-----+-----+
```

3.4. Time Offset

The time offset field specifies the offset of the client's subnet in seconds from Coordinated Universal Time (UTC). The offset is expressed as a signed 32-bit integer.

The code for the time offset option is 2, and its length is 4 octets.

```
 Code   Len        Time Offset
+-----+-----+-----+-----+-----+-----+
|  2  |  4  |  n1 |  n2 |  n3 |  n4 |
+-----+-----+-----+-----+-----+-----+
```

3.5. Router Option

The router option specifies a list of IP addresses for routers on the client's subnet. Routers SHOULD be listed in order of preference.

The code for the router option is 3. The minimum length for the router option is 4 octets, and the length MUST always be a multiple of 4.

```
 Code   Len         Address 1               Address 2
+-----+-----+-----+-----+-----+-----+-----+-----+--
|  3  |  n  |  a1 |  a2 |  a3 |  a4 |  a1 |  a2 |  ...
+-----+-----+-----+-----+-----+-----+-----+-----+--
```

3.6. Time Server Option

The time server option specifies a list of RFC 868 [6] time servers available to the client. Servers SHOULD be listed in order of preference.

The code for the time server option is 4. The minimum length for this option is 4 octets, and the length MUST always be a multiple of 4.

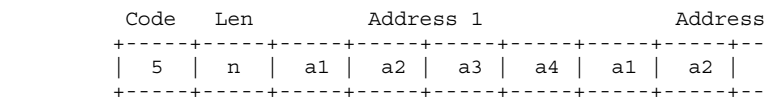
```
 Code   Len         Address 1               Address 2
+-----+-----+-----+-----+-----+-----+-----+-----+--
|  4  |  n  |  a1 |  a2 |  a3 |  a4 |  a1 |  a2 |  ...
+-----+-----+-----+-----+-----+-----+-----+-----+--
```

3.7. Name Server Option

The name server option specifies a list of IEN 116 [7] name servers

available to the client. Servers SHOULD be listed in order of preference.

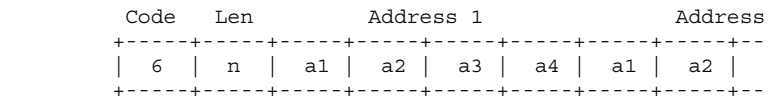
The code for the name server option is 5. The minimum length for this option is 4 octets, and the length MUST always be a multiple of 4.



3.8. Domain Name Server Option

The domain name server option specifies a list of Domain Name System (STD 13, RFC 1035 [8]) name servers available to the client. Servers SHOULD be listed in order of preference.

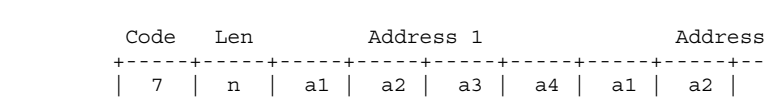
The code for the domain name server option is 6. The minimum length for this option is 4 octets, and the length MUST always be a multiple of 4.



3.9. Log Server Option

The log server option specifies a list of MIT-LCS UDP log servers available to the client. Servers SHOULD be listed in order of preference.

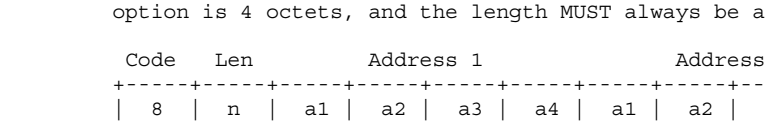
The code for the log server option is 7. The minimum length for this option is 4 octets, and the length MUST always be a multiple of 4.



3.10. Cookie Server Option

The cookie server option specifies a list of RFC 865 [9] cookie servers available to the client. Servers SHOULD be listed in order of preference.

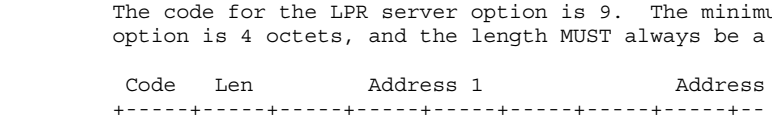
The code for the log server option is 8. The minimum length for this option is 4 octets, and the length MUST always be a multiple of 4.



3.11. LPR Server Option

The LPR server option specifies a list of RFC 1179 [10] line printer servers available to the client. Servers SHOULD be listed in order of preference.

The code for the LPR server option is 9. The minimum length for this option is 4 octets, and the length MUST always be a multiple of 4.



3.12. Impress Server Option

The Impress server option specifies a list of Imagen Impress servers available to the client. Servers SHOULD be listed in order of preference.

The code for the Impress server option is 10. The minimum length for this option is 4 octets, and the length MUST always be a multiple of 4.

```
 Code   Len         Address 1               Address 2
+-----+-----+-----+-----+-----+-----+-----+-----+--
|  10 |  n  |  a1 |  a2 |  a3 |  a4 |  a1 |  a2 |  ...
+-----+-----+-----+-----+-----+-----+-----+-----+--
```

3.13. Resource Location Server Option

This option specifies a list of RFC 887 [11] Resource Location servers available to the client. Servers SHOULD be listed in order of preference.

The code for this option is 11. The minimum length for this option is 4 octets, and the length MUST always be a multiple of 4.

```
 Code   Len         Address 1               Address 2
+-----+-----+-----+-----+-----+-----+-----+-----+--
|  11 |  n  |  a1 |  a2 |  a3 |  a4 |  a1 |  a2 |  ...
+-----+-----+-----+-----+-----+-----+-----+-----+--
```

3.14. Host Name Option

This option specifies the name of the client. The name may or may not be qualified with the local domain name (see section 3.17 for the preferred way to retrieve the domain name). See RFC 1035 for character set restrictions.

The code for this option is 12, and its minimum length is 1.

```
 Code   Len                 Host Name
+-----+-----+-----+-----+-----+-----+-----+-----+--
|  12 |  n  |  h1 |  h2 |  h3 |  h4 |  h5 |  h6 |  ...
+-----+-----+-----+-----+-----+-----+-----+-----+--
```

3.15. Boot File Size Option

This option specifies the length in 512-octet blocks of the default boot image for the client. The file length is specified as an unsigned 16-bit integer.

The code for this option is 13, and its length is 2.

```
 Code   Len   File Size
+-----+-----+-----+-----+
|  13 |  2  |  l1 |  l2 |
+-----+-----+-----+-----+
```

3.16. Merit Dump File

This option specifies the path-name of a file to which the client's core image should be dumped in the event the client crashes. The path is formatted as a character string consisting of characters from the NVT ASCII character set.

The code for this option is 14. Its minimum length is 1.

```
 Code   Len      Dump File Pathname
+-----+-----+-----+-----+-----+-----+---
|  14 |  n  |  n1 |  n2 |  n3 |  n4 | ...
+-----+-----+-----+-----+-----+-----+---
```

3.17. Domain Name

This option specifies the domain name that client should use when resolving hostnames via the Domain Name System.

The code for this option is 15. Its minimum length is 1.

```
 Code   Len        Domain Name
+-----+-----+-----+-----+-----+-----+--
|  15 |  n  |  d1 |  d2 |  d3 |  d4 |  ...
+-----+-----+-----+-----+-----+-----+--
```

3.18. Swap Server

This specifies the IP address of the client's swap server.

The code for this option is 16 and its length is 4.

```
 Code   Len    Swap Server Address
+-----+-----+-----+-----+-----+-----+
|  16 |  n  |  a1 |  a2 |  a3 |  a4 |
+-----+-----+-----+-----+-----+-----+
```

3.19. Root Path

This option specifies the path-name that contains the client's root disk. The path is formatted as a character string consisting of characters from the NVT ASCII character set.

The code for this option is 17. Its minimum length is 1.

```
 Code   Len      Root Disk Pathname
+-----+-----+-----+-----+-----+-----+---
|  17 |  n  |  n1 |  n2 |  n3 |  n4 | ...
+-----+-----+-----+-----+-----+-----+---
```

3.20. Extensions Path

A string to specify a file, retrievable via TFTP, which contains information which can be interpreted in the same way as the 64-octet vendor-extension field within the BOOTP response, with the following exceptions:

- the length of the file is unconstrained;
- all references to Tag 18 (i.e., instances of the BOOTP Extensions Path field) within the file are ignored.

The code for this option is 18. Its minimum length is 1.

```
 Code   Len      Extensions Pathname
+-----+-----+-----+-----+-----+-----+---
|  18 |  n  |  n1 |  n2 |  n3 |  n4 | ...
+-----+-----+-----+-----+-----+-----+---
```

4. IP Layer Parameters per Host

This section details the options that affect the operation of the IP layer on a per-host basis.

4.1. IP Forwarding Enable/Disable Option

This option specifies whether the client should configure its IP layer for packet forwarding. A value of 0 means disable IP forwarding, and a value of 1 means enable IP forwarding.

The code for this option is 19, and its length is 1.

```
 Code   Len  Value
+-----+-----+-----+
|  19 |  1  | 0/1 |
+-----+-----+-----+
```

4.2. Non-Local Source Routing Enable/Disable Option

This option specifies whether the client should configure its IP layer to allow forwarding of datagrams with non-local source routes (see Section 3.3.5 of [4] for a discussion of this topic). A value of 0 means disallow forwarding of such datagrams, and a value of 1 means allow forwarding.

The code for this option is 20, and its length is 1.

```
 Code   Len  Value
+-----+-----+-----+
|  20 |  1  | 0/1 |
+-----+-----+-----+
```

4.3. Policy Filter Option

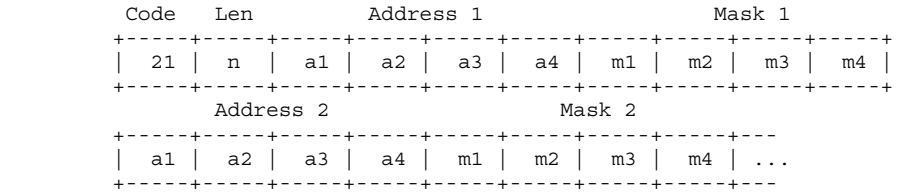
This option specifies policy filters for non-local source routing. The filters consist of a list of IP addresses and masks which specify destination/mask pairs with which to filter incoming source routes.

Any source routed datagram whose next-hop address does not match one

of the filters should be discarded by the client.

See [4] for further information.

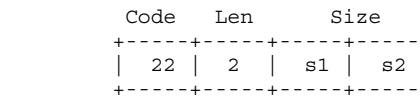
The code for this option is 21. The minimum length of this option is 8, and the length MUST be a multiple of 8.



4.4. Maximum Datagram Reassembly Size

This option specifies the maximum size datagram that the client should be prepared to reassemble. The size is specified as a 16-bit unsigned integer. The minimum value legal value is 576.

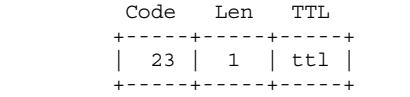
The code for this option is 22, and its length is 2.



4.5. Default IP Time-to-live

This option specifies the default time-to-live that the client should use on outgoing datagrams. The TTL is specified as an octet with a value between 1 and 255.

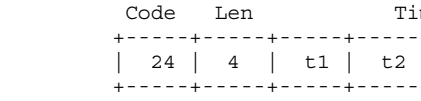
The code for this option is 23, and its length is 1.



4.6. Path MTU Aging Timeout Option

This option specifies the timeout (in seconds) to use when aging Path MTU values discovered by the mechanism defined in RFC 1191 [12]. The timeout is specified as a 32-bit unsigned integer.

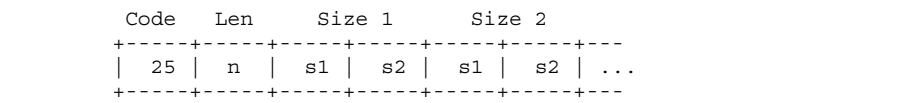
The code for this option is 24, and its length is 4.



4.7. Path MTU Plateau Table Option

This option specifies a table of MTU sizes to use when performing Path MTU Discovery as defined in RFC 1191. The table is formatted as a list of 16-bit unsigned integers, ordered from smallest to largest. The minimum MTU value cannot be smaller than 68.

The code for this option is 25. Its minimum length is 2, and the length MUST be a multiple of 2.



5. IP Layer Parameters per Interface

This section details the options that affect the operation of the IP layer on a per-interface basis. It is expected that a client can issue multiple requests, one per interface, in order to configure interfaces with their specific parameters.

5.1. Interface MTU Option

This option specifies the MTU to use on this interface. The MTU is specified as a 16-bit unsigned integer. The minimum legal value for the MTU is 68.

The code for this option is 26, and its length is 2.

Code	Len	MTU	
26	2	m1	m2

5.2. All Subnets are Local Option

This option specifies whether or not the client may assume that all subnets of the IP network to which the client is connected use the same MTU as the subnet of that network to which the client is directly connected. A value of 1 indicates that all subnets share the same MTU. A value of 0 means that the client should assume that some subnets of the directly connected network may have smaller MTUs.

The code for this option is 27, and its length is 1.

Code	Len	Value
27	1	0/1

5.3. Broadcast Address Option

This option specifies the broadcast address in use on the client's subnet. Legal values for broadcast addresses are specified in section 3.2.1.3 of [4].

The code for this option is 28, and its length is 4.

Code	Len	Broadcast Address			
28	4	b1	b2	b3	b4

5.4. Perform Mask Discovery Option

This option specifies whether or not the client should perform subnet mask discovery using ICMP. A value of 0 indicates that the client should not perform mask discovery. A value of 1 means that the client should perform mask discovery.

The code for this option is 29, and its length is 1.

Code	Len	Value
29	1	0/1

5.5. Mask Supplier Option

This option specifies whether or not the client should respond to subnet mask requests using ICMP. A value of 0 indicates that the client should not respond. A value of 1 means that the client should respond.

The code for this option is 30, and its length is 1.

Code	Len	Value
30	1	0/1

5.6. Perform Router Discovery Option

This option specifies whether or not the client should solicit routers using the Router Discovery mechanism defined in RFC 1256 [13]. A value of 0 indicates that the client should not perform router discovery. A value of 1 means that the client should perform router discovery.

The code for this option is 31, and its length is 1.

```
 Code   Len  Value
+-----+-----+-----+
|  31 |  1  | 0/1 |
+-----+-----+-----+
```

5.7. Router Solicitation Address Option

This option specifies the address to which the client should transmit router solicitation requests.

The code for this option is 32, and its length is 4.

```
 Code   Len            Address
+-----+-----+-----+-----+-----+-----+
|  32 |  4  |  a1 |  a2 |  a3 |  a4 |
+-----+-----+-----+-----+-----+-----+
```

5.8. Static Route Option

This option specifies a list of static routes that the client should install in its routing cache. If multiple routes to the same destination are specified, they are listed in descending order of priority.

The routes consist of a list of IP address pairs. The first address is the destination address, and the second address is the router for the destination.

The default route (0.0.0.0) is an illegal destination for a static route. See section 3.5 for information about the router option.

The code for this option is 33. The minimum length of this option is 8, and the length MUST be a multiple of 8.

```
 Code   Len         Destination 1           Router 1
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|  33 |  n  |  d1 |  d2 |  d3 |  d4 |  r1 |  r2 |  r3 |  r4 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
        Destination 2           Router 2
+-----+-----+-----+-----+-----+-----+-----+-----+---
|  d1 |  d2 |  d3 |  d4 |  r1 |  r2 |  r3 |  r4 | ...
+-----+-----+-----+-----+-----+-----+-----+-----+---
```

6. Link Layer Parameters per Interface

This section lists the options that affect the operation of the data link layer on a per-interface basis.

6.1. Trailer Encapsulation Option

This option specifies whether or not the client should negotiate the use of trailers (RFC 893 [14]) when using the ARP protocol. A value of 0 indicates that the client should not attempt to use trailers. A value of 1 means that the client should attempt to use trailers.

The code for this option is 34, and its length is 1.

```
 Code   Len  Value
+-----+-----+-----+
|  34 |  1  | 0/1 |
+-----+-----+-----+
```

6.2. ARP Cache Timeout Option

This option specifies the timeout in seconds for ARP cache entries. The time is specified as a 32-bit unsigned integer.

The code for this option is 35, and its length is 4.

```
 Code   Len           Time
+-----+-----+-----+-----+-----+-----+
|  35 |  4  |  t1 |  t2 |  t3 |  t4 |
+-----+-----+-----+-----+-----+-----+
```

6.3. Ethernet Encapsulation Option

This option specifies whether or not the client should use Ethernet

Version 2 (RFC 894 [15]) or IEEE 802.3 (RFC 1042 [16]) encapsulation if the interface is an Ethernet. A value of 0 indicates that the client should use RFC 894 encapsulation. A value of 1 means that the client should use RFC 1042 encapsulation.

The code for this option is 36, and its length is 1.

Code	Len	Value
36	1	0/1

7. TCP Parameters

This section lists the options that affect the operation of the TCP layer on a per-interface basis.

7.1. TCP Default TTL Option

This option specifies the default TTL that the client should use when sending TCP segments. The value is represented as an 8-bit unsigned integer. The minimum value is 1.

The code for this option is 37, and its length is 1.

Code	Len	TTL
37	1	n

7.2. TCP Keepalive Interval Option

This option specifies the interval (in seconds) that the client TCP should wait before sending a keepalive message on a TCP connection. The time is specified as a 32-bit unsigned integer. A value of zero indicates that the client should not generate keepalive messages on connections unless specifically requested by an application.

The code for this option is 38, and its length is 4.

Code	Len	Time			
38	4	t1	t2	t3	t4

7.3. TCP Keepalive Garbage Option

This option specifies the whether or not the client should send TCP keepalive messages with a octet of garbage for compatibility with older implementations. A value of 0 indicates that a garbage octet should not be sent. A value of 1 indicates that a garbage octet should be sent.

The code for this option is 39, and its length is 1.

Code	Len	Value
39	1	0/1

8. Application and Service Parameters

This section details some miscellaneous options used to configure miscellaneous applications and services.

8.1. Network Information Service Domain Option

This option specifies the name of the client's NIS [17] domain. The domain is formatted as a character string consisting of characters from the NVT ASCII character set.

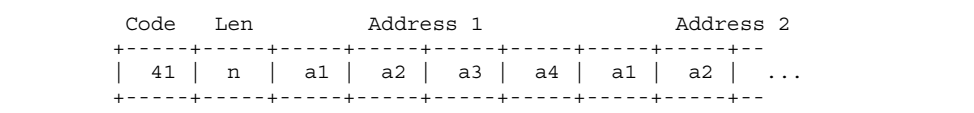
The code for this option is 40. Its minimum length is 1.

Code	Len	NIS Domain Name				
40	n	n1	n2	n3	n4	...

8.2. Network Information Servers Option

This option specifies a list of IP addresses indicating NIS servers available to the client. Servers SHOULD be listed in order of preference.

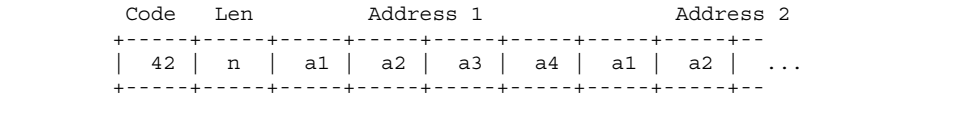
The code for this option is 41. Its minimum length is 4, and the length MUST be a multiple of 4.



8.3. Network Time Protocol Servers Option

This option specifies a list of IP addresses indicating NTP [18] servers available to the client. Servers SHOULD be listed in order of preference.

The code for this option is 42. Its minimum length is 4, and the length MUST be a multiple of 4.



8.4. Vendor Specific Information

This option is used by clients and servers to exchange vendor-specific information. The information is an opaque object of n octets, presumably interpreted by vendor-specific code on the clients and servers. The definition of this information is vendor specific. The vendor is indicated in the class-identifier option. Servers not equipped to interpret the vendor-specific information sent by a client MUST ignore it (although it may be reported). Clients which do not receive desired vendor-specific information SHOULD make an attempt to operate without it, although they may do so (and announce they are doing so) in a degraded mode.

If a vendor potentially encodes more than one item of information in this option, then the vendor SHOULD encode the option using "Encapsulated vendor-specific options" as described below:

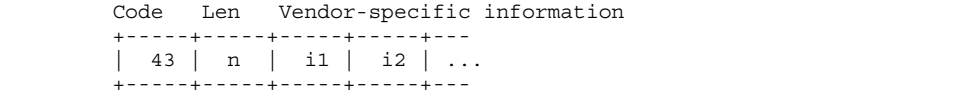
The Encapsulated vendor-specific options field SHOULD be encoded as a sequence of code/length/value fields of identical syntax to the DHCP options field with the following exceptions:

1) There SHOULD NOT be a "magic cookie" field in the encapsulated vendor-specific extensions field.

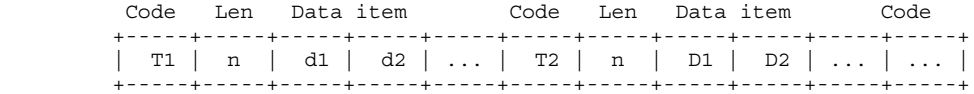
2) Codes other than 0 or 255 MAY be redefined by the vendor within the encapsulated vendor-specific extensions field, but SHOULD conform to the tag-length-value syntax defined in section 2.

3) Code 255 (END), if present, signifies the end of the encapsulated vendor extensions, not the end of the vendor extensions field. If no code 255 is present, then the end of the enclosing vendor-specific information field is taken as the end of the encapsulated vendor-specific extensions field.

The code for this option is 43 and its minimum length is 1.



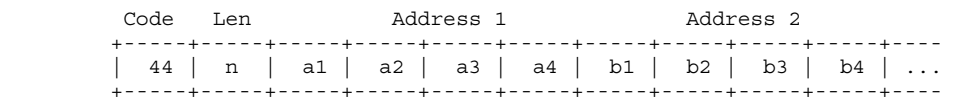
When encapsulated vendor-specific extensions are used, the information bytes 1-n have the following format:



8.5. NetBIOS over TCP/IP Name Server Option

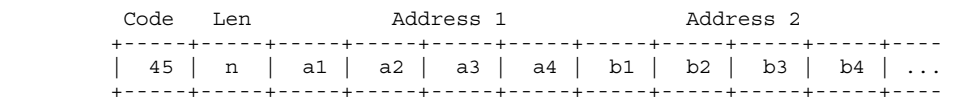
The NetBIOS name server (NBNS) option specifies a list of RFC 1001/1002 [19] [20] NBNS name servers listed in order of preference.

The code for this option is 44. The minimum length of the option is 4 octets, and the length must always be a multiple of 4.



8.6. NetBIOS over TCP/IP Datagram Distribution Server Option

The NetBIOS datagram distribution server (NBDD) option specifies a list of RFC 1001/1002 NBDD servers listed in order of preference. The code for this option is 45. The minimum length of the option is 4 octets, and the length must always be a multiple of 4.



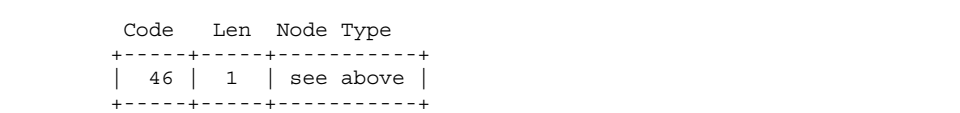
8.7. NetBIOS over TCP/IP Node Type Option

The NetBIOS node type option allows NetBIOS over TCP/IP clients which are configurable to be configured as described in RFC 1001/1002. The value is specified as a single octet which identifies the client type as follows:

Value	Node Type
0x1	B-node
0x2	P-node
0x4	M-node
0x8	H-node

In the above chart, the notation '0x' indicates a number in base-16 (hexadecimal).

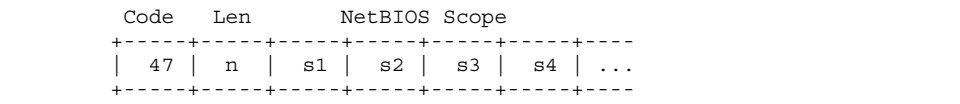
The code for this option is 46. The length of this option is always 1.



8.8. NetBIOS over TCP/IP Scope Option

The NetBIOS scope option specifies the NetBIOS over TCP/IP scope parameter for the client as specified in RFC 1001/1002. See [19], [20], and [8] for character-set restrictions.

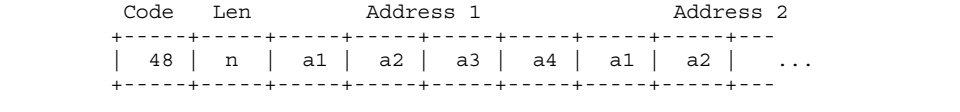
The code for this option is 47. The minimum length of this option is 1.



8.9. X Window System Font Server Option

This option specifies a list of X Window System [21] Font servers available to the client. Servers SHOULD be listed in order of preference.

The code for this option is 48. The minimum length of this option is 4 octets, and the length MUST be a multiple of 4.



8.10. X Window System Display Manager Option

This option specifies a list of IP addresses of systems that are running the X Window System Display Manager and are available to the client.

Addresses SHOULD be listed in order of preference.

The code for the this option is 49. The minimum length of this option is 4, and the length MUST be a multiple of 4.

Code	Len	Address 1			Address 2			
49	n	a1	a2	a3	a4	a1	a2	...

9. DHCP Extensions

This section details the options that are specific to DHCP.

9.1. Requested IP Address

This option is used in a client request (DHCPDISCOVER) to allow the client to request that a particular IP address be assigned.

The code for this option is 50, and its length is 4.

Code	Len	Address			
50	4	a1	a2	a3	a4

9.2. IP Address Lease Time

This option is used in a client request (DHCPDISCOVER or DHCPREQUEST) to allow the client to request a lease time for the IP address. In a server reply (DHCPOFFER), a DHCP server uses this option to specify the lease time it is willing to offer.

The time is in units of seconds, and is specified as a 32-bit unsigned integer.

The code for this option is 51, and its length is 4.

Code	Len	Lease Time			
51	4	t1	t2	t3	t4

9.3. Option Overload

This option is used to indicate that the DHCP "sname" or "file" fields are being overloaded by using them to carry DHCP options. A DHCP server inserts this option if the returned parameters will exceed the usual space allotted for options.

If this option is present, the client interprets the specified additional fields after it concludes interpretation of the standard option fields.

The code for this option is 52, and its length is 1. Legal values for this option are:

Value	Meaning
1	the "file" field is used to hold options
2	the "sname" field is used to hold options
3	both fields are used to hold options

Code	Len	Value
52	1	1/2/3

9.4. DHCP Message Type

This option is used to convey the type of the DHCP message. The code for this option is 53, and its length is 1. Legal values for this

option are:

Value	Message Type
1	DHCPDISCOVER
2	DHCPOFFER
3	DHCPREQUEST
4	DHCPDECLINE
5	DHCPACK
6	DHCPNAK
7	DHCPRELEASE

Code	Len	Type
53	1	1-7

9.5. Server Identifier

This option is used in DHCPOFFER and DHCPREQUEST messages, and may optionally be included in the DHCPACK and DHCPNAK messages. DHCP servers include this option in the DHCPOFFER in order to allow the client to distinguish between lease offers. DHCP clients indicate which of several lease offers is being accepted by including this option in a DHCPREQUEST message.

The identifier is the IP address of the selected server.
The code for this option is 54, and its length is 4.

Code	Len	Address			
54	4	a1	a2	a3	a4

9.6. Parameter Request List

This option is used by a DHCP client to request values for specified configuration parameters. The list of requested parameters is specified as n octets, where each octet is a valid DHCP option code as defined in this document.

The client MAY list the options in order of preference. The DHCP server is not required to return the options in the requested order, but MUST try to insert the requested options in the order requested by the client.

The code for this option is 55. Its minimum length is 1.

Code	Len	Option Codes		
55	n	c1	c2	...

9.7. Message

This option is used by a DHCP server to provide an error message to a DHCP client in a DHCPNAK message in the event of a failure. A client may use this option in a DHCPDECLINE message to indicate the why the client declined the offered parameters. The message consists of n octets of NVT ASCII text, which the client may display on an available output device.

The code for this option is 56 and its minimum length is 1.

Code	Len	Text		
56	n	c1	c2	...

9.8. Maximum DHCP Message Size

This option specifies the maximum length DHCP message that it is willing to accept. The length is specified as an unsigned 16-bit integer. A client may use the maximum DHCP message size option in DHCPDISCOVER or DHCPREQUEST messages, but should not use the option in DHCPDECLINE messages.

The code for this option is 57, and its length is 2. The minimum

legal value is 576 octets.

Code	Len	Length	
57	2	l1	l2

9.9. Renewal (T1) Time Value

This option specifies the time interval from address assignment until the client transitions to the RENEWING state.

The value is in units of seconds, and is specified as a 32-bit unsigned integer.

The code for this option is 58, and its length is 4.

Code	Len	T1 Interval			
58	4	t1	t2	t3	t4

9.10. Rebinding (T2) Time Value

This option specifies the time interval from address assignment until the client transitions to the REBINDING state.

The value is in units of seconds, and is specified as a 32-bit unsigned integer.

The code for this option is 59, and its length is 4.

Code	Len	T2 Interval			
59	4	t1	t2	t3	t4

9.11. Class-identifier

This option is used by DHCP clients to optionally identify the type and configuration of a DHCP client. The information is a string of n octets, interpreted by servers. Vendors and sites may choose to define specific class identifiers to convey particular configuration or other identification information about a client. For example, the identifier may encode the client's hardware configuration. Servers not equipped to interpret the class-specific information sent by a client MUST ignore it (although it may be reported).

The code for this option is 60, and its minimum length is 1.

Code	Len	Class-Identifier		
60	n	i1	i2	...

9.12. Client-identifier

This option is used by DHCP clients to specify their unique identifier. DHCP servers use this value to index their database of address bindings. This value is expected to be unique for all clients in an administrative domain.

Identifiers consist of a type-value pair, similar to the

It is expected that this field will typically contain a hardware type and hardware address, but this is not required. Current legal values for hardware types are defined in [22].

The code for this option is 61, and its minimum length is 2.

Code	Len	Type	Client-Identifier		
61	n	t1	i1	i2	...

G.3 RFC 2131

2. Protocol Summary

From the client's point of view, DHCP is an extension of the BOOTP mechanism. This behavior allows existing BOOTP clients to interoperate with DHCP servers without requiring any change to the clients' initialization software. RFC 1542 [2] details the interactions between BOOTP and DHCP clients and servers [9]. There are some new, optional transactions that optimize the interaction between DHCP clients and servers that are described in sections 3 and 4.

Figure 1 gives the format of a DHCP message and table 1 describes each of the fields in the DHCP message. The numbers in parentheses indicate the size of each field in octets. The names for the fields given in the figure will be used throughout this document to refer to the fields in DHCP messages.

There are two primary differences between DHCP and BOOTP. First, DHCP defines mechanisms through which clients can be assigned a network address for a finite lease, allowing for serial reassignment of network addresses to different clients. Second, DHCP provides the mechanism for a client to acquire all of the IP configuration parameters that it needs in order to operate.

DHCP introduces a small change in terminology intended to clarify the meaning of one of the fields. What was the "vendor extensions" field in BOOTP has been re-named the "options" field in DHCP. Similarly, the tagged data items that were used inside the BOOTP "vendor extensions" field, which were formerly referred to as "vendor extensions," are now termed simply "options."

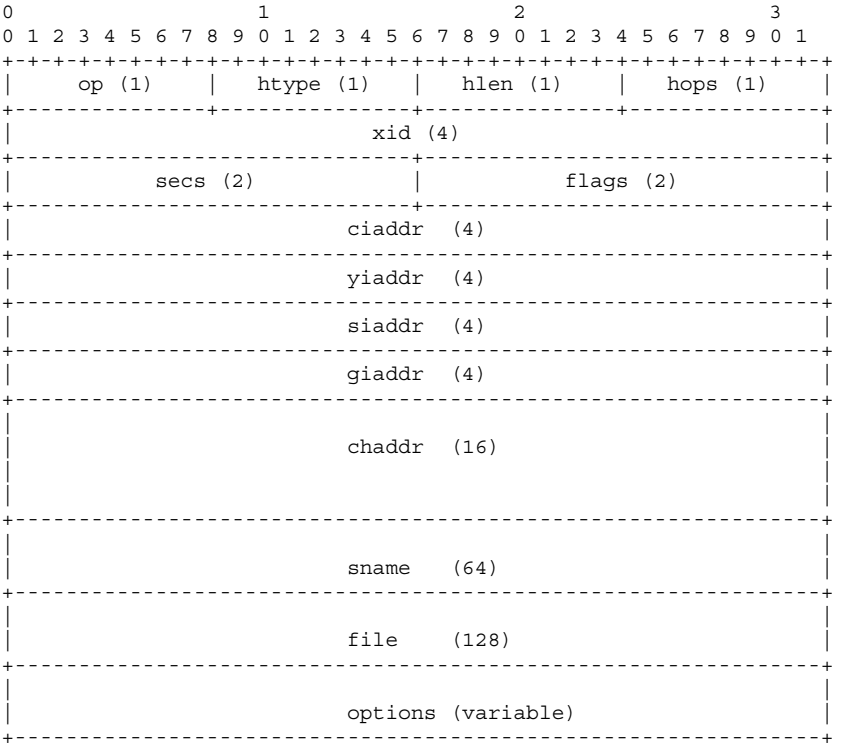


Figure 1: Format of a DHCP message

DHCP defines a new 'client identifier' option that is used to pass an explicit client identifier to a DHCP server. This change eliminates the overloading of the 'chaddr' field in BOOTP messages, where 'chaddr' is used both as a hardware address for transmission of BOOTP reply messages and as a client identifier. The 'client identifier' is an opaque key, not to be interpreted by the server; for example, the 'client identifier' may contain a hardware address, identical to the contents of the 'chaddr' field, or it may contain another type of identifier, such as a DNS name. The 'client identifier' chosen by a DHCP client MUST be unique to that client within the subnet to which

the client is attached. If the client uses a 'client identifier' in one message, it MUST use that same identifier in all subsequent messages, to ensure that all servers correctly identify the client.

DHCP clarifies the interpretation of the 'siaddr' field as the address of the server to use in the next step of the client's bootstrap process. A DHCP server may return its own address in the 'siaddr' field, if the server is prepared to supply the next bootstrap service (e.g., delivery of an operating system executable image). A DHCP server always returns its own address in the 'server identifier' option.

FIELD	OCTETS	DESCRIPTION
op	1	Message op code / message type. 1 = BOOTREQUEST, 2 = BOOTREPLY
htype	1	Hardware address type, see ARP section in "Assigned Numbers" RFC; e.g., '1' = 10mb ethernet.
hlen	1	Hardware address length (e.g. '6' for 10mb ethernet).
hops	1	Client sets to zero, optionally used by relay agents when booting via a relay agent.
xid	4	Transaction ID, a random number chosen by the client, used by the client and server to associate messages and responses between a client and a server.
secs	2	Filled in by client, seconds elapsed since client began address acquisition or renewal process.
flags	2	Flags (see figure 2).
ciaddr	4	Client IP address; only filled in if client is in BOUND, RENEW or REBINDING state and can respond to ARP requests.
yiaddr	4	'your' (client) IP address.
siaddr	4	IP address of next server to use in bootstrap; returned in DHCPOFFER, DHCPACK by server.
giaddr	4	Relay agent IP address, used in booting via a relay agent.
chaddr	16	Client hardware address.
sname	64	Optional server host name, null terminated string.
file	128	Boot file name, null terminated string; "generic" name or null in DHCPDISCOVER, fully qualified directory-path name in DHCPOFFER.
options	var	Optional parameters field. See the options documents for a list of defined options.

Table 1: Description of fields in a DHCP message

The 'options' field is now variable length. A DHCP client must be prepared to receive DHCP messages with an 'options' field of at least length 312 octets. This requirement implies that a DHCP client must be prepared to receive a message of up to 576 octets, the minimum IP datagram size an IP host must be prepared to accept [3]. DHCP clients may negotiate the use of larger DHCP messages through the 'maximum DHCP message size' option. The options field may be further extended into the 'file' and 'sname' fields.

In the case of a client using DHCP for initial configuration (before the client's TCP/IP software has been completely configured), DHCP requires creative use of the client's TCP/IP software and liberal interpretation of RFC 1122. The TCP/IP software SHOULD accept and forward to the IP layer any IP packets delivered to the client's hardware address before the IP address is configured; DHCP servers and BOOTP relay agents may not be able to deliver DHCP messages to clients that cannot accept hardware unicast datagrams before the TCP/IP software is configured.

To work around some clients that cannot accept IP unicast datagrams before the TCP/IP software is configured as discussed in the previous paragraph, DHCP uses the 'flags' field [21]. The leftmost bit is defined as the BROADCAST (B) flag. The semantics of this flag are discussed in section 4.1 of this document. The remaining bits of the flags field are reserved for future use. They MUST be set to zero by clients and ignored by servers and relay agents. Figure 2 gives the format of the 'flags' field.

```
                                1 1 1 1 1 1
            0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|B|            MBZ              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

B: BROADCAST flag

MBZ: MUST BE ZERO (reserved for future use)

Figure 2: Format of the 'flags' field

G.4 RFC 3315

6. Client/Server Message Formats

All DHCP messages sent between clients and servers share an identical fixed format header and a variable format area for options.

All values in the message header and in options are in network byte order.

Options are stored serially in the options field, with no padding between the options. Options are byte-aligned but are not aligned in any other way such as on 2 or 4 byte boundaries.

The following diagram illustrates the format of DHCP messages sent between clients and servers:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    msg-type   |               transaction-id                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                            options                            .
.                           (variable)                          .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

msg-type	Identifies the DHCP message type; the available message types are listed in section 5.3.
transaction-id	The transaction ID for this message exchange.
options	Options carried in this message; options are described in section 22.

7. Relay Agent/Server Message Formats

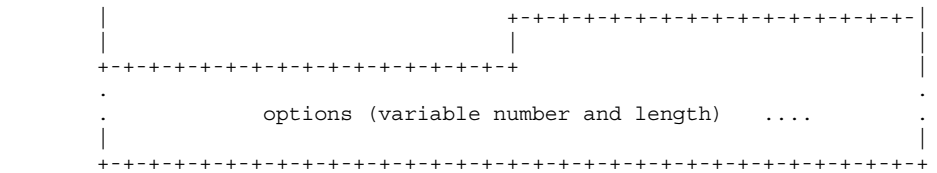
Relay agents exchange messages with servers to relay messages between clients and servers that are not connected to the same link.

All values in the message header and in options are in network byte order.

Options are stored serially in the options field, with no padding between the options. Options are byte-aligned but are not aligned in any other way such as on 2 or 4 byte boundaries.

There are two relay agent messages, which share the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    msg-type   |   hop-count   |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
|                         link-address                          |
|                                                               |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
|                         peer-address                          |
|                                                               |
```



22. DHCP Options

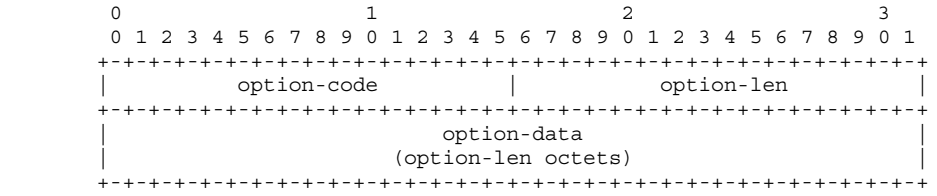
Options are used to carry additional information and parameters in DHCP messages. Every option shares a common base format, as described in section 22.1. All values in options are represented in network byte order.

This document describes the DHCP options defined as part of the base DHCP specification. Other options may be defined in the future in separate documents.

Unless otherwise noted, each option may appear only in the options area of a DHCP message and may appear only once. If an option does appear multiple times, each instance is considered separate and the data areas of the options MUST NOT be concatenated or otherwise combined.

22.1. Format of DHCP Options

The format of DHCP options is:



option-code An unsigned integer identifying the specific option type carried in this option.

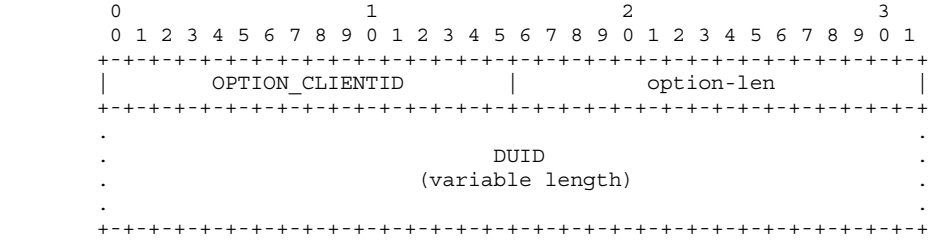
option-len An unsigned integer giving the length of the option-data field in this option in octets.

option-data The data for the option; the format of this data depends on the definition of the option.

DHCPv6 options are scoped by using encapsulation. Some options apply generally to the client, some are specific to an IA, and some are specific to the addresses within an IA. These latter two cases are discussed in sections 22.4 and 22.6.

22.2. Client Identifier Option

The Client Identifier option is used to carry a DUID (see section 9) identifying a client between a client and a server. The format of the Client Identifier option is:



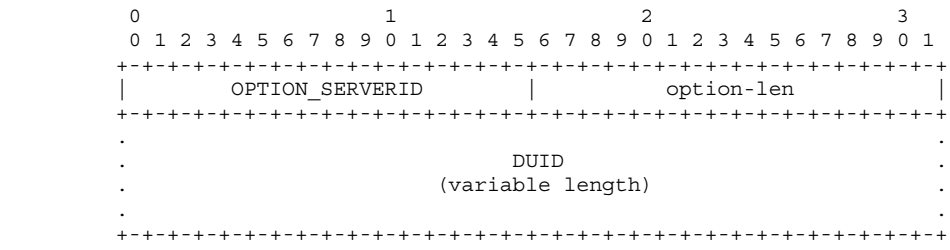
option-code OPTION_CLIENTID (1).

option-len Length of DUID in octets.

DUID The DUID for the client.

22.3. Server Identifier Option

The Server Identifier option is used to carry a DUID (see section 9) identifying a server between a client and a server. The format of the Server Identifier option is:



option-code OPTION_SERVERID (2).

option-len Length of DUID in octets.

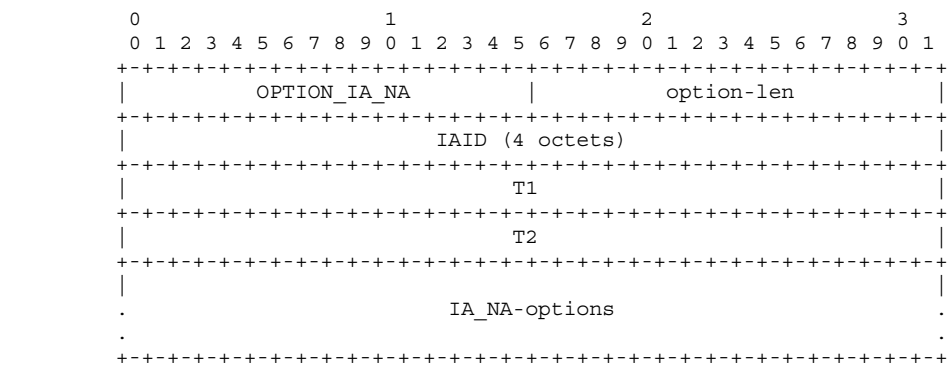
DUID The DUID for the server.

22.4. Identity Association for Non-temporary Addresses Option

The Identity Association for Non-temporary Addresses option (IA_NA option) is used to carry an IA_NA, the parameters associated with the IA_NA, and the non-temporary addresses associated with the IA_NA.

Addresses appearing in an IA_NA option are not temporary addresses (see section 22.5).

The format of the IA_NA option is:



option-code OPTION_IA_NA (3).

option-len 12 + length of IA_NA-options field.

IAID The unique identifier for this IA_NA; the IAID must be unique among the identifiers for all of this client's IA_NAs. The number space for IA_NA IAIDs is separate from the number space for IA_TA IAIDs.

T1 The time at which the client contacts the server from which the addresses in the IA_NA were obtained to extend the lifetimes of the addresses assigned to the IA_NA; T1 is a time duration relative to the current time expressed in units of seconds.

T2 The time at which the client contacts any available server to extend the lifetimes of the addresses assigned to the IA_NA; T2 is a time duration relative to the current time expressed in units of seconds.

IA_NA-options Options associated with this IA_NA.

The IA_NA-options field encapsulates those options that are specific to this IA_NA. For example, all of the IA Address Options carrying the addresses associated with this IA_NA are in the IA_NA-options field.

An IA_NA option may only appear in the options area of a DHCP message. A DHCP message may contain multiple IA_NA options.

The status of any operations involving this IA_NA is indicated in a Status Code option in the IA_NA-options field.

Note that an IA_NA has no explicit "lifetime" or "lease length" of its own. When the valid lifetimes of all of the addresses in an IA_NA have expired, the IA_NA can be considered as having expired. T1 and T2 are included to give servers explicit control over when a client recontacts the server about a specific IA_NA.

In a message sent by a client to a server, values in the T1 and T2 fields indicate the client's preference for those parameters. The client sets T1 and T2 to 0 if it has no preference for those values. In a message sent by a server to a client, the client MUST use the values in the T1 and T2 fields for the T1 and T2 parameters, unless those values in those fields are 0. The values in the T1 and T2 fields are the number of seconds until T1 and T2.

The server selects the T1 and T2 times to allow the client to extend the lifetimes of any addresses in the IA_NA before the lifetimes expire, even if the server is unavailable for some short period of time. Recommended values for T1 and T2 are .5 and .8 times the shortest preferred lifetime of the addresses in the IA that the server is willing to extend, respectively. If the "shortest" preferred lifetime is 0xffffffff ("infinity"), the recommended T1 and T2 values are also 0xffffffff. If the time at which the addresses in an IA_NA are to be renewed is to be left to the discretion of the client, the server sets T1 and T2 to 0.

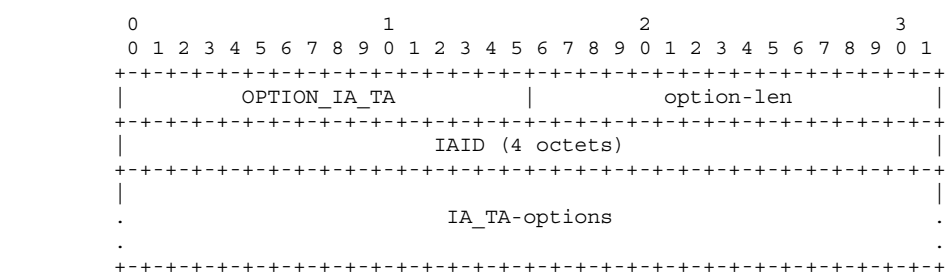
If a server receives an IA_NA with T1 greater than T2, and both T1 and T2 are greater than 0, the server ignores the invalid values of T1 and T2 and processes the IA_NA as though the client had set T1 and T2 to 0.

If a client receives an IA_NA with T1 greater than T2, and both T1 and T2 are greater than 0, the client discards the IA_NA option and processes the remainder of the message as though the server had not included the invalid IA_NA option.

Care should be taken in setting T1 or T2 to 0xffffffff ("infinity"). A client will never attempt to extend the lifetimes of any addresses in an IA with T1 set to 0xffffffff. A client will never attempt to use a Rebind message to locate a different server to extend the lifetimes of any addresses in an IA with T2 set to 0xffffffff.

22.5. Identity Association for Temporary Addresses Option

The Identity Association for the Temporary Addresses (IA_TA) option is used to carry an IA_TA, the parameters associated with the IA_TA and the addresses associated with the IA_TA. All of the addresses in this option are used by the client as temporary addresses, as defined in RFC 3041 [12]. The format of the IA_TA option is:



option-code OPTION_IA_TA (4).

option-len 4 + length of IA_TA-options field.

IAID The unique identifier for this IA_TA; the IAID must be unique among the identifiers for all of this client's IA_TAs. The number space for IA_TA IAIDs is separate from the number space for IA_NA IAIDs.

IA_TA-options Options associated with this IA_TA.

The IA_TA-Options field encapsulates those options that are specific to this IA_TA. For example, all of the IA Address Options carrying the addresses associated with this IA_TA are in the IA_TA-options field.

Each IA_TA carries one "set" of temporary addresses; that is, at most one address from each prefix assigned to the link to which the client is attached.

An IA_TA option may only appear in the options area of a DHCP message. A DHCP message may contain multiple IA_TA options.

The status of any operations involving this IA_TA is indicated in a Status Code option in the IA_TA-options field.

Note that an IA has no explicit "lifetime" or "lease length" of its own. When the valid lifetimes of all of the addresses in an IA_TA have expired, the IA can be considered as having expired.

An IA_TA option does not include values for T1 and T2. A client MAY request that the lifetimes on temporary addresses be extended by including the addresses in a IA_TA option sent in a Renew or Rebind message to a server. For example, a client would request an extension on the lifetime of a temporary address to allow an application to continue to use an established TCP connection.

The client obtains new temporary addresses by sending an IA_TA option with a new IAID to a server. Requesting new temporary addresses from the server is the equivalent of generating new temporary addresses as described in RFC 3041. The server will generate new temporary addresses and return them to the client. The client should request new temporary addresses before the lifetimes on the previously assigned addresses expire.

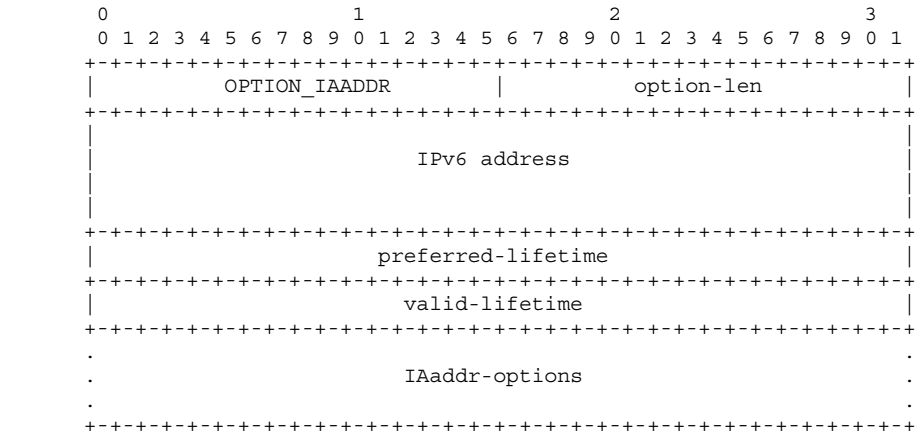
A server MUST return the same set of temporary address for the same IA_TA (as identified by the IAID) as long as those addresses are still valid. After the lifetimes of the addresses in an IA_TA have expired, the IAID may be reused to identify a new IA_TA with new temporary addresses.

This option MAY appear in a Confirm message if the lifetimes on the temporary addresses in the associated IA have not expired.

22.6. IA Address Option

The IA Address option is used to specify IPv6 addresses associated with an IA_NA or an IA_TA. The IA Address option must be encapsulated in the Options field of an IA_NA or IA_TA option. The Options field encapsulates those options that are specific to this address.

The format of the IA Address option is:



option-code OPTION_IAADDR (5).

option-len 24 + length of IAaddr-options field.

IPv6 address An IPv6 address.

preferred-lifetime The preferred lifetime for the IPv6 address in the option, expressed in units of seconds.

valid-lifetime The valid lifetime for the IPv6 address in the option, expressed in units of seconds.

IAaddr-options Options associated with this address.

In a message sent by a client to a server, values in the preferred and valid lifetime fields indicate the client's preference for those parameters. The client may send 0 if it has no preference for the preferred and valid lifetimes. In a message sent by a server to a client, the client MUST use the values in the preferred and valid lifetime fields for the preferred and valid lifetimes. The values in the preferred and valid lifetimes are the number of seconds remaining in each lifetime.

A client discards any addresses for which the preferred lifetime is greater than the valid lifetime. A server ignores the lifetimes set by the client if the preferred lifetime is greater than the valid lifetime and ignores the values for T1 and T2 set by the client if those values are greater than the preferred lifetime.

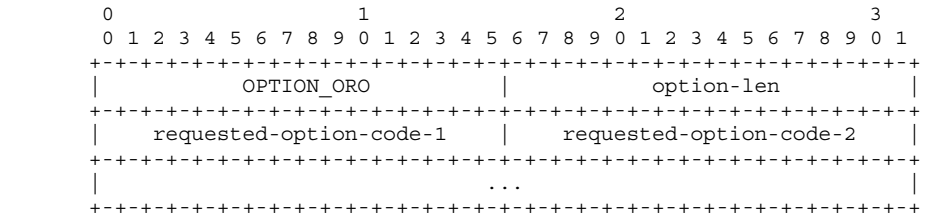
Care should be taken in setting the valid lifetime of an address to 0xffffffff ("infinity"), which amounts to a permanent assignment of an address to a client.

An IA Address option may appear only in an IA_NA option or an IA_TA option. More than one IA Address Option can appear in an IA_NA option or an IA_TA option.

The status of any operations involving this IA Address is indicated in a Status Code option in the IAaddr-options field.

22.7. Option Request Option

The Option Request option is used to identify a list of options in a message between a client and a server. The format of the Option Request option is:



option-code OPTION_ORO (6).

option-len 2 * number of requested options.

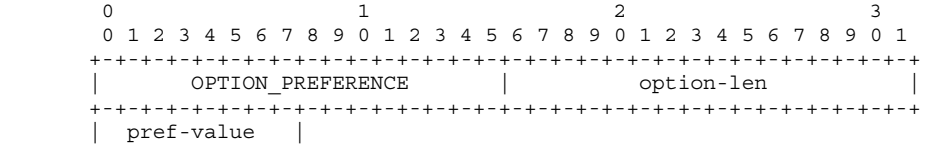
requested-option-code-n The option code for an option requested by the client.

A client MAY include an Option Request option in a Solicit, Request, Renew, Rebind, Confirm or Information-request message to inform the server about options the client wants the server to send to the client. A server MAY include an Option Request option in a Reconfigure option to indicate which options the client should request from the server.

22.8. Preference Option

The Preference option is sent by a server to a client to affect the selection of a server by the client.

The format of the Preference option is:



```
+-+-+-+-+-+-+-+-+
```

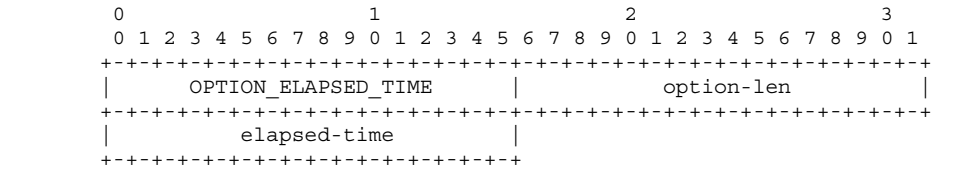
option-code OPTION_PREFERENCE (7).

option-len 1.

pref-value The preference value for the server in this message.

A server MAY include a Preference option in an Advertise message to control the selection of a server by the client. See section 17.1.3 for the use of the Preference option by the client and the interpretation of Preference option data value.

22.9. Elapsed Time Option



option-code OPTION_ELAPSED_TIME (8).

option-len 2.

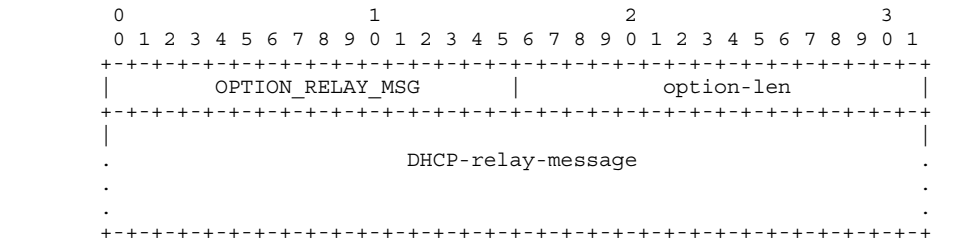
elapsed-time The amount of time since the client began its current DHCP transaction. This time is expressed in hundredths of a second (10^-2 seconds).

A client MUST include an Elapsed Time option in messages to indicate how long the client has been trying to complete a DHCP message exchange. The elapsed time is measured from the time at which the client sent the first message in the message exchange, and the elapsed-time field is set to 0 in the first message in the message exchange. Servers and Relay Agents use the data value in this option as input to policy controlling how a server responds to a client message. For example, the elapsed time option allows a secondary DHCP server to respond to a request when a primary server has not answered in a reasonable time. The elapsed time value is an unsigned, 16 bit integer. The client uses the value 0xffff to represent any elapsed time values greater than the largest time value that can be represented in the Elapsed Time option.

22.10. Relay Message Option

The Relay Message option carries a DHCP message in a Relay-forward or Relay-reply message.

The format of the Relay Message option is:



option-code OPTION_RELAY_MSG (9)

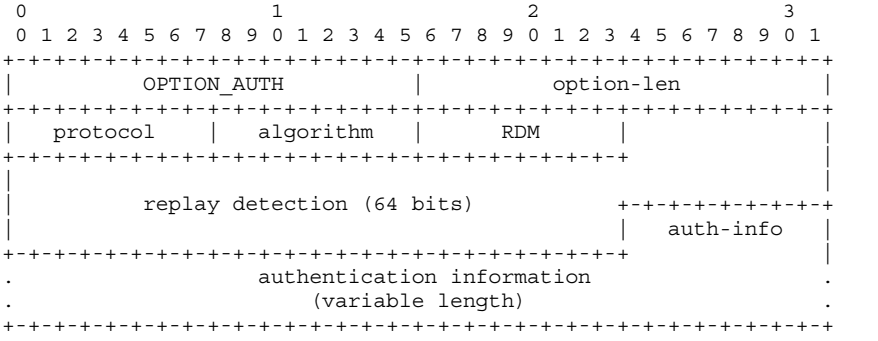
option-len Length of DHCP-relay-message

DHCP-relay-message In a Relay-forward message, the received message, relayed verbatim to the next relay agent or server; in a Relay-reply message, the message to be copied and relayed to the relay agent or client whose address is in the peer-address field of the Relay-reply message

22.11. Authentication Option

The Authentication option carries authentication information to authenticate the identity and contents of DHCP messages. The use of

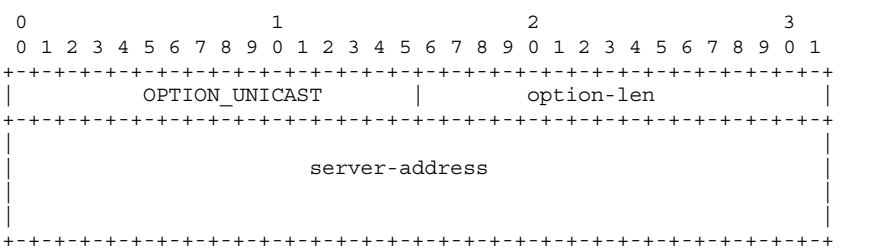
the Authentication option is described in section 21. The format of the Authentication option is:



option-code	OPTION_AUTH (11)
option-len	11 + length of authentication information field
protocol	The authentication protocol used in this authentication option
algorithm	The algorithm used in the authentication protocol
RDM	The replay detection method used in this authentication option
Replay detection	The replay detection information for the RDM
authentication information	The authentication information, as specified by the protocol and algorithm used in this authentication option

22.12. Server Unicast Option

The server sends this option to a client to indicate to the client that it is allowed to unicast messages to the server. The format of the Server Unicast option is:



option-code	OPTION_UNICAST (12).
option-len	16.
server-address	The IP address to which the client should send messages delivered using unicast.

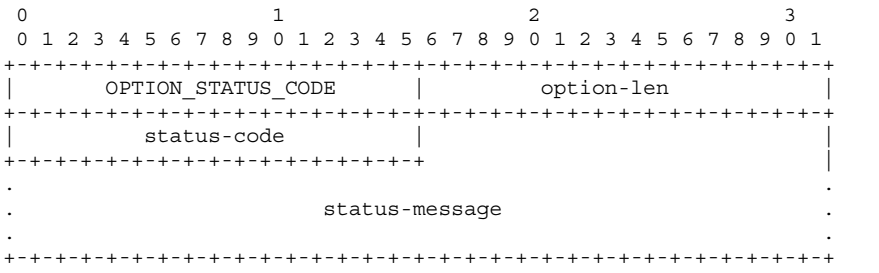
The server specifies the IPv6 address to which the client is to send unicast messages in the server-address field. When a client receives this option, where permissible and appropriate, the client sends messages directly to the server using the IPv6 address specified in the server-address field of the option.

When the server sends a Unicast option to the client, some messages from the client will not be relayed by Relay Agents, and will not include Relay Agent options from the Relay Agents. Therefore, a server should only send a Unicast option to a client when Relay Agents are not sending Relay Agent options. A DHCP server rejects any messages sent inappropriately using unicast to ensure that messages are relayed by Relay Agents when Relay Agent options are in use.

Details about when the client may send messages to the server using unicast are in section 18.

22.13. Status Code Option

This option returns a status indication related to the DHCP message or option in which it appears. The format of the Status Code option is:



option-code: OPTION_STATUS_CODE (13).

option-len: 2 + length of status-message.

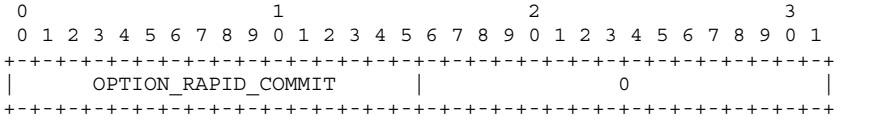
status-code: The numeric code for the status encoded in this option. The status codes are defined in section 24.4.

status-message: A UTF-8 encoded text string suitable for display to an end user, which MUST NOT be null-terminated.

A Status Code option may appear in the options field of a DHCP message and/or in the options field of another option. If the Status Code option does not appear in a message in which the option could appear, the status of the message is assumed to be Success.

22.14. Rapid Commit Option

The Rapid Commit option is used to signal the use of the two message exchange for address assignment. The format of the Rapid Commit option is:



option-code: OPTION_RAPID_COMMIT (14).

option-len: 0.

A client MAY include this option in a Solicit message if the client is prepared to perform the Solicit-Reply message exchange described in section 17.1.1.

A server MUST include this option in a Reply message sent in response to a Solicit message when completing the Solicit-Reply message exchange.

DISCUSSION:

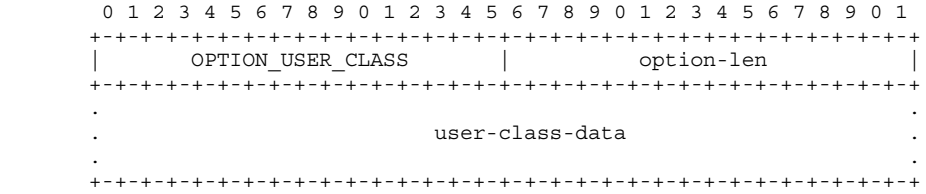
Each server that responds with a Reply to a Solicit that includes a Rapid Commit option will commit the assigned addresses in the Reply message to the client, and will not receive any confirmation that the client has received the Reply message. Therefore, if more than one server responds to a Solicit that includes a Rapid Commit option, some servers will commit addresses that are not actually used by the client.

The problem of unused addresses can be minimized, for example, by designing the DHCP service so that only one server responds to the Solicit or by using relatively short lifetimes for assigned addresses.

22.15. User Class Option

The User Class option is used by a client to identify the type or category of user or applications it represents.

The format of the User Class option is:



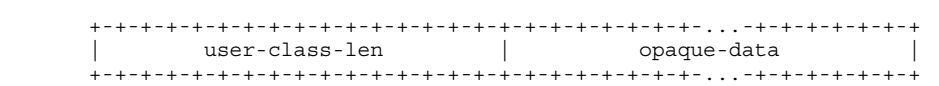
option-code OPTION_USER_CLASS (15).

option-len Length of user class data field.

user-class-data The user classes carried by the client.

The information contained in the data area of this option is contained in one or more opaque fields that represent the user class or classes of which the client is a member. A server selects configuration information for the client based on the classes identified in this option. For example, the User Class option can be used to configure all clients of people in the accounting department with a different printer than clients of people in the marketing department. The user class information carried in this option MUST be configurable on the client.

The data area of the user class option MUST contain one or more instances of user class data. Each instance of the user class data is formatted as follows:

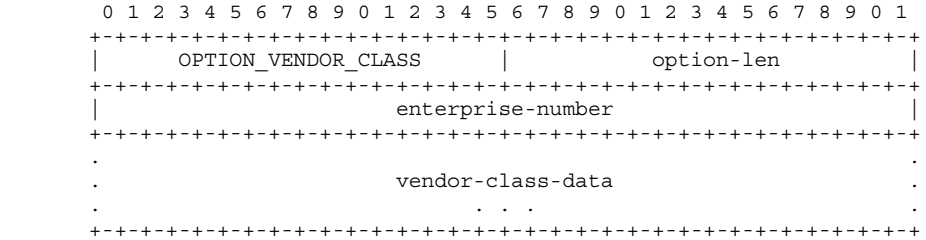


The user-class-len is two octets long and specifies the length of the opaque user class data in network byte order.

A server interprets the classes identified in this option according to its configuration to select the appropriate configuration information for the client. A server may use only those user classes that it is configured to interpret in selecting configuration information for a client and ignore any other user classes. In response to a message containing a User Class option, a server includes a User Class option containing those classes that were successfully interpreted by the server, so that the client can be informed of the classes interpreted by the server.

22.16. Vendor Class Option

This option is used by a client to identify the vendor that manufactured the hardware on which the client is running. The information contained in the data area of this option is contained in one or more opaque fields that identify details of the hardware configuration. The format of the Vendor Class option is:



option-code OPTION_VENDOR_CLASS (16).

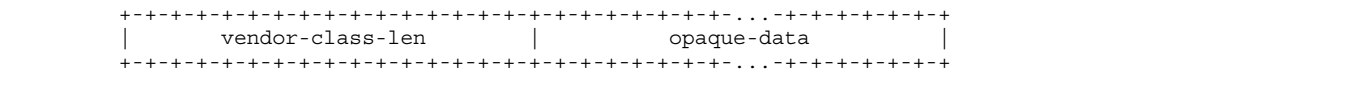
option-len 4 + length of vendor class data field.

enterprise-number The vendor's registered Enterprise Number as registered with IANA [6].

vendor-class-data The hardware configuration of the host on which the client is running.

The vendor-class-data is composed of a series of separate items, each of which describes some characteristic of the client's hardware configuration. Examples of vendor-class-data instances might include the version of the operating system the client is running or the amount of memory installed on the client.

Each instance of the vendor-class-data is formatted as follows:

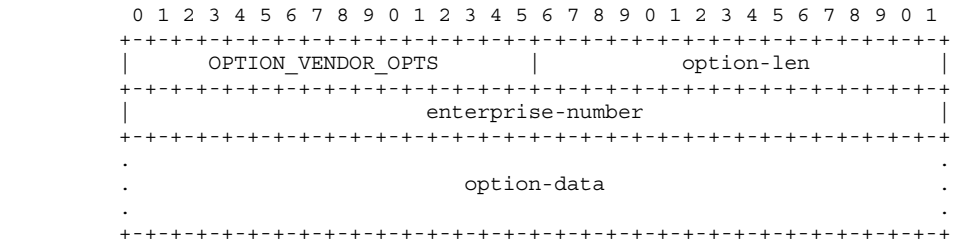


The vendor-class-len is two octets long and specifies the length of the opaque vendor class data in network byte order.

22.17. Vendor-specific Information Option

This option is used by clients and servers to exchange vendor-specific information.

The format of the Vendor-specific Information option is:



option-code OPTION_VENDOR_OPTS (17)

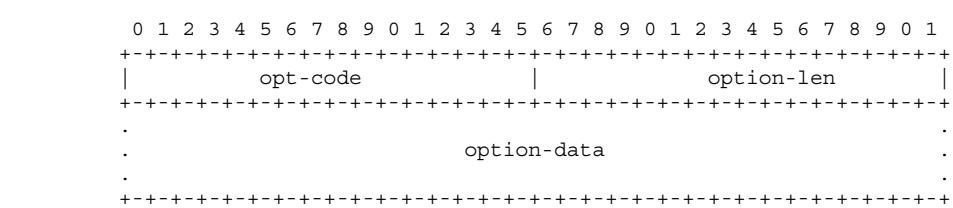
option-len 4 + length of option-data field

enterprise-number The vendor's registered Enterprise Number as registered with IANA [6].

option-data An opaque object of option-len octets, interpreted by vendor-specific code on the clients and servers

The definition of the information carried in this option is vendor specific. The vendor is indicated in the enterprise-number field. Use of vendor-specific information allows enhanced operation, utilizing additional features in a vendor's DHCP implementation. A DHCP client that does not receive requested vendor-specific information will still configure the host device's IPv6 stack to be functional.

The encapsulated vendor-specific options field MUST be encoded as a sequence of code/length/value fields of identical format to the DHCP options field. The option codes are defined by the vendor identified in the enterprise-number field and are not managed by IANA. Each of the encapsulated options is formatted as follows:



opt-code The code for the encapsulated option.

option-len An unsigned integer giving the length of the option-data field in this encapsulated option in octets.

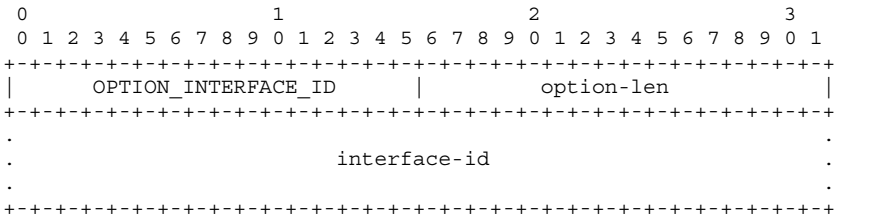
option-data The data area for the encapsulated option.

Multiple instances of the Vendor-specific Information option may appear in a DHCP message. Each instance of the option is interpreted according to the option codes defined by the vendor identified by the Enterprise Number in that option.

22.18. Interface-Id Option

The relay agent MAY send the Interface-id option to identify the interface on which the client message was received. If a relay agent receives a Relay-reply message with an Interface-id option, the relay agent relays the message to the client through the interface identified by the option.

The format of the Interface ID option is:



option-code OPTION_INTERFACE_ID (18).

option-len Length of interface-id field.

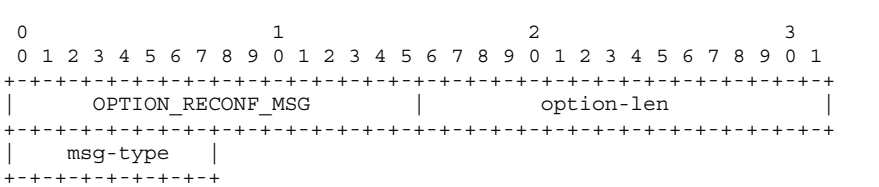
interface-id An opaque value of arbitrary length generated by the relay agent to identify one of the relay agent's interfaces.

The server MUST copy the Interface-Id option from the Relay-Forward message into the Relay-Reply message the server sends to the relay agent in response to the Relay-Forward message. This option MUST NOT appear in any message except a Relay-Forward or Relay-Reply message.

Servers MAY use the Interface-ID for parameter assignment policies. The Interface-ID SHOULD be considered an opaque value, with policies based on exact match only; that is, the Interface-ID SHOULD NOT be internally parsed by the server. The Interface-ID value for an interface SHOULD be stable and remain unchanged, for example, after the relay agent is restarted; if the Interface-ID changes, a server will not be able to use it reliably in parameter assignment policies.

22.19. Reconfigure Message Option

A server includes a Reconfigure Message option in a Reconfigure message to indicate to the client whether the client responds with a Renew message or an Information-request message. The format of this option is:



option-code OPTION_RECONF_MSG (19).

option-len 1.

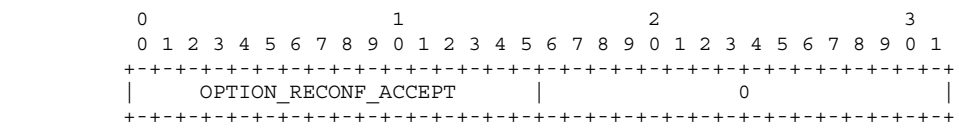
msg-type 5 for Renew message, 11 for Information-request message.

The Reconfigure Message option can only appear in a Reconfigure message.

22.20. Reconfigure Accept Option

A client uses the Reconfigure Accept option to announce to the server whether the client is willing to accept Reconfigure messages, and a server uses this option to tell the client whether or not to accept Reconfigure messages. The default behavior, in the absence of this

option, means unwillingness to accept Reconfigure messages, or instruction not to accept Reconfigure messages, for the client and server messages, respectively. The following figure gives the format of the Reconfigure Accept option:



option-code OPTION_RECONF_ACCEPT (20).

option-len 0.

G.5 RFC 3319

3.1 SIP Servers Domain Name List

The option length is followed by a sequence of labels, encoded according to Section 3.1 of RFC 1035 [5], quoted below:

"Domain names in messages are expressed in terms of a sequence of labels. Each label is represented as a one octet length field followed by that number of octets. Since every domain name ends with the null label of the root, a domain name is terminated by a length byte of zero. The high order two bits of every length octet must be zero, and the remaining six bits of the length field limit the label to 63 octets or less. To simplify implementations, the total length of a domain name (i.e., label octets and label length octets) is restricted to 255 octets or less."

RFC 1035 encoding was chosen to accommodate future internationalized domain name mechanisms.

The option MAY contain multiple domain names, but these SHOULD refer to different NAPTR records, rather than different A records. The client MUST try the records in the order listed, applying the mechanism described in Section 4.1 of RFC 3263 [3] for each. The client only resolves the subsequent domain names if attempts to contact the first one failed or yielded no common transport protocols between client and server or denote a domain administratively prohibited by client policy. Domain names MUST be listed in order of preference.

Use of multiple domain names is not meant to replace NAPTR or SRV records, but rather to allow a single DHCP server to indicate outbound proxy servers operated by multiple providers.

The DHCPv6 option has the format shown in Fig. 1.

option-code: OPTION_SIP_SERVER_D (21)

option-length: Length of the 'SIP Server Domain Name List' field in octets; variable.

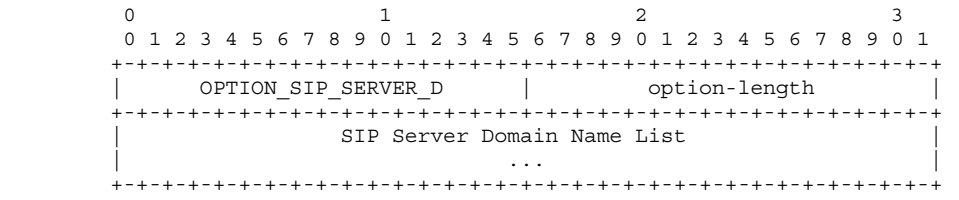
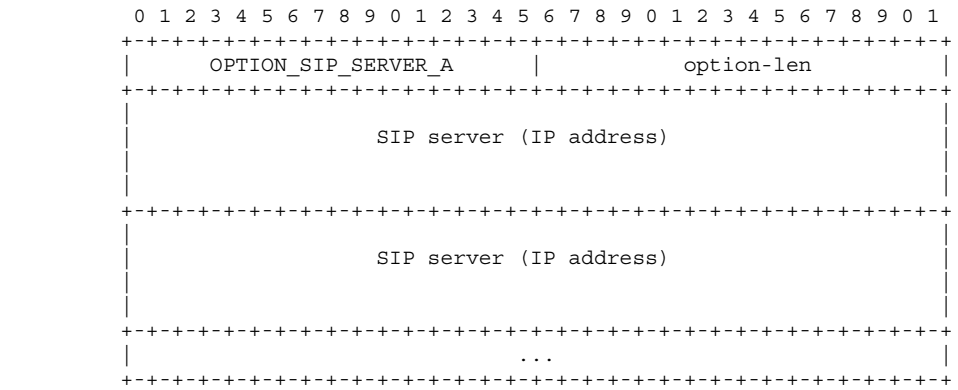


Figure 1: DHCPv6 option for SIP Server Domain Name List

SIP Server Domain Name List: The domain names of the SIP outbound proxy servers for the client to use. The domain names are encoded as specified in Section 8 ("Representation and use of domain names") of the DHCPv6 specification [1].

3.2 SIP Servers IPv6 Address List

This option specifies a list of IPv6 addresses indicating SIP outbound proxy servers available to the client. Servers MUST be listed in order of preference.



option-code: OPTION_SIP_SERVER_A (22)

option-length: Length of the 'options' field in octets; must be a multiple of 16.

SIP server: IPv6 address of a SIP server for the client to use. The servers are listed in the order of preference for use by the client.

G.6 RFC 3361

3.1 Domain Name List

If the 'enc' byte has a value of 0, the encoding byte is followed by a sequence of labels, encoded according to Section 3.1 of RFC 1035 [6], quoted below:

> Domain names in messages are expressed in terms of a sequence of labels. Each label is represented as a one octet length field followed by that number of octets. Since every domain name ends with the null label of the root, a domain name is terminated by a length byte of zero. The high order two bits of every length octet must be zero, and the remaining six bits of the length field limit the label to 63 octets or less. To simplify implementations, the total length of a domain name (i.e., label octets and label length octets) is restricted to 255 octets or less.

RFC 1035 encoding was chosen to accommodate future internationalized domain name mechanisms.

The minimum length for this encoding is 3.

The option MAY contain multiple domain names, but these SHOULD refer to different NAPTR records, rather than different A records. The client MUST try the records in the order listed, applying the mechanism described in Section 4.1 of RFC 3263 [3] for each. The client only resolves the subsequent domain names if attempts to contact the first one failed or yielded no common transport protocols between client and server or denote a domain administratively prohibited by client policy.

> Use of multiple domain names is not meant to replace NAPTR and SRV records, but rather to allow a single DHCP server to indicate outbound proxy servers operated by multiple providers.

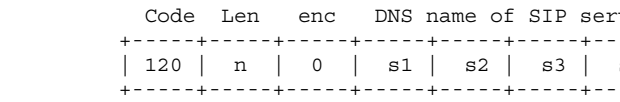
Clients MUST support compression according to the encoding in Section 4.1.4 of "Domain Names - Implementation And Specification" [6].

> Since the domain names are supposed to be different domains, compression will likely have little effect, however.

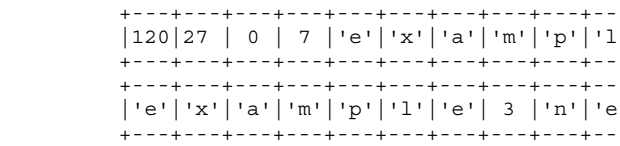
If the length of the domain list exceeds the maximum permissible within a single option (254 octets), then the domain list MUST be

represented in the DHCP message as specified in [7].

The DHCP option for this encoding has the following format:



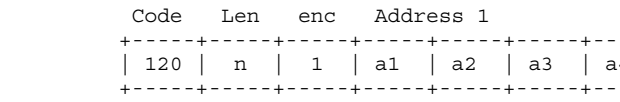
As an example, consider the case where the server wants to offer two outbound proxy servers, "example.com" and "example.net". These would be encoded as follows:



3.2 IPv4 Address List

If the 'enc' byte has a value of 1, the encoding byte is followed by a list of IPv4 addresses indicating SIP outbound proxy servers available to the client. Servers MUST be listed in order of preference.

Its minimum length is 5, and the length MUST be a multiple of 4 plus one. The DHCP option for this encoding has the following format:



Annex H (informative): Change history

Meet-ing	TSG doc	CR	Rev	Subject	Cat	Old vers	New vers	WG doc
RP-31	RP-060054	-	-	Update to version 1.0.0 and present to RAN#31 for information	-	-	1.0.0	R5-060513
RP-34	RP-060664	-	-	Present version 1.3.0 to RAN#34 for information	-	-	1.3.0	R5-063500
RP-35	RP-070010	-	-	Presented as version 2.0.0 for approval to go under revision control	-	-	2.0.0	R5-070456
-	-	-	-	Upgrated to version 5.0.0 by the 3GPP support	-	-	5.0.0	-
RP-36	RP-070352	0001	-	Addition of IMS-CC test case 8.6 to IMS_CC ATS V1.3.0	F	5.0.0	5.1.0	R5s070101
RP-36	RP-070353	0002	-	CR to 34.229-3: Add new verified and e-mail agreed TTCN test cases in the TC lists in 34.229-3 (prose), Annex A	F	5.0.0	5.1.0	-
RP-37	RP-070594	0003	-	Extension to TTCN ASP DeactivatePDPContextReq	F	5.1.0	5.2.0	R5-072509
RP-37	RP-070594	0004	-	IMS CC / PIXIT parameter px_CellId	F	5.1.0	5.2.0	R5-072546

History

Document history		
V5.0.0	March 2007	Publication
V5.1.0	June 2007	Publication
V5.2.0	October 2007	Publication