Supplemental Box

In case **the space in any of the preceding boxes is not sufficient.**

Continuation of: Box V

D3 discloses a system for managing address allocation of a mobile terminal in wireless LAN (WLAN) to inter-work with another WLAN or a public cellular network, wherein a data structure comprises: Message_Type, Message_Length, Domain_Name, MT_ID, Service_Request, Session_ID, Address_Request, Tunnel_Request, WLAN_ID and Security_Field.

Novelty
The subject matter of claim 60 is considered to be novel and complies with the requirement of **Article 33(2) of the PCT**. The cited references, when taken alone, fail to disclose a data structure for access by an apparatus for producing a routing message for use by a call routing controller in a communications system, the data structure comprising: dialing profile records comprising fields for associating with respective subscribers to the system: a subscriber user name; direct-in-dial records comprising fields for associating with respective subscriber usernames: a user domain; and a direct-in-dial number; prefix to node records comprising fields for associating with at least a portion of said respective subscriber usernames: a node address of a node in said system, whereby a subscriber name can be used to find a user domain, at least a portion of said subscriber name can be used to find a node with which the subscriber identified by the subscriber name is associated, and a user domain and subscriber name can be located in response to a direct-in-dial number.

The subject matter of claim 61 is considered to be novel and complies with the requirement of **Article 33(2) of the PCT**. The cited references, when taken alone, fail to disclose a data structure for access by an apparatus for producing a routing message for use by a call routing controller in a communications system, the data structure comprising: master list records comprising fields for associating a dialing code with respective master list identifiers; and supplier list records linked to master list records by said master list identifiers, said supplier list records comprising fields for associating with a communications service supplier: a supplier id; a master list id; a route identifier; and a billing rate code, whereby communications services suppliers are associated with dialing codes, such that dialing codes can be used to locate suppliers capable of providing a communications link associated with a given dialing code.

Inventive Step
The subject matter of claim 60 is considered to involve an inventive step and does comply with **Article 33(3) of the PCT**. The prior art, D2 and D3, alone or in combination, does not fairly suggest a data structure comprising: dialing profile records; direct-in-dial records; prefix to node records, whereby a subscriber name can be used to find a user domain, at least a portion of said subscriber name can be used to find a node with which the subscriber identified by the subscriber name is associated, and a user domain and subscriber name can be located in response to a direct-in-dial number.

The subject matter of claim 61 is considered to involve an inventive step and does comply with **Article 33(3) of the PCT**. The prior art, D2 and D3, alone or in combination, does not fairly suggest a data structure comprising: master list records; and supplier list records linked to master list records, said supplier list records comprising fields for associating with a communications service supplier, whereby communications services suppliers are associated with dialing codes, such that dialing codes can be used to locate suppliers capable of providing a communications link associated with a given dialing code.

Industrial Applicability
Claims 60 and 61 are considered to be industrially applicable and do comply with **Article 33(4) of the PCT**.

Group III (Claims 62-84)
The following documents are referred to in this communication:
D4 US 6058300
D5 US 2005/0177843 A1

D4 discloses, in part, a calculation of a maximum call duration in response to a customer account balance for a prepay telecommunications system.

D5 discloses, in part, calculation of a maximum call duration to a specific callee in response to a caller request to make a call in a prepay telecommunications system. If the maximum call duration is sufficient, the system permits the call to take place.

Novelty
Each of D4 and D5 fail to individually disclose all the elements of claims 62-84; therefore, claims 62-84 **are considered to be novel** in accordance with Article 33(2) PCT.

(Continued in next Supplemental Box)

Supplemental Box

In case the space in any of the preceding boxes is not sufficient.

Continuation of: Previous Supplemental Box

Inventive Step
Claim 62 claims a method of determining a time to permit a communications session to be conducted (ie, a maximum call duration). Either of D4 or D5 disclose determination of a maximum call duration and cause claim 62 to lack an inventive step. Both of D4 and D5 teach determination of a cost per unit time (D4: "rate per minute" (col. 5, line 58); D5: "call credits" (para. 65)), calculating a first time value as a sum of a free time attributed to a participant in the communication session and the quotient of a funds balance held by the participant to the cost per unit time value (D4: col. 5, lines 61 - 65; D5: para. 67), and producing a second time value in response to the first time value and a billing pattern (D4: roaming or not roaming; D5: "call history"), the second time value being the time to permit a communications session to be conducted. Additional differences between claim 62 and either D4 or D5 such as "free time", "cost per unit time" and "billing pattern" also lack inventive step. Thus claim 62 is considered to **lack an inventive step** in accordance with Article 33(3) PCT.
As claims 73 and 74 are apparatus for carrying out methods steps similar or identical to those of claim 62, these claims **lack an inventive step** in accordance with Article 33(3) PCT for the same reasons as listed above.

Claim 63 and 75 lack an inventive step in view of either of D4 or D5 in that D4 and D5 disclose retrieving a record associated with said participant (D4: "customer's account" (col. 5, lines 63-64); D5: "certificate information" (para. 67)) and obtaining from said record said funds balance (D4: col. 5, line 63; D5: para. 67). To also obtain a participant's free time also lack an inventive step. Thus, claims 63 and 75 **lack an inventive step** in accordance with Article 33(3) PCT.

Claims 64-72 and 76-84 are found to be inventive since no combination of prior art documents were found which disclose the subject matter as set forth in claims 64-72 and 76-84 in accordance with Article 33(3) PCT.

Industrial Applicability
Determination of maximum time for a communication session finds application within Internet telephony; thus, claims 62-84 are considered to **have industrial applicability** in accordance with Article 33(4) PCT.

Group IV (Claims 85-107)
The following document is referred to in this communication:
D6 US 6188752

D6 is considered to form the closest prior art. D6 discloses provision of prepaid telecommunications services by a telecommunications network. A database record includes subscriber information fields such as account numbers, prepaid account information, and a current prepayment monetary amounts. Once a call or communication session has been established, the network monitors parameters related to any fee to be charged for the service such as start time, elapsed time, origination and destination locations, and rate information (ie, billing pattern) preferably in real time. D6 further discloses determining the cost of the call and debiting the account balance associated with the subscriber.

Novelty
D6 fails to individually disclose all the elements of claims 85-107; therefore, claims 85-107 **are considered to be novel** in accordance with Article 33(2) PCT.

Inventive Step
Independent claims 85, 96, and 97 each claim attributing charges for communications services including determining a chargeable time in response to a communication session time and a pre-defined billing pattern, determining a user cost value, and changing account balances associated with the user, reseller, and operator of the communications services. D6 teaches attributing charges for communications services, determining a chargeable time in response to a communication session time and a pre-defined billing pattern, determining a user cost value in response to said first chargeable time associated with a user of said communications services, and changing an account balance associated with said user in response to a user cost per unit time. However, D6 fails to suggest a free time value, nor does D6 teach changing the account balances of either a reseller or an operator of said communications services.

Claims 86-95 and 98-107 depend on independent claims 85 and 97, respectively.

Therefore, claims 85-107 **are considered to have an inventive step** in accordance with Article 33(3) PCT.

Industrial Applicability
Billing or attributing charges for communications services finds use in telecommunications, and, thus, claims 85-107 are considered to

PATENT COOPERATION TREATY

From the INTERNATIONAL SEARCHING AUTHORITY

To:
SMART & BIGGAR
Box 11560 Vancouver Centre
2200 - 650 W. Georgia Street
VANCOUVER, British Columbia
Canada, V6B 4N8

RECEIVED
INFO TO BE FILED
2008 JUN 25 A 9:49
DUE: SEP 20 08
DOCKETED
DOCKET

PCT

NOTIFICATION OF TRANSMITTAL OF THE INTERNATIONAL SEARCH REPORT AND THE WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY, OR THE DECLARATION

(PCT Rule 44.1)

Date of mailing (day/month/year): 20 June 2008 (20-06-2008)

Applicant's or agent's file reference: 83636-18

FOR FURTHER ACTION See paragraphs 1 and 4 below

AMENDS UNDER ART 19
DUE: AUG 20/08
DOCKETED

International application No.: PCT/CA2008/000545

International filing date (day/month/year): 20 March 2008 (20-03-2008)

Applicant: DIGIFONICA (INTERNATIONAL) LIMITED ET AL

1. [X] The applicant is hereby notified that the international search report and the written opinion of the International Searching Authority have been established and are transmitted herewith.

   Filing of amendments and statement under Article 19 :
   The applicant is entitled, if he so wishes, to amend the claims of the international application (see Rule 46) :

   When? The time limit for filing such amendments is normally two months from the date of transmittal of the international search report.

   Where? Directly to the International Bureau of WIPO, 34 chemin des Colombettes 1211 Geneva 20, Switzerland, Facsimile No.: +41 22 338 82 70

   For more detailed instructions, see the notes on the accompanying sheet.

2. [ ] The applicant is hereby notified that no international search report will be established and that the declaration under Article 17(2)(a) to that effect and the written opinion of the International Searching Authority are transmitted herewith.

3. [ ] **With regard to the protest** against payment of (an) additional fee(s) under Rule 40.2, the applicant is notified that :

   [ ] the protest together with the decision thereon has been transmitted to the International Bureau together with the applicant's request to forward the texts of both the protest and the decision thereon to the designated Offices.

   [ ] no decision has been made yet on the protest; the applicant will be notified as soon as a decision is made.

4. **Reminders**

   Shortly after the expiration of **18 months** from the priority date, the international application will be published by the International Bureau. If the applicant wishes to avoid or postpone publication, a notice of withdrawal of the international application, or of the priority claim, must reach the International Bureau as provided in Rules 90*bis*.1 and 90*bis*.3, respectively, before the completion of the technical preparations for the international publication.

   The applicant may submit comments on an informal basis on the written opinion of the International Searching Authority to the International Bureau. The International Bureau will send a copy of such comments to all designated Offices unless an international preliminary examination report has been or is to be established. These comments would also be made available to the public but not before the expiration of 30 months from the priority date.

   Within **19 months** from the priority date, but only in respect of some designated Offices, a demand for international preliminary examination must be filed if the applicant wishes to postpone the entry into the national phase **until 30 months** from the priority date (in some Offices even later); otherwise, the applicant must, **within 20 months** from the priority date, perform the prescribed acts for entry into the national phase before those designated Offices.

   In respect of other designated Offices, the time limit of **30 months** (or later) will apply even if no demand is filed within 19 months.

   See the Annex to Form PCT/IB/301 and, for details about the applicable time limits, Office by Office, see the *PCT Applicant's Guide*, Volume II, National Chapters and the WIPO Internet site.

Name and mailing address of the ISA/CA
Canadian Intellectual Property Office
Place du Portage I, C114 - 1st Floor, Box PCT
50 Victoria Street
Gatineau, Quebec K1A 0C9
Facsimile No.: 001-819-953-2476

Authorized officer
Donna Daly 819- 953-8972

Form PCT/ISA/220 (October 2005) *(See notes on accompanying sheet)*

The letter must indicate the differences between the claims as filed and the claims as amended. It must, in particular, indicate, in connection with each claim appearing in the international application (it being understood that identical indications concerning several claims may be grouped), whether

(i) the claim is unchanged;
(ii) the claim is cancelled;
(iii) the claim is new;
(iv) the claim replaces one or more claims as filed;
(v) the claim is the result of the division of a claim as filed.

The following examples illustrate the manner in which amendments must be explained in the accompanying letter :

1. [Where originally there were 48 claims and after amendment of some claims there are 51]:
"Claims 1 to 29, 31, 32, 34, 35, 37 to 48 replaced by amended claims bearing the same numbers; claims 30, 33 and 36 unchanged; new claims 49 to 51 added."

2. [Where originally there were 15 claims and after amendment of all claims there are 11]:
"Claims 1 to 15 replaced by amended claims 1 to 11."

3. [Where originally there were 14 claims and the amendments consist in cancelling some claims and in adding new claims]:
"Claims 1 to 6 and 14 unchanged; claims 7 to 13 cancelled; new claims 15, 16 and 17 added." or
"Claims 7 to 13 cancelled; new claims 15, 16 and 17 added; all other claims unchanged."

4. [Where various kinds of amendments are made]:
"Claims 1-10 unchanged; claims 11 to 13, 18 and 19 cancelled; claims 14, 15 and 16 replaced by amended claim 14; claim 17 subdivided into amended claims 15, 16 and 17; new claims 20 and 21 added."

"Statement under Article 19(1)" (Rule 46.4)

The amendments may be accompanied by a statement explaining the amendments and indicating any impact that such amendments might have on the description and the drawings (which cannot be amended under Article 19(1)).

The statement will be published with the international application and the amended claims.

It must be in the language in which the international application is to be published.

It must be brief, not exceeding 500 words if in English or if translated into English.

It should not be confused with and does not replace the letter indicating the differences between the claims as filed and as amended. It must be filed on a separate sheet and must be identified as such by a heading, preferably by using the words "Statement under Article 19(1)."

It may not contain any disparaging comments on the international search report or the relevance of citations contained in that report. Reference to citations, relevant to a given claim, contained in the international search report may be made only in connection with an amendment of that claim.

Consequence if a demand for international preliminary examination has already been filed

If, at the time of filing any amendments and any accompanying statement, under Article 19, a demand for international preliminary examination has already been submitted, the applicant must preferably, at the time of filing the amendments (and any statement) with the International Bureau, also file with the International Preliminary Examining Authority a copy of such amendments (and of any statement) and, where required, a translation of such amendments for the procedure before that Authority (see Rules 55.3(a) and 62.2, first sentence). For further information, see the Notes to the demand form (PCT/IPEA/401).

If a demand for international preliminary examination is made, the written opinion of the International Searching Authority will, except in certain cases where the International Preliminary Examining Authority did not act as International Searching Authority and where it has notified the International Bureau under Rule 66.1*bis*(b), be considered to be a written opinion of the International Preliminary Examining Authority. If a demand is made, the applicant may submit to the International Preliminary Examining Authority a reply to the written opinion together, where appropriate, with amendments before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later (Rule 43*bis*.1(c)).

Consequence with regard to translation of the international application for entry into the national phase

The applicant's attention is drawn to the fact that, upon entry into the national phase, a translation of the claims as amended under Article 19 may have to furnished to the designated/elected Offices, instead of, or in addition to, the translation of the claims as filed.

For further details on the requirements of each designated/elected Office, see the *PCT Applicant's Guide*, Volume II.

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

RECEIVED
2008 JUN 25 A 8:49
2200-650 WEST GEORGIA ST.

Applicant's or agent's file reference 83636-18	**FOR FURTHER ACTION**	See Form PCT/ISA/220 as well as, where applicable, item 5 below
International application No. **PCT/CA2008/000545**	International filing date *(day/month/year)* 20 March 2008 (20-03-2008)	(Earliest) Priority date *(day/month/year)* 26 March 2007 (26-03-2007)

Applicant
DIGIFONICA (INTERNATIONAL) LIMITED ET AL

This international search report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This international search report consists of a total of 3 sheets.

[X] It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**

a. With regard to the **language,** the international search was carried out on the basis of:

[X] the international application in the language in which it was filed

[ ] a translation of the international application into , which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b))

b. [ ] This international search report has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43.6*bis*(a)).

c. [ ] With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, see Box No. I

2. [ ] **Certain claims were found unsearchable** (see Box No. II)

3. [ ] **Unity of invention is lacking** (see Box No. III)

4. With regard to the **title,**

[X] the text is approved as submitted by the applicant

[ ] the text has been established by this Authority to read as follows :

5. With regard to the **abstract,**

[X] the text is approved as submitted by the applicant

[ ] the text has been established, according to Rule 38.2, by this Authority as it appears in Box No. IV. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority

6. With regard to the **drawings,**

a. the figure of the **drawings** to be published with the abstract is Figure No. 1

[X] as suggested by the applicant

[ ] as selected by this Authority, because the applicant failed to suggest a figure

[ ] as selected by this Authority, because this figure better characterizes the invention

b. [ ] none of the figures is to be published with the abstract

DCC
JWK

A. CLASSIFICATION OF SUBJECT MATTER
IPC: ***H04L 12/66*** (2006.01) , ***H04M 11/06*** (2006.01) , ***H04M 3/42*** (2006.01) , ***H04Q 3/00*** (2006.01) , ***H04Q 3/64*** (2006.01)
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC: H04L (2006.01), H04M (2006.01), H04Q (2006.01); US classes: 379/45, 370 in combination with keywords

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Canadian Patent Database, USPTO West, Delphion. Keywords: pbx, inward dialing, dial-in, voip, sip, psap, erc, ali, flag and variations thereof.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X, P	US 2008/0063153 A1 (Krivorot et al.) 13 March 2008 (13-03-2008) * Para. [0004] - [0024], [0042] - [0082]; Fig. 1 *	1, 15, 29, 43
A, P		2-14, 16-28, 30-42
A	US 2005/0083911 A1 (Grabelsky et al.) 21 April 2005 (21-04-2005) * Para. [0041] - [0055], [0060] - [0138]; Fig. 1, 2 *	1-43
A	US 2005/0169248 A1 (Truesdale et al.) 4 August 2005 (04-08-2005) * Para. [0013] - [0050] *	1-43
A, P	US 2008/0037715 A1 (Prozeniuk et al.) 14 February 2008 (14-02-2008) * Entire document *	1-43
A, P	WO 2007/044454 A2 (Croy et al.) 19 April 2007 (19-04-2007) * Entire document *	1-43

[ ] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents :
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
6 June 2008 (06-06-2008)	20 June 2008 (20-06-2008)

Name and mailing address of the ISA/CA	Authorized officer
Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	Arthur Smith 819-953-1360

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
US2008063153	13-03-2008	CA2598200 A1	21-02-2008
US2005083911	21-04-2005	EP1526697 A2	27-04-2005
US2005169248	04-08-2005	NONE	
US2008037715	14-02-2008	NONE	
WO2007044454	19-04-2007	US2007091831 A1	26-04-2007
		US2007091906 A1	26-04-2007
		US2007092070 A1	26-04-2007
		WO2007044455 A2	19-04-2007
		WO2007055971 A2	18-05-2007

PATENT COOPERATION TREATY

From the
INTERNATIONAL SEARCHING AUTHORITY

To:
SMART & BIGGAR
Box 11560 Vancouver Centre
2200 - 650 W. Georgia Street
VANCOUVER, British Columbia
Canada, V6B 4N8

RECEIVED
2008 JUN 25 A 9:48
2200-650 WEST GEORGIA ST.
VANCOUVER, B.C

PCT

WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY
(PCT Rule 43*bis*.1)

Date of mailing *(day/month/year)*	20 June 2008 (20-06-2008)
Applicant's or agent's file reference 83636-18	**FOR FURTHER ACTION** See paragraph 2 below

International application No.	International filing date *(day/month/year)*	Priority date *(day/month/year)*
PCT/CA2008/000545	20 March 2008 (20-03-2008)	26 March 2007 (26-03-2007)

International Patent Classification (IPC) or both national classification and IPC
IPC: ***H04L 12/66*** (2006.01), ***H04M 11/06*** (2006.01), ***H04M 3/42*** (2006.01), ***H04Q 3/00*** (2006.01), ***H04Q 3/64*** (2006.01)

Applicant
DIGIFONICA (INTERNATIONAL) LIMITED ET AL

WRITTEN OPINION
DUE: JAN 26 '09
DOCK: JAN 23 '09
DOCKET

1. This opinion contains indications relating to the following items :

[X]	Box No. I	Basis of the opinion
[ ]	Box No. II	Priority
[ ]	Box No. III	Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
[ ]	Box No. IV	Lack of unity of invention
[X]	Box No. V	Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
[ ]	Box No. VI	Certain documents cited
[X]	Box No. VII	Certain defects in the international application
[X]	Box No. VIII	Certain observations on the international application

2. **FURTHER ACTION**

If a demand for international preliminary examination is made, this opinion will be considered to be a written opinion of the International Preliminary Examining Authority ("IPEA") except that this does not apply where the applicant chooses an Authority other than this one to be the IPEA and the chosen IPEA has notified the International Bureau under Rule 66.1*bis*(b) that written opinions of this International Searching Authority will not be so considered.

If this opinion is, as provided above, considered to be a written opinion of the IPEA, the applicant is invited to submit to the IPEA a written reply together, where appropriate, with amendments, before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later.

For further options, see Form PCT/ISA/220.

3. For further details, see notes to Form PCT/ISA/220.

Name and mailing address of the ISA/CA	Date of completion of this opinion	Authorized officer
Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	13 June 2008 (13-06-2008)	Arthur Smith 819-953-1360

Box No. V Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. Statement

Novelty (N)	Claims	1-43	YES
	Claims	None	NO
Inventive step (IS)	Claims	1-43	YES
	Claims	None	NO
Industrial applicability (IA)	Claims	1-43	YES
	Claims	None	NO

2. Citations and explanations :

The following document is referred to in this communication:
D1 US2005/0083911 A1

D1 is considered to form the closest prior art. D1 describes providing E911 emergency services to an IP telephony-based PBX or similar system by establishing a 911 Location Server Database comprising an Emergency Response Location (ERL) database and a Phone Location database. The location of a caller stored in the ERL database is used to route an emergency call to an end-office switch corresponding to the nearest Public Safety Answering Point (PSAP). The Phone Location database stores one record for each registered phone in the system.

Novelty
D1 fails to disclose all the elements of independent claims 1, 15, 29, and 43. In particular, D1 fails to teach a direct inward dialing (DID) identifier or producing a DID identifier for a caller identifier that has no pre-associated DID identifier. Claims 2-14, 16-28 and 30-42 depend on the above independent claims, respectively.

Thus, claims 1-43 **are considered to be novel** in accordance with Article 33(2) PCT.

Inventive Step
D1 fails to teach or suggest producing a routing message including an emergency response center identifier and temporary DID identifier for establishing a route between a caller and an emergency response center as claimed in independent claims 1, 15, 29, and 43. Claims 2-14, 16-28, and 30-42 depend on the above independent claims, respectively.

Thus, claims 1-43 **are considered to have inventive step** in accordance with Article 33(3) PCT.

Industrial Applicability
Including location information for emergency VoIP callers located behind a VoIP PBX finds application in IP telephony networks. Thus, claims 1-43 **are industrially applicable** in accordance with Article 33(4) PCT.

Box No. VII Certain defects in the international application

The following defects in the form or contents of the international application have been noted :

Description-Related Deficiencies

On page 1, line 19, of the description, the term "presended" (sic) is misspelled. Applicant may have intended "presented to".

Box No. VIII Certain observations on the international application

The following observations on the clarity of the claims, description, and drawings or on the question whether the claims are fully supported by the description, are made :

Claims-Related Deficiencies

Claims 1 and 22 do not comply with PCT Article 6. In claim 1, the term "the method" (line 3) lacks an antecedent **and** in claim 22, the term "said pool" (line 15) lacks an antecedent.

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

RECEIVED
2010 MAR 2 A 9:19
VANCOUVER B.C.

Applicant's or agent's file reference 83636-56	**FOR FURTHER ACTION**	see Form PCT/ISA/220 as well as, where applicable, item 5 below
International application No. **PCT/CA2009/001317**	International filing date *(day/month/year)* 17 September 2009 (17-09-2009)	(Earliest)Priority date *(day/month/year)*

Applicant
DIGIFONICA (INTERNATIONAL) LIMITED ET AL

This international search report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This international search report consists of a total of 3 sheets.

[X] It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**

a. With regard to the **language**, the international search was carried out on the basis of:

[X] the international application in the language in which it was filed

[ ] a translation of the international application into , which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b))

b. [ ] This international search report has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43.6*bis*(a)).

c. [ ] With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, see Box No. I

2. [ ] **Certain claims were found unsearchable** (see Box No. II)

3. [ ] **Unity of invention is lacking** (see Box No. III)

4. With regard to the **title**,

[X] the text is approved as submitted by the applicant

[ ] the text has been established by this Authority to read as follows :

5. With regard to the **abstract**,

[X] the text is approved as submitted by the applicant

[ ] the text has been established, according to Rule 38.2, by this Authority as it appears in Box No. IV. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority

6. With regard to the **drawings**,

a. the figure of the **drawings** to be published with the abstract is Figure No. 1

[X] as suggested by the applicant

[ ] as selected by this Authority, because the applicant failed to suggest a figure

[ ] as selected by this Authority, because this figure better characterizes the invention

b. [ ] none of the figures is to be published with the abstract

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2009/001317

A. CLASSIFICATION OF SUBJECT MATTER
IPC: ***H04L 12/66*** (2006.01) , ***H04L 29/06*** (2006.01) , ***H04W 36/02*** (2009.01) , ***H04W 36/18*** (2009.01)
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC (2006.01): H04L, H04W

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Canadian patent database, IEEE Xplore, TotalPatent: media relay, Internet Protocol (IP), Real Time Transport Protocol (RT)P, endpoint, change, transmissions, data, port identifier, caller, callee, destination, port match, source, destination, address identifier, Caller Synchronization Source (SSRC) identifier, base station, GSM, UDP, record, maintenance, port identifier, matching and all such related terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2009/0028146 A1 KLEYMAN et al. 29 January 2009 (29-01-2009) Entire document	1 to 16
A	US 7454510 B2 KLEYMAN et al. 18 November 2008 (18-11-2008) Entire document	1 to 16
A	US 2007/0253418 A1 SHIRI et al. 01 November 2007 (01-11-2007) Entire document	1 to 16
A	US 2007/0036143 A1 ALT et al. 15 February 2007 (15-02-2007) Entire document	1 to 16

[ ] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents :
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
17 July 2010 (17-06-2010)	18 June 2010 (18-06-2010)
Name and mailing address of the ISA/CA Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	Authorized officer Salvatore Ginese (819) 934-4888

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
US2009028146A1	29-01-2009	CN1574798A	02-02-2005
		EP1484883A2	08-12-2004
		JP2004355628A	16-12-2004
		KR20040103441A	08-12-2004
		US2004244010A1	02-12-2004
		US7454510B2	18-11-2008
US7454510B2	18-11-2008	CN1574798A	02-02-2005
		EP1484883A2	08-12-2004
		JP2004355628A	16-12-2004
		KR20040103441A	08-12-2004
		US2004244010A1	02-12-2004
		US2009028146A1	29-01-2009
US2007253418A1	01-11-2007	US2007253418A1	01-11-2007
		WO2007125530A2	08-11-2007
		WO2007125530A3	27-12-2007
US2007036143A1	15-02-2007	AU2005272561A1	23-02-2006
		AU2005272561B2	01-10-2009
		BRPI0514326A	10-06-2008
		CA2577123A1	23-02-2006
		CN101084686A	05-12-2007
		EP1784959A2	16-05-2007
		EP1784959A4	26-08-2009
		EP1784999A1	16-05-2007
		EP1784999A4	09-07-2008
		EP1787441A2	23-05-2007
		EP1787441A4	23-09-2009
		JP2008510393T	03-04-2008
		JP2008510394T	03-04-2008
		JP2008515246T	08-05-2008
		KR20070104509A	26-10-2007
		US2006072542A1	06-04-2006
		US7602748B2	13-10-2009
		US2006209794A1	21-09-2006
		US7706401B2	27-04-2010
		US2009279506A1	12-11-2009
		WO2006020975A2	23-02-2006
		WO2006020975A9	06-04-2006
		WO2006020975A3	26-05-2006
		WO2006020977A1	23-02-2006
		WO2006020997A2	23-02-2006
		WO2006020997A3	07-08-2008

PATENT COOPERATION TREATY

From the
INTERNATIONAL SEARCHING AUTHORITY

To:
SMART & BIGGAR
Box 11560 Vancouver Centre
2200 - 650 W. Georgia Street
VANCOUVER, British Columbia
Canada, V6B 4N8

RECEIVED
2010 JUN 23 A
650 WEST GEORGIA ST.
VANCOUVER

PCT

WRITTEN OPINION OF THE
INTERNATIONAL SEARCHING AUTHORITY

(PCT Rule 43*bis*.1)

Date of mailing *(day/month/year)*	18 June 2010 (18-06-2010)
Applicant's or agent's file reference 83636-56	**FOR FURTHER ACTION** See paragraph 2 below

International application No.	International filing date *(day/month/year)*	Priority date *(day/month/year)*
PCT/CA2009/001317	17 September 2009 (17-09-2009)	

International Patent Classification (IPC) or both national classification and IPC
IPC: ***H04L 12/66*** (2006.01), ***H04L 29/06*** (2006.01), ***H04W 36/02*** (2009.01), ***H04W 36/18*** (2009.01)

Applicant
DIGIFONICA (INTERNATIONAL) LIMITED ET AL

1. This opinion contains indications relating to the following items :

[X]	Box No. I	Basis of the opinion
[ ]	Box No. II	Priority
[ ]	Box No. III	Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
[ ]	Box No. IV	Lack of unity of invention
[X]	Box No. V	Reasoned statement under Rule 43*bis*.1(a)(I) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
[ ]	Box No. VI	Certain documents cited
[ ]	Box No. VII	Certain defects in the international application
[ ]	Box No. VIII	Certain observations on the international application

2. **FURTHER ACTION**
If a demand for international preliminary examination is made, this opinion will be considered to be a written opinion of the International Preliminary Examining Authority ("IPEA") except that this does not apply where the applicant chooses an Authority other than this one to be the IPEA and the chosen IPEA has notified the International Bureau under Rule 66.1*bis*(b) that written opinions of this International Searching Authority will not be so considered.

If this opinion is, as provided above, considered to be a written opinion of the IPEA, the applicant is invited to submit to the IPEA a written reply together, where appropriate, with amendments, before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later.

For further options, see Form PCT/ISA/220.

3. For further details, see notes to Form PCT/ISA/220.

Name and mailing address of the ISA/CA	Date of completion of this opinion	Authorized officer
Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	17 June 2010 (17-06-2010)	Salvatore Ginese (819) 934-4888

Box No. I **Basis of this opinion**

1. With regard to the **language**, this opinion has been established on the basis of:

   [X] the international application in the language in which it was filed

   [ ] a translation of the international application into , which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b)).

2. [ ] This opinion has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43*bis*.1(a))

3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, this opinion has been established on the basis of a sequence listing filed or furnished:

   a. (means)

      [ ] on paper

      [ ] in electronic form

   b. (time)

      [ ] the international application as filed.

      [ ] together with the international application in electronic form

      [ ] subsequently to this Authority for the purposes of search

4. [ ] In addition, in the case that more than one version or copy of a sequence listing has been filed or furnished, the required statements that the information in the subsequent or additional copies is identical to that in the application as filed or does not go beyond the application as filed, as appropriate, were furnished.

5. Additional comments :

Box No. V **Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement**

1. Statement

Novelty (N)	Claims	1 to 16	YES
	Claims	None	NO
Inventive step (IS)	Claims	1 to 16	YES
	Claims	None	NO
Industrial applicability (IA)	Claims	1 to 16	YES
	Claims	None	NO

2. Citations and explanations :

Reference is made to the following documents:

D1: US 2009/0028146 A1	**KLEYMAN et al.**	**29 January 2009**	**(29-01-2009)**
D2: US 7454510 B2	**KLEYMAN et al.**	**18 November 2008**	**(18-11-2008)**
D3: US 2007/0253418 A1	**SHIRI et al.**	**01 November 2007**	**(01-11-2007)**
D4: US 2007/0036143 A1	**ALT et al.**	**15 February 2007**	**(15-02-2007)**

Novelty

None of D1 to D4 individually teaches explicitly a method for facilitating uninterrrupted transmission of internet protocol (IP) transmissions containing real time transport protocol (RTP) data during endpoint changes, the method comprising: maintaining records, each record associating session information, caller information and callee information for IP communication sessions; said session information including caller and callee RTP port identifiers identifying caller and callee RTP ports respectively of a media relay; said caller information including a caller IP address identifier and a caller port identifier to which IP transmission received at said callee RTP port are transmitted form the media relay, and a caller synchronization source (SSRC) identifier; and said callee information including a callee IP address identifier and a callee port identifier to which IP transmissions received at said caller RTP port are transmitted from the media relay, and a callee SSRC identifier; and when an IP transmission is received at said caller RTP port or said callee RTP port; locating one of said records having said caller RTP port identifier or said callee RTP port identifier matching a destination port identifier in said IP transmission; when said one of said records is located and when said destination port identifier in said IP transmission matches the (caller or callee) RTP port identifier of said one of said records; setting a source IP address identifier and source port identifier from said IP transmission as the (caller or callee) IP address identifier and (caller/callee) IP address identifier and(caller/callee) port identifier respectively of said one of said records when said (caller/callee) IP address identifier and (caller/callee) port identifier do not match said source IP address identifier and source port identifier respectively; and a received SSRC identifier in said IP transmission matches said (caller/callee) SSRC identifier.

The subject matter of claims 1 to 16 is therefore considered to be novel, satisfying the requirements of **Article 33(2).**

(Continued in supplemental box 1 of 1)

Supplemental Box

In case **the space in any of the preceding boxes is not sufficient.**

Continuation of: Box No. V (This is supplemental box 1 of 1)

Inventive step

None of D1 to D4 teaches or suggests in combination the aforementioned features.

The subject matter of claims 1 to 16 is therefore considered to contain an inventive step, satisfying the requirements of **Article 33(3).**

Industrial applicability

The subject matter of claims 1 to 16 is considered to be industrially applicable, thus fulfilling the requirements of **Article 33(4)**.

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

RECEIVED

Applicant's or agent's file reference 83636-16	**FOR FURTHER ACTION**	See Form PCT/ISA/220 as well as, where applicable, item 5 below
International application No. **PCT/CA2007/001956**	International filing date *(day/month/year)* 01 November 2007 (01-11-2007)	(Earliest) Priority date *(day/month/year)* 02 November 2006 (02-11-2006)

Applicant
DIGIFONICA (INTERNATIONAL) LIMITED ET AL

This international search report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This international search report consists of a total of 5 sheets.

[X] It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**

   a. With regard to the **language**, the international search was carried out on the basis of:

      [X] the international application in the language in which it was filed

      [ ] a translation of the international application into , which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b))

   b. [ ] This international search report has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43.6*bis*(a)).

   c. [ ] With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, see Box No. I

2. [ ] **Certain claims were found unsearchable** (see Box No. II)

3. [X] **Unity of invention is lacking** (see Box No. III)

4. With regard to the **title**,

   [X] the text is approved as submitted by the applicant

   [ ] the text has been established by this Authority to read as follows :

5. With regard to the **abstract**,

   [X] the text is approved as submitted by the applicant

   [ ] the text has been established, according to Rule 38.2, by this Authority as it appears in Box No. IV. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority

6. With regard to the **drawings**,

   a. the figure of the **drawings** to be published with the abstract is Figure No. 1

      [X] as suggested by the applicant

      [ ] as selected by this Authority, because the applicant failed to suggest a figure

      [ ] as selected by this Authority, because this figure better characterizes the invention

   b. [ ] none of the figures is to be published with the abstract

DOC
JLE
JWK

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of the first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons :

1. [ ] Claim Nos. :
because they relate to subject matter not required to be searched by this Authority, namely :

2. [ ] Claim Nos. :
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically :

3. [ ] Claim Nos. :
because they are dependant claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows :

Group I	Claims 1-59
Group II	Claims 60, 61
Group III	Claims 62-84
Group IV	Claims 85-107

1. [X] As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. [ ] As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.

3. [ ] As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claim Nos. :

4. [ ] No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claim Nos. :

Remark on Protest

[ ] The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

[ ] The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

[X] No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2007/001956

A. CLASSIFICATION OF SUBJECT MATTER
IPC: ***H04L 12/66*** (2006.01) , ***H04L 12/14*** (2006.01) , ***H04M 11/06*** (2006.01) , ***H04M 15/00*** (2006.01) , ***H04Q 3/64*** (2006.01)
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC: H04L (2006.01), H04M (2006.01), H04Q (2006.01); US classes: 370, 379 in combination with keywords

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Canadian Patent Database, USPTO West, Delphion. Keywords: public network, private network, routing message, instant messaging, ip phone, voip, routing controller, sip, gateway, ttl, metric, skype, data structure, routing message, billing, communication session, prepaid

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CA2249668 C (Bruno et al.) 7 April 1999 (07-04-1999) * Page 9, line 4 to page 14, line 18; Figs 1, 2 *	1-59
A	US7120682 B1 (Salama) 10 October 2006 (10-10-2006) * Col. 1, line 47 to col. 4, line 67 *	1-59
A	US2006/0160565 A1 (Singh et al.) 20 July 2006 (20-07-2006) * Paragraphs 14, 15, 18; Figs 1, 2 *	1-59
A	US2006/0177035 A1 (Cope et al.) 10 August 2006 (10-08-2006) * Paragraphs 5, 6, 12 *	1-59
A, P	US7212522 B1 (Shankar et al.) 1 May 2007 (01-05-2007) * Col. 4, line 47 to col. 5, line 11; Fig. 1 *	1-59

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
6 February 2008 (06-02-2008)	20 February 2008 (20-02-2008)
Name and mailing address of the ISA/CA Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	Authorized officer Arthur Smith 819-953-1360

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US7068772 (Widger et al.) 27 June 2006 (27-06-2006) * Col. 12, line 49 to col. 14, line 44; col. 15, line 26 to col. 16, line 30 Figs. 3, 5 *	60, 61
A	US2006/0209768 A1 (Yan et al.) 21 September 2006 (21-09-2006) * Paras. 71-99, 111-118, 128-141, 179-188; Figs. 3, 4, 7-9 *	60, 61
X	US6058300 (Hanson) 2 May 2000 (02-05-2000) * Col. 2, lines 9-13; col. 5, line 55 to col. 6, line 23; col. 6, line 55 to	62, 63, 73-75
A	col. 7, line 18 *	64-72, 76-84
X	US2005/0177843 A1 (Williams) 11 August 2005 (11-08-2005) * Paragraphs 64 - 69 *	62, 63,73-75
A		64-72, 76-84
A	US6188752 B1 (Lesley) 13 February 2001 (13-02-2001) * Col. 4, line 24 to col. 9, line 6; Figs 1, 3 *	85-107
A	US6507644 B1 (Henderson et al.) 14 January 2003 (14-01-2003) * Col. 1, line 51 to col. 6, line 28 *	85-107
A	US5359642 (Castro) 25 October 1994 (25-10-1994) * Abstract; Col. 5, lines 7-12, col. 6, line 5 to col. 8, line 38; col. 10, line 10 to col. 12, line 68 *	85-107

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
CA2249668	07-04-1999	EP0915594 A2	12-05-1999
		US6614765 B1	02-09-2003
US7120682	10-10-2006	NONE	
US2006160565	20-07-2006	NONE	
US2006177035	10-08-2006	CA2595429 A1	03-08-2006
		WO2006081115 A1	03-08-2006
US7212522	01-05-2007	US6570869 B1	27-05-2003
		US6658022 B1	02-12-2003
		US6768733 B1	27-07-2004
US7068772	27-06-2006	NONE	
US2006209768	21-09-2006	CA2512959 A1	10-09-2004
		CN1762129 A	19-04-2006
		EP1585270 A1	12-10-2005
		JP2004266310 A	24-09-2004
		KR20050092405 A	21-09-2005
		WO2004077754 A1	10-09-2004
US6058300	02-05-2000	AU6142498 A	25-08-1998
		CA2250845 A1	06-08-1998
		US6029062 A	22-02-2000
		US6208851 B1	27-03-2001
		US6625438 B2	23-09-2003
		US7162220 B2	09-01-2007
		WO9834393 A2	06-08-1998
US2005177843	11-08-2005	AU2002351582 A1	15-07-2003
		CA2469959 A1	10-07-2003
		CA2471113 A1	10-07-2003
		US2003120553 A1	26-06-2003
		US2006190353 A1	24-08-2006
		WO03056803 A2	10-07-2003
US6188752	13-02-2001	AU730021B B2	22-02-2001
		AU5073398 A	03-06-1998
		BR9713025 A	25-01-2000
		CA2271311 A1	22-05-1998
		CN1244987 A	16-02-2000
		DE69732526D D1	24-03-2005
		DE69732526T T2	28-07-2005
		EP0944994 A1	29-09-1999
		ES2237791T T3	01-08-2005
		JP2001504299T T	27-03-2001
		KR20000053241 A	25-08-2000
		NO992280 A	12-07-1999
		US6333976 B2	25-12-2001
		WO9821874 A1	22-05-1998
US6507644	14-01-2003	NONE	
US5359642	25-10-1994	NONE	

PATENT COOPERATION TREATY

From the
INTERNATIONAL SEARCHING AUTHORITY

To:
SMART & BIGGAR
Box 11560 Vancouver Centre
2200 - 650 W. Georgia Street
VANCOUVER, British Columbia
Canada, V6B 4N8

RECEIVED PCT/VED
2008 FEB 26 A 10:27
650 WEST GEORGIA ST. 650 WEST GEORGIA ST.
VANCOUVER, B.C VANCOUVER, B.C

WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY
(PCT Rule 43*bis*.1)

Date of mailing *(day/month/year)*	20 February 2008 (20-02-2008)
Applicant's or agent's file reference 83636-16	**FOR FURTHER ACTION** See paragraph 2 below

International application No.	International filing date *(day/month/year)*	Priority date *(day/month/year)*
PCT/CA2007/001956	01 November 2007 (01-11-2007)	02 November 2006 (02-11-2006)

International Patent Classification (IPC) or both national classification and IPC
IPC: ***H04L 12/66*** (2006.01), ***H04L 12/14*** (2006.01), ***H04M 11/06*** (2006.01), ***H04M 15/00*** (2006.01), ***H04Q 3/64*** (2006.01)

Applicant
DIGIFONICA (INTERNATIONAL) LIMITED ET AL

NKS
Written opinion
due Sept 2/08
DOCKET
dock: Sept 1/08

1. This opinion contains indications relating to the following items :

[X]	Box No. I	Basis of the opinion
[ ]	Box No. II	Priority
[ ]	Box No. III	Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
[X]	Box No. IV	Lack of unity of invention
[X]	Box No. V	Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
[ ]	Box No. VI	Certain documents cited
[ ]	Box No. VII	Certain defects in the international application
[X]	Box No. VIII	Certain observations on the international application

2. **FURTHER ACTION**
If a demand for international preliminary examination is made, this opinion will be considered to be a written opinion of the International Preliminary Examining Authority ("IPEA") except that this does not apply where the applicant chooses an Authority other than this one to be the IPEA and the chosen IPEA has notified the International Bureau under Rule 66.1*bis*(b) that written opinions of this International Searching Authority will not be so considered.

If this opinion is, as provided above, considered to be a written opinion of the IPEA, the applicant is invited to submit to the IPEA a written reply together, where appropriate, with amendments, before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later.

For further options, see Form PCT/ISA/220.

3. For further details, see notes to Form PCT/ISA/220.

Name and mailing address of the ISA/CA	Date of completion of this opinion	Authorized officer
Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	7 February 2008 (07-02-2008)	Arthur Smith 819-953-1360

DOC JCC JWK

Box No. I Basis of this opinion

1. With regard to the **language**, this opinion has been established on the basis of:

   [X] the international application in the language in which it was filed

   [ ] a translation of the international application into , which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b)).

2. [ ] This opinion has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43*bis*.1(a))

3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application and necessary to the claimed invention, this opinion has been established on the basis of :

   a. type of material

      [ ] a sequence listing

      [ ] table(s) related to the sequence listing

   b. format of material

      [ ] on paper

      [ ] in electronic form

   c. time of filing/furnishing

      [ ] contained in the international application as filed.

      [ ] filed together with the international application in electronic form

      [ ] furnished subsequently to this Authority for the purposes of search.

4. [ ] In addition, in the case that more than one version or copy of a sequence listing and/or table(s) relating thereto has been filed or furnished, the required statements that the information in the subsequent or additional copies is identical to that in the application as filed or does not go beyond the application as filed, as appropriate, were furnished.

5. Additional comments :

Box No. IV **Lack of unity of invention**

1. [X] In response to the invitation (Form PCT/ISA/206) to pay additional fees the applicant has, within the applicable time limit :

   [X] paid additional fees

   [ ] paid additional fees under protest and, where applicable, the protest fee

   [ ] paid additional fees under protest but the applicable protest fee was not paid

   [ ] not paid additional fees

2. [ ] This Authority found that the requirement of unity of invention is not complied with and chose not to invite the applicant to pay additional fees.

3. This Authority considers that the requirement of unity of invention in accordance with Rules 13.1, 13.2 and 13.3 is

   [ ] complied with

   [X] not complied with for the following reasons :

   This International Searching Authority considers that there are four inventions claimed in the international application covered by the claims indicated below:

   I Claims 1-59
   II Claims 60, 61
   III Claims 62-84
   IV Claims 85-107

   The claims of Group I have in common a call routing controller for facilitating communications between callers and callees in a communications system comprising a plurality of nodes in which, in response to initiation of a call, uses call classification criteria to classify the call as a public network call or a private network call, and produces accordingly a routing message.

   The claims of Group II have in common a data structure for access by an apparatus for producing a routing message for use by a call routing controller in a communications system.

   The claims of Group III have in common determining a time to permit a communication session to be conducted, the determination based on calculating a cost per unit time, a participant's billing pattern, and the quotient of a funds balance held by the participant.

   The claims of Group IV have in common attributing charges for communications services by determining chargeable times and changing account balances of both user and communications services reseller.

   Groups I and II have in common the call routing controller; however, call routing controllers are well known in the art so the claims of Groups I and II lack unity *a posteriori*.

   Because the remainder of the claims of Groups I, II, III, and IV have no elements in common and would require separate searches by the examiner, these groups lack unity *a priori*.

4. Consequently, this opinion has been established in respect of the following parts of the international application :

   [X] all parts

   [ ] the parts relating to claim Nos.

Box No. V **Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement**

1. Statement

Novelty (N)	Claims	1-107	YES
	Claims	None	NO
Inventive step (IS)	Claims	1-61, 64-72, 76-107	YES
	Claims	62, 63, 73-75	NO
Industrial applicability (IA)	Claims	1-107	YES
	Claims	None	NO

2. Citations and explanations :

Group I (Claims 1-59)
The following document is referred to in this communication:
D1 CA 2249668

D1 is considered to form the closest prior art. D1 discloses routing information in an integrated global communications network in which a central routing processor collects routing capabilities of network nodes for which it has responsibility. The routing processor evaluates the routing requirements of a routing query signal transmitted by a source router, determines which routers and communication paths within the network are capable and available to route the information, evaluates the statistical availability of such routers, and selects an optimal routing path to a destination router.

Novelty
D1 fails to individually disclose all the elements of claims 1-59; therefore, claims 1-59 **are considered to be novel** in accordance with Article 33(2) PCT.

Inventive Step
Independent claims 1, 30, and 31 each claim operating a call routing controller to facilitate communication between callers and callees in a system or network comprising a plurality of nodes in which call classification criteria associated with a caller identifier is used to classify the call as a public network call or a private network call, and producing a routing message in accordance with the classification. D1 teaches facilitation of communication between callers and callees within a private network, including producing a routing message for a private network call. However, D1 fails to teach classification of a call as a public network call, and fails to teach producing an appropriate routing message for a public network call.

Claims 2-29 and 32-59 depend on independent claims 1 and 31, respectively.

Therefore, claims 1-59 **are considered to have an inventive step** in accordance with Article 33(3) PCT.

Industrial Applicability
Routing IP phone calls through a communication network including both private and public aspects finds use in telecommunications, and, thus, claims 1-59 **are industrially applicable** in accordance with Article 33(4) PCT.

Group II (Claims 60, 61)
The following documents are referred to in this communication:
D2 US7,068,772
D3 US2006/0209768

D2 and D3 are considered to form the closest prior art. D2 discloses a call processing system and method for providing one-number telecommunication services, wherein a data structure of a subscriber record for access by an apparatus for producing a routing message, the data structure comprising the subscriber's various profiles and the associated 1-800 number or address that the subscriber may be contacted.

(Continued in Supplemental Box)

Box No. VIII Certain observations on the international application

The following observations on the clarity of the claims, description, and drawings or on the question whether the claims are fully supported by the description, are made :

Claim-Related Objections

Claim 60 is unclear and does not comply with Article 6 of the PCT. The following terms lack a proper antecedent basis:
"the subscriber" (claim 60, page 83, line 16)
"subscriber name" (claim 60, page 83, line 17)

Claim 60 is unclear and does not comply with Article 6 of the PCT. The double inclusion of any element renders the claims indefinite. The following expressions have already been defined previously in the claims and should therefore be referred to using a definite article:
"a user domain" (claim 60, page 83, lines 14, 17)
"a direct-in-dial number" (claim 60, page 83, line 18)

Claim 60 is indefinite and does not comply with Article 6 of the PCT. The terms "a subscriber user name" (claim 60, page 82, line 32) and "subscriber name" (claim 60, page 83, lines 14, 15, 16-17, 17) cause ambiguity. It is not clear whether they are the same or different.

Claim 61 is indefinite and does not comply with Article 6 of the PCT. The term "master list records" (page 83, line 27) causes ambiguity. It should read "said master list records".

Claim 61 is indefinite and does not comply with Article 6 of the PCT. The term "aid" (page 83, line 28) causes ambiguity. It should read "said aid".

Claim 61 is indefinite and does not comply with Article 6 of the PCT. The term "dialing codes" (page 84, line 8) causes ambiguity. It should read "said dialing codes".

Supplemental Box

In case **the space in any of the preceding boxes is not sufficient.**

Continuation of: Box V

D3 discloses a system for managing address allocation of a mobile terminal in wireless LAN (WLAN) to inter-work with another WLAN or a public cellular network, wherein a data structure comprises: Message_Type, Message_Length, Domain_Name, MT_ID, Service_Request, Session_ID, Address_Request, Tunnel_Request, WLAN_ID and Security_Field.

Novelty
The subject matter of claim 60 is considered to be novel and complies with the requirement of **Article 33(2) of the PCT**. The cited references, when taken alone, fail to disclose a data structure for access by an apparatus for producing a routing message for use by a call routing controller in a communications system, the data structure comprising: dialing profile records comprising fields for associating with respective subscribers to the system: a subscriber user name; direct-in-dial records comprising fields for associating with respective subscriber usernames: a user domain; and a direct-in-dial number; prefix to node records comprising fields for associating with at least a portion of said respective subscriber usernames: a node address of a node in said system, whereby a subscriber name can be used to find a user domain, at least a portion of said subscriber name can be used to find a node with which the subscriber identified by the subscriber name is associated, and a user domain and subscriber name can be located in response to a direct-in-dial number.

The subject matter of claim 61 is considered to be novel and complies with the requirement of **Article 33(2) of the PCT**. The cited references, when taken alone, fail to disclose a data structure for access by an apparatus for producing a routing message for use by a call routing controller in a communications system, the data structure comprising: master list records comprising fields for associating a dialing code with respective master list identifiers; and supplier list records linked to master list records by said master list identifiers, said supplier list records comprising fields for associating with a communications service supplier: a supplier id; a master list id; a route identifier; and a billing rate code, whereby communications services suppliers are associated with dialing codes, such that dialing codes can be used to locate suppliers capable of providing a communications link associated with a given dialing code.

Inventive Step
The subject matter of claim 60 is considered to involve an inventive step and does comply with **Article 33(3) of the PCT**. The prior art, D2 and D3, alone or in combination, does not fairly suggest a data structure comprising: dialing profile records; direct-in-dial records; prefix to node records, whereby a subscriber name can be used to find a user domain, at least a portion of said subscriber name can be used to find a node with which the subscriber identified by the subscriber name is associated, and a user domain and subscriber name can be located in response to a direct-in-dial number.

The subject matter of claim 61 is considered to involve an inventive step and does comply with **Article 33(3) of the PCT**. The prior art, D2 and D3, alone or in combination, does not fairly suggest a data structure comprising: master list records; and supplier list records linked to master list records, said supplier list records comprising fields for associating with a communications service supplier, whereby communications services suppliers are associated with dialing codes, such that dialing codes can be used to locate suppliers capable of providing a communications link associated with a given dialing code.

Industrial Applicability
Claims 60 and 61 are considered to be industrially applicable and do comply with **Article 33(4) of the PCT.**

Group III (Claims 62-84)
The following documents are referred to in this communication:
D4 US 6058300
D5 US 2005/0177843 A1

D4 discloses, in part, a calculation of a maximum call duration in response to a customer account balance for a prepay telecommunications system.

D5 discloses, in part, calculation of a maximum call duration to a specific callee in response to a caller request to make a call in a prepay telecommunications system. If the maximum call duration is sufficient, the system permits the call to take place.

Novelty
Each of D4 and D5 fail to individually disclose all the elements of claims 62-84; therefore, claims 62-84 **are considered to be novel** in accordance with Article 33(2) PCT.

(Continued in next Supplemental Box)

Supplemental Box

In case **the space in any of the preceding boxes is not sufficient.**

Continuation of: Previous Supplemental Box

Inventive Step
Claim 62 claims a method of determining a time to permit a communications session to be conducted (ie, a maximum call duration). Either of D4 or D5 disclose determination of a maximum call duration and cause claim 62 to lack an inventive step. Both of D4 and D5 teach determination of a cost per unit time (D4: "rate per minute" (col. 5, line 58); D5: "call credits" (para. 65)), calculating a first time value as a sum of a free time attributed to a participant in the communication session and the quotient of a funds balance held by the participant to the cost per unit time value (D4: col. 5, lines 61 - 65; D5: para. 67), and producing a second time value in response to the first time value and a billing pattern (D4: roaming or not roaming; D5: "call history"), the second time value being the time to permit a communications session to be conducted. Additional differences between claim 62 and either D4 or D5 such as "free time", "cost per unit time" and "billing pattern" also lack inventive step. Thus claim 62 is considered to **lack an inventive step** in accordance with Article 33(3) PCT.
As claims 73 and 74 are apparatus for carrying out methods steps similar or identical to those of claim 62, these claims **lack an inventive step** in accordance with Article 33(3) PCT for the same reasons as listed above.

Claim 63 and 75 lack an inventive step in view of either of D4 or D5 in that D4 and D5 disclose retrieving a record associated with said participant (D4: "customer's account" (col. 5, lines 63-64); D5: "certificate information" (para. 67)) and obtaining from said record said funds balance (D4: col. 5, line 63; D5: para. 67). To also obtain a participant's free time also lack an inventive step. Thus, claims 63 and 75 **lack an inventive step** in accordance with Article 33(3) PCT.

Claims 64-72 and 76-84 are found to be inventive since no combination of prior art documents were found which disclose the subject matter as set forth in claims 64-72 and 76-84 in accordance with Article 33(3) PCT.

Industrial Applicability
Determination of maximum time for a communication session finds application within Internet telephony; thus, claims 62-84 are considered to **have industrial applicability** in accordance with Article 33(4) PCT.

Group IV (Claims 85-107)
The following document is referred to in this communication:
D6 US 6188752

D6 is considered to form the closest prior art. D6 discloses provision of prepaid telecommunications services by a telecommunications network. A database record includes subscriber information fields such as account numbers, prepaid account information, and a current prepayment monetary amounts. Once a call or communication session has been established, the network monitors parameters related to any fee to be charged for the service such as start time, elapsed time, origination and destination locations, and rate information (ie, billing pattern) preferably in real time. D6 further discloses determining the cost of the call and debiting the account balance associated with the subscriber.

Novelty
D6 fails to individually disclose all the elements of claims 85-107; therefore, claims 85-107 **are considered to be novel** in accordance with Article 33(2) PCT.

Inventive Step
Independent claims 85, 96, and 97 each claim attributing charges for communications services including determining a chargeable time in response to a communication session time and a pre-defined billing pattern, determining a user cost value, and changing account balances associated with the user, reseller, and operator of the communications services. D6 teaches attributing charges for communications services, determining a chargeable time in response to a communication session time and a pre-defined billing pattern, determining a user cost value in response to said first chargeable time associated with a user of said communications services, and changing an account balance associated with said user in response to a user cost per unit time. However, D6 fails to suggest a free time value, nor does D6 teach changing the account balances of either a reseller or an operator of said communications services.

Claims 86-95 and 98-107 depend on independent claims 85 and 97, respectively.

Therefore, claims 85-107 **are considered to have an inventive step** in accordance with Article 33(3) PCT.

Industrial Applicability
Billing or attributing charges for communications services finds use in telecommunications, and, thus, claims 85-107 are considered to **have industrial applicability** in accordance with Article 33(4) PCT.

PATENT COOPERATION TREATY

From the INTERNATIONAL SEARCHING AUTHORITY

To:
SMART & BIGGAR
Box 11560 Vancouver Centre
2200 - 650 W. Georgia Street
VANCOUVER, British Columbia
Canada, V6B 4N8

INFO TO BE FILED
DUE: ON
DOCK: JUN 14/08
DOCKET
JMW

PCT

NOTIFICATION OF TRANSMITTAL OF THE INTERNATIONAL SEARCH REPORT AND THE WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY, OR THE DECLARATION
(PCT Rule 44.1)

RECEIVED 2008 MAR 20 A 8:54 SMART & BIGGAR VANCOUVER, B.C.

AMENDS UNDER ART 19
DUE: MAY 14/08
DOCUMENTS
JMW

Date of mailing *(day/month/year)*	14 March 2008 (14-03-2008)
Applicant's or agent's file reference 83636-13	**FOR FURTHER ACTION** See paragraphs 1 and 4 below
International application No. PCT/CA2007/002150	International filing date *(day/month/year)* 29 November 2007 (29-11-2007)

Applicant
DIGIFONICA (INTERNATIONAL) LIMITED ET AL

1. [X] The applicant is hereby notified that the international search report and the written opinion of the International Searching Authority have been established and are transmitted herewith.

   Filing of amendments and statement under Article 19 :
   The applicant is entitled, if he so wishes, to amend the claims of the international application (see Rule 46) :

   When? The time limit for filing such amendments is normally two months from the date of transmittal of the international search report.

   Where? Directly to the International Bureau of WIPO, 34 chemin des Colombettes 1211 Geneva 20, Switzerland, Facsimile No.: +41 22 338 82 70

   For more detailed instructions, see the notes on the accompanying sheet.

2. [ ] The applicant is hereby notified that no international search report will be established and that the declaration under Article 17(2)(a) to that effect and the written opinion of the International Searching Authority are transmitted herewith.

3. [ ] **With regard to the protest** against payment of (an) additional fee(s) under Rule 40.2, the applicant is notified that :

   [ ] the protest together with the decision thereon has been transmitted to the International Bureau together with the applicant's request to forward the texts of both the protest and the decision thereon to the designated Offices.

   [ ] no decision has been made yet on the protest; the applicant will be notified as soon as a decision is made.

4. **Reminders**

   Shortly after the expiration of **18 months** from the priority date, the international application will be published by the International Bureau. If the applicant wishes to avoid or postpone publication, a notice of withdrawal of the international application, or of the priority claim, must reach the International Bureau as provided in Rules 90*bis*.1 and 90*bis*.3, respectively, before the completion of the technical preparations for the international publication.

   The applicant may submit comments on an informal basis on the written opinion of the International Searching Authority to the International Bureau. The International Bureau will send a copy of such comments to all designated Offices unless an international preliminary examination report has been or is to be established. These comments would also be made available to the public but not before the expiration of 30 months from the priority date.

   Within **19 months** from the priority date, but only in respect of some designated Offices, a demand for international preliminary examination must be filed if the applicant wishes to postpone the entry into the national phase **until 30 months** from the priority date (in some Offices even later); otherwise, the applicant must, **within 20 months** from the priority date, perform the prescribed acts for entry into the national phase before those designated Offices.

   In respect of other designated Offices, the time limit of **30 months** (or later) will apply even if no demand is filed within 19 months.

   See the Annex to Form PCT/IB/301 and, for details about the applicable time limits, Office by Office, see the *PCT Applicant's Guide*, Volume II, National Chapters and the WIPO Internet site.

Name and mailing address of the ISA/CA Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	Authorized officer Donna Daly 819- 953-8972

Form PCT/ISA/220 (October 2005) *(See notes on accompanying sheet)*

DOC
SLG
JMW

NOTES TO FROM PCT/ISA/220

These Notes are intended to give instructions concerning the filing of amendments under Article 19. The Notes are based on the requirements of the Patent Cooperation Treaty, the Regulations and the Administrative Instructions under that Treaty. In case of discrepancy between these Notes and those requirements, the latter are applicable. For more detailed information, see also the *PCT Applicant's Guide*, a publication of WIPO.

In these Notes, "Article," "Rule" and "Section" refer to the provisions of the PCT, the PCT Regulations and the PCT Administrative Instructions, respectively.

INSTRUCTIONS CONCERNING AMENDMENTS UNDER ARTICLE 19

The applicant has, after having received the international search report and the written opinion of the International Searc Authority, one opportunity to amend the claims of the international application. It should however be emphasized that, since all parts of the international application (claims, description and drawings) may be amended during the international preliminary examination procedure, there is usually no need to file amendments of the claims under Article 19 except where, e.g. the applicant wants the latter to be published for the purposes of provisional protection or has another reason for amending the claims before international publication. Furthermore, it should be emphasized that provisional protection is available in some States only (see *PCT Applicant's Guide*, Volume I/A, Annexes B1 and B2).

The attention of the applicant is drawn to the fact that amendments to the claims under Article 19 are not allowed where the International Searching Authority has declared, under Article 17(2), that no international search report would be established (see *PCT Applicant's Guide*, Volume I/A, paragraph 296).

What parts of the international application may be amended?

Under Article 19, only the claims may be amended.

During the international phase, the claims may also be amended (or further amended) under Article 34 before the International Preliminary Examining Authority. The description and drawings may only be amended under Article 34 before the International Preliminary Examining Authority.

Upon entry into the national phase, all parts of the international application may be amended under Article 28 or, where applicable, Article 41.

When? Within 2 months from the date of transmittal of the international search report or 16 months from the priority date, whichever time limit expires later. It should be noted, however, that the amendments will be considered as having been received on time if they are received by the International Bureau after the expiration of the applicable time limit but before the completion of the technical preparations for international publication (Rule 46.1).

Where not to file the amendments?

The amendments may only be filed with the International Bureau and not with the receiving Office or the International Searching Authority (Rule 46.2).

Where a demand for international preliminary examination has been/is filed, see below.

How? Either by cancelling one or more entire claims, by adding one or more new claims or by amending the text of one or more of the claims as filed.

A replacement sheet must be submitted for each sheet of the claims which, on account of an amendment or amendments, differs from the sheet originally filed.

All the claims appearing on a replacement sheet must be numbered in Arabic numerals. Where a claim is cancelled, no renumbering of the other claims is required. In all cases where claims are renumbered, they must be renumbered consecutively (Section 205(b)).

The amendments must be made in the language in which the international application is to be published.

What documents must/may accompany the amendments?

Letter (Section 205(b)) :

The amendments must be submitted with a letter.
The letter will not be published with the international application and the amended claims. It should not be confused with the "Statement under Article 19(1)" (see below, under "Statement under Article 19(1)").
The letter must be in English or French, at the choice of the applicant. However, if the language of the international application is English, the letter must be in English; if the language of the international application is French, the letter must be in French.

The letter must indicate the differences between the claims as filed and the claims as amended. It must, in particular, indicate, in connection with each claim appearing in the international application (it being understood that identical indications concerning several claims may be grouped), whether

(i) the claim is unchanged;
(ii) the claim is cancelled;
(iii) the claim is new;
(iv) the claim replaces one or more claims as filed;
(v) the claim is the result of the division of a claim as filed.

The following examples illustrate the manner in which amendments must be explained in the accompanying letter :

1. [Where originally there were 48 claims and after amendment of some claims there are 51]:
"Claims 1 to 29, 31, 32, 34, 35, 37 to 48 replaced by amended claims bearing the same numbers; claims 30, 33 and 36 unchanged; new claims 49 to 51 added."

2. [Where originally there were 15 claims and after amendment of all claims there are 11]:
"Claims 1 to 15 replaced by amended claims 1 to 11."

3. [Where originally there were 14 claims and the amendments consist in cancelling some claims and in adding new claims]:
"Claims 1 to 6 and 14 unchanged; claims 7 to 13 cancelled; new claims 15, 16 and 17 added." or
"Claims 7 to 13 cancelled; new claims 15, 16 and 17 added; all other claims unchanged."

4. [Where various kinds of amendments are made]:
"Claims 1-10 unchanged; claims 11 to 13, 18 and 19 cancelled; claims 14, 15 and 16 replaced by amended claim 14; claim 17 subdivided into amended claims 15, 16 and 17; new claims 20 and 21 added."

"Statement under Article 19(1)" (Rule 46.4)

The amendments may be accompanied by a statement explaining the amendments and indicating any impact that such amendments might have on the description and the drawings (which cannot be amended under Article 19(1)).

The statement will be published with the international application and the amended claims.

It must be in the language in which the international application is to be published.

It must be brief, not exceeding 500 words if in English or if translated into English.

It should not be confused with and does not replace the letter indicating the differences between the claims as filed and as amended. It must be filed on a separate sheet and must be identified as such by a heading, preferably by using the words "Statement under Article 19(1)."

It may not contain any disparaging comments on the international search report or the relevance of citations contained in that report. Reference to citations, relevant to a given claim, contained in the international search report may be made only in connection with an amendment of that claim.

Consequence if a demand for international preliminary examination has already been filed

If, at the time of filing any amendments and any accompanying statement, under Article 19, a demand for international preliminary examination has already been submitted, the applicant must preferably, at the time of filing the amendments (and any statement) with the International Bureau, also file with the International Preliminary Examining Authority a copy of such amendments (and of any statement) and, where required, a translation of such amendments for the procedure before that Authority (see Rules 55.3(a) and 62.2, first sentence). For further information, see the Notes to the demand form (PCT/IPEA/401).

If a demand for international preliminary examination is made, the written opinion of the International Searching Authority will, except in certain cases where the International Preliminary Examining Authority did not act as International Searching Authority and where it has notified the International Bureau under Rule 66.1*bis*(b), be considered to be a written opinion of the International Preliminary Examining Authority. If a demand is made, the applicant may submit to the International Preliminary Examining Authority a reply to the written opinion together, where appropriate, with amendments before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later (Rule 43*bis*.1(c)).

Consequence with regard to translation of the international application for entry into the national phase

The applicant's attention is drawn to the fact that, upon entry into the national phase, a translation of the claims as amended under Article 19 may have to furnished to the designated/elected Offices, instead of, or in addition to, the translation of the claims as filed.

For further details on the requirements of each designated/elected Office, see the *PCT Applicant's Guide*, Volume II.

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

RECEIVED
2008 MAR 20 A 9:55
U.S. PATENT & TRADEMARK OFFICE

Applicant's or agent's file reference 83636-13	**FOR FURTHER ACTION**	see Form PCT/ISA/220 as well as, where applicable, item 5 below
International application No. **PCT/CA2007/002150**	International filing date *(day/month/year)* 29 November 2007 (29-11-2007)	(Earliest) Priority date *(day/month/year)* 29 November 2006 (29-11-2006)

Applicant
DIGIFONICA (INTERNATIONAL) LIMITED ET AL

This international search report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This international search report consists of a total of 4 sheets.

[X] It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**

a. With regard to the **language**, the international search was carried out on the basis of:

[X] the international application in the language in which it was filed

[ ] a translation of the international application into , which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b))

b. [ ] This international search report has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43.6*bis*(a)).

c. [ ] With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, see Box No. I

2. [ ] **Certain claims were found unsearchable** (see Box No. II)

3. [ ] **Unity of invention is lacking** (see Box No. III)

4. With regard to the **title**,

[X] the text is approved as submitted by the applicant

[ ] the text has been established by this Authority to read as follows :

5. With regard to the **abstract**,

[X] the text is approved as submitted by the applicant

[ ] the text has been established, according to Rule 38.2, by this Authority as it appears in Box No. IV. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority

6. With regard to the **drawings**,

a. the figure of the **drawings** to be published with the abstract is Figure No. 1

[X] as suggested by the applicant

[ ] as selected by this Authority, because the applicant failed to suggest a figure

[ ] as selected by this Authority, because this figure better characterizes the invention

b. [ ] none of the figures is to be published with the abstract

Doc
JLG
TNM

T-Mobile Exhibit 1002
Part 4 of 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2007/002150

A. CLASSIFICATION OF SUBJECT MATTER
IPC: ***H04L 12/26*** (2006.01) , ***H04L 12/66*** (2006.01) , ***H04M 11/06*** (2006.01) , ***H04M 3/22*** (2006.01)
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC: ***H04L 12/26*** (2006.01) , ***H04L 12/66*** (2006.01) , ***H04M 11/06*** (2006.01) , ***H04M 3/22*** (2006.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
West, Delphion, Canadian Patents Database, IEEEXplore, Google
Keywords: lawful intercept, (monitor* OR record* or intercept*) near (communicat* OR voip OR phone call* OR audio OR video), electronic surveillance, intercept* near device*, intercept* same IP network*, record* same IP network*, intercept* same voip, record* same voip, media relay, mediation device, (intercept* field OR flag) same profile

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y	US 2004/0181599 A1 (Kreusch et al.) 16 September 2004 (16-09-2004) *paragraphs [0011]-[0015], [0019]-[0022], [0028], [0034]-[0036], [0048]-[0053], [0055]-[0061], [0067], [0072]-[0074], [0078]-[0083]; Figs. 1, 2a-2b; claims 1-3, 7-8, 25-26*	1-2, 4-5, 10-15, 17-18, 23-26 3, 6-9, 16, 19-22
X	US 2003/0219103 A1 (Rao et al.) 27 November 2003 (27-11-2003) *Abstract; paragraphs [0005], [0026]-[0037], [0051]-[0062], [0071]-[0079], [0086]-[0090]; Figs. 1-9*	1, 14
Y	US 2002/0051518 A1 (Bondy et al.) 2 May 2002 (02-05-2002) *Abstract; paragraphs [0030]-[0032], [0036]-[0037], [0039], [0044]-[0052], [0055]-[0057], [0060]; Figs 1, 3, 5; claims 1-5*	3, 6-9, 16, 19-22
A	EP 1 389 862 B1 (Shen et al.) 3 November 2004 (03-11-2004) *paragraphs [0007]-[0014], [0051]-[0060]; Fig. 2; claim 1*	1-3, 14-16
A	US 2004/0165709 A1 (Pence et al.) 26 August 2004 (26-08-2004) *whole document*	1-26

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

*	Special categories of cited documents :
"A"	document defining the general state of the art which is not considered to be of particular relevance
"E"	earlier application or patent but published on or after the international filing date
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O"	document referring to an oral disclosure, use, exhibition or other means
"P"	document published prior to the international filing date but later than the priority date claimed
"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&"	document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
3 March 2008 (03-03-2008)	14 March 2008 (14-03-2008)

Name and mailing address of the ISA/CA	Authorized officer
Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	Daniela Savin 819- 934-4890

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2004/0255126 A1 (Reith) 16 December 2004 (16-12-2004) *paragraphs [0010]-[0014], [0020]-[0031], [0041]-[0046]; Figs. 1-2, 4-6; claims 1, 6-10*	1-26
A	US 2004/0157629 A1 (Kallio et al.) 12 August 2004 (12-08-2004) *paragraphs [0006]-[0021], [0050]-[0057], [0080]-[0109]; Figs. 1-12; claims 1, 7-23, 29-43*	1-26
A	US 2005/0174937 A1 (Scoggins et al.) 11 August 2005 (11-08-2005) *paragraphs [0068]-[0089], [0112]-[0138], [0153]-[0156], [0173]-[0176], [0184]-[0193]; Figs. 1-11; claims 1-2*	1-26

PATENT COOPERATION TREATY

From the
INTERNATIONAL SEARCHING AUTHORITY

To:
SMART & BIGGAR
Box 11560 Vancouver Centre
2200 - 650 W. Georgia Street
VANCOUVER, British Columbia
Canada, V6B 4N8

PCT RECEIVED

WRITTEN OPINION OF THE
INTERNATIONAL SEARCHING AUTHORITY 2008 MAR 06 A 9:54

(PCT Rule 43*bis*.1) 650 WEST GEORGIA ST. VANCOUVER, B.C.

Date of mailing *(day/month/year)*	14 March 2008 (14-03-2008)
Applicant's or agent's file reference 83636-13	**FOR FURTHER ACTION** See paragraph 2 below

International application No:	International filing date *(day/month/year)*	Priority date *(day/month/year)*
PCT/CA2007/002150	29 November 2007 (29-11-2007)	29 November 2006 (29-11-2006)

International Patent Classification (IPC) or both national classification and IPC
IPC: ***H04L 12/26*** (2006.01), ***H04L 12/66*** (2006.01), ***H04M 11/06*** (2006.01), ***H04M 3/22*** (2006.01)

Applicant
DIGIFONICA (INTERNATIONAL) LIMITED ET AL

WRITTEN OPINION DUE: ON 29/08 DOCKET Docket SEP 26/08

1. This opinion contains indications relating to the following items :

[X]	Box No. I	Basis of the opinion
[ ]	Box No. II	Priority
[ ]	Box No. III	Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
[ ]	Box No. IV	Lack of unity of invention
[X]	Box No. V	Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
[ ]	Box No. VI	Certain documents cited
[ ]	Box No. VII	Certain defects in the international application
[X]	Box No. VIII	Certain observations on the international application

2. **FURTHER ACTION**
If a demand for international preliminary examination is made, this opinion will be considered to be a written opinion of the International Preliminary Examining Authority ("IPEA") except that this does not apply where the applicant chooses an Authority other than this one to be the IPEA and the chosen IPEA has notified the International Bureau under Rule 66.1*bis*(b) that written opinions of this International Searching Authority will not be so considered.

If this opinion is, as provided above, considered to be a written opinion of the IPEA, the applicant is invited to submit to the IPEA a written reply together, where appropriate, with amendments, before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later.

For further options, see Form PCT/ISA/220.

3. For further details, see notes to Form PCT/ISA/220.

Name and mailing address of the ISA/CA	Date of completion of this opinion	Authorized officer
Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	04 March 2008 (04-03-2008)	Daniela Savin 819- 934-4890

Form PCT/ISA/237 (cover sheet) (April 2007) Page 1 of 6

DOC JLC Tmk

T-Mobile Exhibit 1002
Part 4 of 6

Box No. I **Basis of this opinion**

1. With regard to the **language**, this opinion has been established on the basis of:

   [X] the international application in the language in which it was filed

   [ ] a translation of the international application into , which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b)).

2. [ ] This opinion has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43*bis*.1(a))

3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application and necessary to the claimed invention, this opinion has been established on the basis of :

   a. type of material

      [ ] a sequence listing

      [ ] table(s) related to the sequence listing

   b. format of material

      [ ] on paper

      [ ] in electronic form

   c. time of filing/furnishing

      [ ] contained in the international application as filed.

      [ ] filed together with the international application in electronic form

      [ ] furnished subsequently to this Authority for the purposes of search.

4. [ ] In addition, in the case that more than one version or copy of a sequence listing and/or table(s) relating thereto has been filed or furnished, the required statements that the information in the subsequent or additional copies is identical to that in the application as filed or does not go beyond the application as filed, as appropriate, were furnished.

5. Additional comments :

Box No. V **Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement**

1. Statement

Novelty (N)	Claims	3, 6-9, 12-13, 16, 19-22, 25-26	YES
	Claims	1-2, 4-5, 10-11, 14-15, 17-18, 23-24	NO
Inventive step (IS)	Claims	None	YES
	Claims	1-26	NO
Industrial applicability (IA)	Claims	1-26	YES
	Claims	None	NO

2. Citations and explanations :

Reference is made to the following documents cited in the International Search Report:
D1: US 2004/0181599 A1 (Kreusch et al.) 16 September 2004 (16-09-2004)
D2: US 2003/0219103 A1 (Rao et al.) 27 November 2003 (27-11-2003)
D3: US 2002/0051518 A1 (Bondy et al.) 2 May 2002 (02-05-2002)

2.1 **Novelty**

a) **Claims 1 and 14 do not comply** with **PCT Article 33(2)**. D1 or D2 disclosed the claimed subject matter before the claim date.

Using the wording of **claim 1**, D1 or D2 each discloses (*references in parenthesis apply to D1 or D2*):

A method for intercepting communications in an Internet Protocol network, the method comprising (*see D1: paragraphs [0011], [0022]; Fig. 1; claims 1-3; or see D2: Abstract; Figs. 1-9*):

maintaining dialing profiles for respective subscribers to the IP network, each said dialing profile including a username associated with the corresponding subscriber (*see D1: paragraphs [0019]-[0021], [0034]-[0036], [0053]; claims 7-8; or see D2: paragraphs [0027], [0029]*);

associating intercept information with the dialing profile of a subscriber whose communications are to be monitored, said intercept information including determination information for determining whether to intercept a communication involving said subscriber, and destination information identifying a device to which intercepted communications involving said subscriber are to be sent (*see D1: paragraphs [0055], [0067], [0078]; Figs. 2a-2b; or see D2: paragraphs [0027], [0029], [0052], [0072]*); and

when said determination information meets intercept criteria, communicating with a media relay through which said communications involving said subscriber will be conducted or are being conducted to cause said media relay to send a copy of said communications to a mediation device specified by said destination information (*see D1: paragraphs [0012], [0015], [0019], [0022], [0028], [0053], [0055], [0060]-[0061], [0081]-[0083]; claim 7; or see D2: paragraphs [0026]-[0037], [0051]-[0062], [0071]-[0079]; [0086]-[0090]; Figs. 1-9*).

Since the language of claim 1 reads on the prior art represented by D1 or D2, the subject matter of this claim is considered to lack novelty.

As **independent claim 14** is of the same scope as claim 1, it is similarly considered to lack novelty, for the same reasons indicated above for claim 1.

b) **Claims 2, 4-5, 10-11, 15, 17-18 and 23-24 do not comply** with **PCT Article 33(2)**. D1 disclosed the claimed subject matter before the claim date.

Regarding **claim 2**, D1 discloses associating intercept information with the subscriber profile at login time, when communications involving the subscriber are not in progress (*see D1: paragraphs [0048]-[0053], [0072]-[0074]*).

Box No. VIII Certain observations on the international application

The following observations on the clarity of the claims, description, and drawings or on the question whether the claims are fully supported by the description, are made :

The description does not comply with **Article 5** of the **PCT**. All documents referred to in the description of an application must be available to the public. Reference to the document on page 1, lines 5-6 must be deleted or replaced by its corresponding patent number or publication number.

Claims 1 and 14 are indefinite and do not comply with **PCT Article 6**. The following terms have no antecedents: "*dialing profile*" (claim 1, line 6; claim 14, line 8), "*intercept information*" (claim 1, line 10; claim 14, line 12) and "*intercept criteria*" (claim 1, line 18; claim 14, line 25).

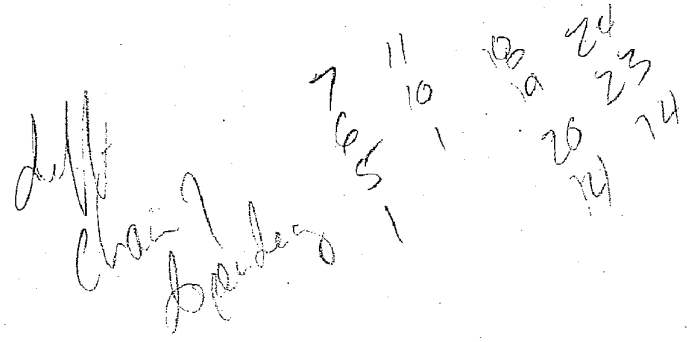
Claims 1 and 14 are indefinite and do not comply with **PCT Article 6**. The inclusion of "... identifying *a device* to which intercepted communications ..." (claim 1, lines 14-15; claim 14, lines 17-18) causes ambiguity. It is not clear whether "a device" refers to a mediation device or any other type of device.

Claims 2-3 and 15-16 are indefinite and do not comply with **PCT Article 6**. The inclusion of "method of *clam* 1" and "apparatus of *clam* 14" causes ambiguity. The applicant likely meant "method of *claim* 1" and "apparatus of *claim* 14".

Claim 11 is unclear and does not comply with **PCT Article 6**. The inclusion of "... *identify a media relay* ..." causes ambiguity. It is not clear whether the media relay is identified by the intercept request message handler as recited by claim 11, or in response to the routing message, as recited by claim 7. The same comment applies to claim 24 (but in regards to routing message recited by claim 20), since claim 24 is of the same scope as claim 11.

Claim 12 is unclear and does not comply with **PCT Article 6**. The inclusion of "maintaining *a active call records*" causes ambiguity, as the aforementioned expression is not grammatically correct.

Claim 24 is unclear and does not comply with **PCT Article 6**. The inclusion of "a) means for *find* ..." causes ambiguity. It is suggested that "means for find ..." should read "means for *finding* ...", in order to be consistent with the tense used for the other verbs in claim 24.



Supplemental Box

In case the space in any of the preceding boxes is not sufficient.

Continuation of: Box V

Regarding **claim** 4, D1 discloses populating the intercept information fields in the hidden database, which is similar to the dialing profile recited by this claim *(see D1: paragraph [0079]; claims 7-8, 25-26)*.

As **claim 17** is of the same scope as claim 4, it is similarly considered to lack novelty, for the same reasons indicated above for claim 4.

Regarding **claim 5**, D1 discloses the routing message containing at least part of the intercept information, when determination information meets intercept criteria *(see D1: paragraphs [0055]-[0060], [0067], [0072], [0081]; Figs. 2a-2b)*.

As **claim 18** is of the same scope as claim 5, it is similarly considered to lack novelty, for the same reasons indicated above for claim 5.

Regarding **claims 10-11**, D1 discloses that in response to receipt of an intercept request message, the corresponding message handler finds the dialing profile associated with the subscriber whose communications are to be monitored, associates the intercept information with the dialing profile, determines whether the intercept criteria are met and identifies the media relay through which communications are being conducted *(see D1: paragraphs [0019], [0067], [0072], [0079])*.

As **claims 23-24** are of the same scope as claims 10-11, they are similarly considered to lack novelty, for the same reasons indicated above for claims 10-11.

However, **claims 3, 6-9, 12-13, 16, 19-22** and **25-26** appear to be novel and are deemed to **comply** with **PCT Article 33(2)** since the search of the prior art has not revealed a single document disclosing the claimed subject matter.

2.2 Inventive Step

c) **Claims 1 and 14** do not involve an inventive step over D1 or D2 since they are not novel over D1 or D2, and therefore they **do not comply** with **PCT Article 33(3)**. **Claims 2, 4-5, 10-11, 15, 17-18 and 23-24** do not involve an inventive step over D1 since they are not novel over D1, and therefore they **do not comply** with **PCT Article 33(3)**.

e) **Claim 3 does not comply** with **PCT Article 33(3)**. The subject matter of this claim does not appear to involve an inventive step in view of D1 and D3. D1 does not specifically disclose associating the intercept information with the dialing profile when communications involving the subscriber are in progress. D1 mentions that the interception is possible even when the interception subject is mobile and changes location (*see D1: paragraph [0013]*). D3, however complements the teachings of D1 by disclosing the interception happening not only at login or call origination, but also for any service invocation during the call, such as call waiting, conference call, call forwarding or message retrieval (*see D3: paragraph [0030]; claims 4-5*).

As **claim 16** is of the same scope as claim 3, it is similarly considered to lack an inventive step in view of D1 and D3.

f) **Claim 6 does not comply** with **PCT Article 33(3)**. The subject matter of this claim does not appear to involve an inventive step in view of D1 and D3. D1 does not specifically disclose determining whether a current date and time is within a range specified by the determination information. However, D3 discloses the interception information containing the start and stop time for the interception (*see D3: paragraph [0039]; claim 3*), and it would have been obvious for someone skilled in the art to add these start and stop times to the determination information taught by D1, in order to determine whether to intercept a call or not, depending on the current time.

As **claim 19** is of the same scope as claim 6, it is similarly considered to lack an inventive step in view of D1 and D3.

g) **Claims 7-9 do not comply** with **PCT Article 33(3)**. The subject matter of these claims does not appear to involve an inventive step in view of D1 and D3. D1 does not specifically disclose the routing message containing an identification of the media relay through which communications and interception will be conducted, and pre-associating this media relay with the dialing profile. However, the inclusion of the media relay address in the routing message is a normal design procedure, since no communication would be possible without the presence of the media relay. Moreover, D3 discloses the surveillance information containing all the entities comprising the surveillance path, including the identification of the media relay, and it would have been obvious for someone skilled in the art to include this identification to the monitoring information stored in the hidden database taught by D1 (*see D3: paragraphs [0032], [0044], [0051]-[0052], [0057], [0060]; claims 1-3*).

As **claims 20-22** are of the same scope as claims 7-9, they are similarly considered to lack an inventive step in view of D1 and D3.

Supplemental Box

In case **the space in any of the preceding boxes is not sufficient.**

Continuation of: Supplemental sheet on page 5

h **Claims 12-13 do not comply** with **PCT Article 33(3)**. The subject matter of these claims does not appear to involve an inventive step in view of D1 and the common knowledge in the art. D1 does not specifically disclose maintaining a list of all the active call records, linking the username with the corresponding media relay identifier, and another list that associates every PST telephone number with its corresponding username, for all the subscribers in the network. However, creating and maintaining these lists would have been an obvious design alternative on the claim date to a person skilled in the art having regards to D1, since keeping a record of all the calls in progress is a standard functionality of any generic network management entity.

As **claims 25-26** are of the same scope as claims 12-13, they are similarly considered to lack an inventive step in view of D1 and the common knowledge in the art.

2.3 Industrial Applicability

Claims 1-26 are considered to be industrially applicable as per **PCT Article 33(4)**.

INFORMATION DISCLOSURE STATEMENT

Applicant	:	Johan Emil Viktor Björsell
App. No	:	13/056277
Filed	:	January 27, 2011
For	:	MOBILE GATEWAY
Examiner	:	Unassigned
Art Unit	:	2473
Conf No.	:	2403

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Enclosed for filing in the above-identified application is a PTO/SB/08 Equivalent listing 113 references, of which 18 are enclosed/submitted.

This Information Disclosure Statement is being filed before the receipt of a first Office Action on the merits, and presumably no fee is required. If a first Office Action on the merits was mailed before the mailing date of this Statement, the Commissioner is authorized to charge the fee set forth in 37 C.F.R. § 1.17(p) to Deposit Account No. 11-1410.

Respectfully submitted,
KNOBBE, MARTENS, OLSON & BEAR, LLP

Dated: 5/5/11

By: _____
John M. Carson
Registration No. 34,303
Attorney of Record
Customer No. 20995
(858) 836-9000

11005191\cey
040511

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
13/056,277	01/27/2011	Johan Emil Viktor Bjorsell	SMARB19.004APC

CONFIRMATION NO. 2403

20995
KNOBBE MARTENS OLSON & BEAR LLP
2040 MAIN STREET
FOURTEENTH FLOOR
IRVINE, CA 92614

PUBLICATION NOTICE



Title:MOBILE GATEWAY

Publication No.US-2011-0122827-A1
Publication Date:05/26/2011

NOTICE OF PUBLICATION OF APPLICATION

The above-identified application will be electronically published as a patent application publication pursuant to 37 CFR 1.211, et seq. The patent application publication number and publication date are set forth above.

The publication may be accessed through the USPTO's publically available Searchable Databases via the Internet at www.uspto.gov. The direct link to access the publication is currently http://www.uspto.gov/patft/.

The publication process established by the Office does not provide for mailing a copy of the publication to applicant. A copy of the publication may be obtained from the Office upon payment of the appropriate fee set forth in 37 CFR 1.19(a)(1). Orders for copies of patent application publications are handled by the USPTO's Office of Public Records. The Office of Public Records can be reached by telephone at (703) 308-9726 or (800) 972-6382, by facsimile at (703) 305-8759, by mail addressed to the United States Patent and Trademark Office, Office of Public Records, Alexandria, VA 22313-1450 or via the Internet.

In addition, information on the status of the application, including the mailing date of Office actions and the dates of receipt of correspondence filed in the Office, may also be accessed via the Internet through the Patent Electronic Business Center at www.uspto.gov using the public side of the Patent Application Information and Retrieval (PAIR) system. The direct link to access this status information is currently http://pair.uspto.gov/. Prior to publication, such status information is confidential and may only be obtained by applicant using the private side of PAIR.

Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 1-866-217-9197.

Office of Data Managment, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

INFORMATION DISCLOSURE STATEMENT BY APPLICANT	Application No.	13/056,277
	Filing Date	January 27, 2011
	First Named Inventor	Johan Emil Viktor Bjorsell et al.
	Art Unit	2473
(Multiple sheets used when necessary)	Examiner	Kwang Bin Yao
SHEET 1 OF 1	Attorney Docket No.	SMARB19.004APC

U.S. PATENT DOCUMENTS					
Examiner Initials	Cite No.	Document Number *Number - Kind Code (if known)* Example: 1,234,567 B1	Publication Date MM-DD-YYYY	Name of Patentee or Applicant	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear

FOREIGN PATENT DOCUMENTS						
Examiner Initials	Cite No.	Foreign Patent Document *Country Code-Number-Kind Code* Example: JP 1234567 A1	Publication Date MM-DD-YYYY	Name of Patentee or Applicant	Pages, Columns, Lines Where Relevant Passages or Relevant Figures Appear	T[1]
	1	EP 1 389 862 B1	11/3/2004	Shen et al.		

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[1]
	2	A copy of the International Preliminary Report on Patentability issued on September 29, 2009 for PCT/CA2008/000545.	
	3	A copy of the International Preliminary Report on Patentability issued on March 20, 2012 for PCT/CA2009/001317.	
	4	Townsley, et al.; "RFC 2661 - Layer Two Tunneling Protocol 'L2TP' ", August 1999.	
	5	IP2Location, http://www.ip2location.com/; printed June 20, 2012.	
	6	DOTS IP Address Validation, "Overview", http://www.serviceobjects.com/products/dots_ipgeo.asp; printed June 21, 2012.	
	7	List of North American Numbering Plan area codes, http://en.wikipedia.org/wiki/List_of_NANP_area_codes; printed June 20, 2012.	
	8	DOTS Phone Exchange, "Overview", http://www.serviceobjects.com/demos/PhoneExchangeDemo.asp (URL no longer valid, current URL is http://www.serviceobjects.com/products/phone/phone-exchange); printed June 21, 2012.	
	9	Rosenberg, et al.; "RFC 3261 - SIP: Session Initiation Protocol", June 2002.	

13465299-sjw 061512

Examiner Signature	Date Considered
***Examiner:** Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.	

T[1] - Place a check mark in this area when an English language Translation is attached.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 389 862 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
03.11.2004 Bulletin 2004/45

(51) Int Cl.[7]: **H04L 29/06**, H04M 7/00

(21) Application number: **02360235.2**

(22) Date of filing: **08.08.2002**

(54) **Lawful interception for VoIP calls in IP based networks**

Legales Abfangen für VOIP Anrufe in einem IP-Fernmeldenetz

Interception legale pour appels VOIP dans un réseau de telecommunications IP

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SK TR
Designated Extension States:
AL LT LV MK RO SI

(43) Date of publication of application:
18.02.2004 Bulletin 2004/08

(73) Proprietor: **ALCATEL**
75008 Paris (FR)

(72) Inventors:
- **Shen, Yuzhong**
  70499 Stuttgart (DE)
- **Gorges, Thomas**
  71638 Ludwigsburg (DE)

(74) Representative:
Menzietti, Domenico, Dipl.-Ing et al
Alcatel
Intellectual Property Department, Stuttgart
70430 Stuttgart (DE)

(56) References cited:
EP-A- 1 111 892
WO-A-01/89145
WO-A-02/15627
WO-A-99/17499

- **THERNELIUS F: "SIP, NAT, and Firewalls" MASTER'S THESIS, KUNGST TEKNISKA HÖGSKOLAN, DEPARTMENT OF TELEINFORMATICS - ERICSSON, May 2000 (2000-05), XP002209773**

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

Printed by Jouve, 75001 PARIS (FR)

EP 1 389 862 B1

T-Mobile Exhibit 1002
Part 4 of 6

Description

TECHNICAL FIELD OF THE INVENTION

[0001] This invention is related in general to the field of telecommunications systems. More particularly, the invention is related to a lawful interception device for media streams, in particular VoIP calls in IP based networks.

BACKGROUND OF THE INVENTION

[0002] Current lawful interceptions are deployed in class4/class5 switches of PSTN/PLMN networks. In 3G/UMTS or next generation networks, a connection may be IP end to end. No traffics will go through class 5/class4 switches. That means current lawful interception solutions cannot be used here. One solution may undertake an analysis of IP packets in a related network node, but it's difficult to know which route a call (media stream) will take through the network.

[0003] In Thernelius F: "SIP, NAT, and Firewalls' Master's Thesis, Kungst Tekniska Hoegskolan, Department of Teleinformatics-Ericsson, May 200 (2000-05), a method for performing SIP signalling for a media stream is disclosed. The method includes receiving a SIP invite message of a first IP party, adapting at least one connection parameter in the SDP of the received SIP invite message, transmitting the adapted SIP invite message to a second IP party, receiving a SIP response message from the second IP party, adapting at least one connection parameter in the SDP of the received SIP response message, and transmitting the adpated SIP response message to the first IP party.

[0004] In WO 02 15627 A a method for providing a mode selection procedure is disclosed. A network element is adapted to perform a mode selection procedure for selecting the same mode for bidirectional communication between the network elements. The mode selection ensures the use of one and the same mode in uplink and downlink direction and thus enables e.g. IP telephony in UMTS using SIP protocol.

[0005] In EP-A-1 111 1 892 a method for IP network surveillance is disclosed. An authentication server associated with an end user device determines whether the end user device is under surveillance. In case of surveillance the authentication server notifies a proxy server which copies call signaling information, extracts media stream identification and decoding information thereof and forwards it to an edge router which copies the media stream of the end user.

[0006] In WO 01 89145 A a method of monitoring calls with a mobile terminal in an IP-based network is disclosed. Whem the mobile terminal sends an Admission Request message to the Gatekeeper, the Gatekeeper sends a query the Monitoring Station asking whether the mobile terminal is to be monitored. The Monitoring Station sends a reply to the Gatekeeper indicating that the mobile station is to be monitored and provides an IP address where monitored packets are to be sent. The Gatekeeper then sends a monitoring request message to the access router associated with the mobile terminal. The request identifies the mobile terminal to be monitored, instructs the access router to monitor the mobile station, and provides a unique call identification and the IP address where monitored packets are to be sent. When the access router detects a packet associated with the mobile station, the router sends all packets associated with the mobile terminal to the Monitoring Station.

SUMMARY OF THE INVENTION

[0007] It is an object of the invention to provide a lawful interception device for VoIP calls in IP based networks.

[0008] The inventive lawful interception device detects information in the signalling information being transmitted between two IP parties and generates instructions out of the detected signalling information for instructing a RTP (Real-time Transport Protocol) proxy server to create channels to bypass a VoIP call to be intercepted via an intermediate storage medium. Instead of voice each media stream could be intercepted, e.g. data, internet access, e-mail, video, real-time pictures, etc.

[0009] In a SIP (Session Initiation Protocol) interception proxy server, where interception should be controlled, applications for interception are running to chose calls for interception. If a call should be monitored, the SIP proxy server has first to hold the invite message from A party. There are listening information in SDP (session description protocol) part of invite message.

[0010] SIP proxy server then instructs a RTP proxy server via a RTP proxy control interface to allocate a bypass channel for monitoring the media stream (A channel: sending to A party). The RTP information of this bypass channel (listening part: ip and port) is included in SDP part in the SIP invite message and passed to its destination.

[0011] When SIP proxy server has received a response of B party, he instructs RTP proxy via RTP proxy control interface to allocate another bypass channel for monitoring the media stream (B channel: sending to B party). The RTP information of this second bypass channel (listening part: ip and port) is included in SDP part in SIP ok message and send to its origination (A party).

[0012] After session setup, both parties will start RTP connections to RTP proxy server depending on connection parameters in its received SIP messages. But those are transparent to A and B. They do not know they are connected to a RTP proxy.

[0013] The RTP proxy can start record both media channels (A and B). At the end of this call, e.g. a media file with two sound tracks will be created by RTP proxy.

Advantages:

[0014]

- centralized network node to intercept media streams,
- low cost of deployment,
- transparent to end users,
- the RTP proxy can also be used in the same way as above in a media gateway control (MEGACO, H. 248) based network or H.323 network.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] For a better understanding of the present invention, reference is made to the accompanying drawings, in which:

FIG. 1 is a simplified block diagram of a portion of an exemplary telecommunications network according to the teachings of the prior art;

FIG. 2 is a simplified block diagram of a portion of an exemplary telecommunications network according to the teachings of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0016] FIG. 1 shows a portion of an exemplary telecommunications network according to the teachings of the prior art.

[0017] Two IP parties, e.g. yshen@ alcatel.de and eric@alcatel.com, are interconnected via two networks: a SIP signaling network and a transmission network. Via the SIP signaling network signaling is performed, e.g. a connection is established between the two IP parties. Via the transmission network the information to be transmitted, e.g. voice, data, etc. is transmitted in media streams (RTP session).

[0018] In the SIP based network, each SIP proxy server is responsible for signaling and session monitoring. The media stream will go from one IP endpoint to another IP endpoint. There is no need of a centralized media path like in PSTN network. A lawful interception of media stream could be done only in the network layer.

[0019] Recording media stream by analyzing network traffics for lawful interception is very expensive, due to the packet route through the IP network could change. Therefor the recording could only be done very closely to the endpoints. Additionaly a resembling of recorded packets is needed. A playing in real time will be difficult.

[0020] In the following definition and background information is provided regarding SIP, proxy server, RTP, SDP, etc.

SIP:

[0021] The Session Initiation Protocol (SIP) is an application-layer control (signaling) protocol for creating, modifying and terminating sessions with one or more participants. These sessions include Internet multimedia conferences, Internet telephone calls and multimedia distribution. Members in a session can communicate via multicast or via a mesh of unicast relations, or a combination of these.

[0022] SIP invitations used to create sessions carry session descriptions which allow participants to agree on a set of compatible media types. SIP supports user mobility by proxying and redirecting requests to the user's current location. Users can register their current location. SIP is not tied to any particular conference control protocol. SIP is designed to be independent of the lower-layer transport protocol and can be extended with additional capabilities.

[0023] The Session Initiation Protocol (SIP) is an application-layer control protocol that can establish, modify and terminate multimedia sessions or calls. These multimedia sessions include multimedia conferences, distance learning, Internet telephony and similar applications. SIP can invite both persons and "robots", such as a media storage service. SIP can invite parties to both unicast and multicast sessions; the initiator does not necessarily have to be a member of the session to which it is inviting. Media and participants can be added to an existing session.

[0024] SIP can be used to initiate sessions as well as invite members to sessions that have been advertised and established by other means. Sessions can be advertised using multicast protocols such as electronic mail, news groups, web pages or directories (LDAP), among others.

[0025] SIP transparently supports name mapping and redirection services, allowing the implementation of ISDN and Intelligent Network telephony subscriber services. These facilities also enable personal mobility. In the parlance of telecommunications intelligent network services, this is defined as: "Personal mobility is the ability of end users to originate and receive calls and access subscribed telecommunication services on any terminal in any location, and the ability of the network to identify end users as they move. Personal mobility is based on the use of a unique personal identity (i.e., personal number)." Personal mobility complements terminal mobility, i.e., the ability to maintain communications when moving a single end system from one subnet to another.

[0026] SIP supports five facets of establishing and terminating multimedia communications:

User location: determination of the end system to be used for communication;

User capabilities: determination of the media and media parameters to be used;

T-Mobile Exhibit 1002
Part 4 of 6

User availability: determination of the willingness of the called party to engage in communications;

Call setup: "ringing", establishment of call parameters at both called and calling party;

Call handling: including transfer and termination of calls.

[0027] SIP can also initiate multi-party calls using a multipoint control unit (MCU) or fully-meshed interconnection instead of multicast. Internet telephony gateways that connect Public Switched Telephone Network (PSTN) parties can also use SIP to set up calls between them.

[0028] SIP is designed as part of the overall IETF multimedia data and control architecture currently incorporating protocols such as the real-time transport protocol (RTP) for transporting real-time data and providing QoS feedback.

[0029] A request and a response form together a transaction. SIP uses e.g. invite and ack messages to build up connections. Other messages used are e.g. ok, bye, options, register, cancel. SIP parties are identified via a SIP-ULR, e.g.: sip:clientname@hostaddress. Each client may transmit requests to a proxy server or directly to an IP address.

[0030] An establishment of a connection is perfomed is three steps: sending an invite (request) message from a first IP party to a second IP party, sending an ok (response) message from the second IP party to the first IP party, sending an ack (response) message from the frist IP party to the second IP party. The invite message includes as much information as needed to allow the second IP party to judge whether a connection is wanted or not. The ack message is an acknowledgement, which serves to increase savety of the connection. SIP is thus not dependent on TCP or UDP.

[0031] The SIP according to the invention is the SIP currently standardized and modifications thereof and equivalents thereof.

RTP:

[0032] The Audio/Video Transport Working Group of IETF was formed to specify a protocol for real-time transmission of audio and video over UDP and IP multicast. This is the Real-time Transport Protocol, RTP, together with its associated profile for audio/video conferences and payload format documents. The payload formats currently under discussion include a number of media specific formats (MPEG-4, DTMF, PureVoice) and FEC techniques applicable to multiple formats (parity FEC, Reed-Solomon coding). RTP is used to replace a normal circuit-switched trunk between two nodes.

[0033] The real-time transport protocol (RTP) is a payload format to be used for e.g. Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) encoded speech signals. RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. RTP does not address resource reservation and does not guarantee quality-of service for real-time services. The data transport is e.g. augmented by the control protocol RTCP (Real-time Transport Control Protocol) to allow monitoring of the data delivery in a manner scalable to large multicast networks, and to provide minimal control and identification functionality. RTP and RTCP are designed to be independent of the underlying transport and network layers. The protocol supports the use of RTP-level translators and mixers. The data transported by RTP in a packet, for example audio samples or compressed video data. A data packet includes e.g. the fixed RTP header, a possibly empty list of contributing sources, and the payload data.

[0034] The RTP according to the invention is the RTP currently under discussion and modifications thereof and equivalents thereof. RTP may be a protocol for both audio and video, or audio only, or video only, or audio, video and data, or audio and data, etc. One modification of RTP is e.g. RTP/I, an application level real-time protocol for distributed interactive media. Typical examples of distributed interactive media are shared whiteboards, networked computer games and distributed virtual environments. RTP/I defines a standardized framing for the transmission of data and provides mechanisms that are universally needed for this media class. Thereby RTP/I enables the development of reusable functionality and generic services that can be employed for multiple distributed interactive media. Examples for this kind of functionality are the ability to record sessions, to support late coming participants, and to provide security services. RTP/I is a protocol that follows the ideas of application level framing and integrated layer processing. It has been designed to be independent of the underlying network and transport layers. Thus RTP/I as a modified RTP protocol that reuses many aspects of RTP while it is thoroughly adapted to the specific needs of distributed interactive media.

Proxy, proxy server:

[0035] An intermediary program that acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, possibly after translation, to other servers. A proxy interprets, and, if necessary, rewrites a request message before forwarding it.

Server:

[0036] A server is an application program that accepts requests in order to service requests and sends back responses to those requests. Servers are either proxy, redirect or user agent servers or registrars.

User agent client (UAC), calling user agent:

[0037] A user agent client is a client application that initiates the SIP request.

SDP:

[0038] The Session Description Protocol (SDP) is intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation.

[0039] The purpose of SDP is to convey information about media streams in multimedia sessions to allow the recipients of a session description to participate in the session. SDP is primarily intended for use in an internetwork, although it is sufficiently general that it can describe conferences in other network environments.

[0040] A multimedia session, for these purposes, is defined as a set of media streams that exist for some duration of time. Media streams can be many-to-many. The times during which the session is active need not be continuous.

[0041] Thus far, multicast based sessions on the Internet have differed from many other forms of conferencing in that anyone receiving the traffic can join the session (unless the session traffic is encrypted). In such an environment, SDP serves two primary purposes. It is a means to communicate the existence of a session, and is a means to convey sufficient information to enable joining and participating in the session. In a unicast environment, only the latter purpose is likely to be relevant.

[0042] Thus SDP includes:

- Session name and purpose
- Time(s) the session is active
- The media comprising the session
- Information to receive those media (addresses, ports, formats and so on)

[0043] As resources necessary to participate in a session may be limited, some additional information may also be desirable:

- Information about the bandwidth to be used by the conference
- Contact information for the person responsible for the session

[0044] In general, SDP must convey sufficient information to be able to join a session (with the possible exception of encryption keys) and to announce the resources to be used to non-participants that may need to know.

[0045] SDP includes:

- The type of media (video, audio, etc)
- The transport protocol (RTP/UDP/IP, H.320, etc)
- The format of the media (H.261 video, MPEG video, etc)

[0046] For an IP multicast session, the following are also conveyed:

- Multicast address for media
- Transport Port for media

[0047] This address and port are the destination address and destination port of the multicast stream, whether being sent, received, or both.

[0048] For an IP unicast session, the following are conveyed:

- Remote address for media
- Transport port for contact address

[0049] The semantics of this address and port depend on the media and transport protocol defined. By default, this is the remote address and remote port to which data is sent, and the remote address and local port on which to receive data. However, some media may define to use these to establish a control channel for the actual media flow.

[0050] The SDP according to the invention is the SDP currently standardized and modifications thereof and equivalents thereof.

[0051] FIG. 2 shows a portion of an exemplary telecommunications network according to the teachings of the present invention.

[0052] Like in fig. 1 two IP parties, e.g. yshen@alcatel.de and eric@alcatel.com, are interconnected via two networks: a SIP signaling network and a transmission network. Via the SIP signaling network signaling is performed, e.g. a connection is established between the two IP parties. Via the transmission network the information to be transmitted, e.g. voice, data, etc. is transmitted in media streams (RTP session).

[0053] Different from fig. 1 a lawful interception device is included in fig. 2. The lawful interception device is e. g. a processor with particular software. The processor is e.g. a digital signal processor, a controller, a microprocessor or the like. Instead of one processor two or more processors could be used. Two or more processors could be located at different sites. One processor could be used to perform SIP proxy server operations and another processor could be used to perform RTP proxy server operations. In general, one, two or more hardwares could be used to run one, two, or more softwares. Each software could in addition be run in parts on different hardware.

[0054] The lawful interception device includes a SIP (Session Initiation Protocol) proxy server or a MGC (Media Gateway Controller) to detect information in the signalling information being transmitted between two IP (Internet Protocol) parties and to generate instructions out of the detected signalling information for instructing a

T-Mobile Exhibit 1002
Part 4 of 6

RTP (Real-time Transport Protocol) proxy server to create channels to bypass a media stream to be intercepted via an intermediate storage medium. Media streams are e.g. VoIP, data, internet access, e-mail, video, real-time pictures, music, video clips, video games, etc. The storage medium could be a compact disk, a magnetic storage medium, a read access memory, or the like.

[0055] The method for performing SIP signaling for a media stream includes the following steps:

receiving a SIP invite message of a first IP party,

adapting at least one connection parameter in the SDP (Session Description Protocol) of the received SIP invite message,

transmitting the adapted SIP invite message to a second IP party,

receiving a SIP response message of the seond IP party,

adapting at least one connection parameter in the SDP (Session Description Protocol) of the received SIP response message,

transmitting the adapted SIP response message to the first IP party.

[0056] At least one RTP parameter includes information about a bypass channel, an address, or a port. The RTP parameters sent to both IP parties differ from each other.

[0057] After receipt of the SIP invite message of the first IP party the SIP interception proxy server sends a request to the RTP interception proxy server to assign at least two channels for bothway communication. The interface used to communicate between SIP interception proxy server and RTP interception proxy server is a XML based API. The number of channels to be assigned may vary dependent of the amount of data to be transmitted, of the bandwith requested, of the quality of service requested, of the kind of information to be transmitted, e.g. voice, voice and data, voice and video, etc. At least one channel is assigned to transmit information between the RTP interception proxy server and the terminal of the first IP party. The terminal could be a phone, a laptop, a personal computer, a screenphone, a mobile phone, etc. At least one other channel is assigned to transmit information between the RTP interception proxy server and the terminal of the second IP party.

[0058] Assume channel A at the RTP interception proxy server is assigned to transmit information between the second IP terminal and the terminal of the first IP party, and channel B is assigned to transmit information between the the terminal of the first IP party and the second IP terminal. Then the RTP interception proxy server sends information about the assignment of channels A and B to the SIP interception proxy server. The SIP interception proxy server includes information about channel A in the invite message to be send to the second IP party. The information about channel A is advantageously included in the connection parameter information to be included in the SDP of the SIP invite message.

[0059] After receipt of the SIP response message of the seond IP party, which corresponds to an ok message stating that a connection to the first IP party is desired, the SIP interception proxy server exchanges the connection parameter included in the SDP part of the ok message by the information about channel B. The modified ok message including the information about channel B is send to the first IP party.

[0060] Thus the first IP party will send data to channel B and receive data via channel A of the RTP interception proxy server. The second IP party will send data to channel A and receive data via channel B of the RTP interception proxy server. Within the lawful interception device the intermediate storage medium is connected to both channel A and B. Thus the information flow between both IP parties will transfer the intermediate storage medium and thus interception is enabled. The first party is not aware on which channel the second party is sending, and the second party is not aware on which channel the first party is sending. Thus interception is transparent regarding the two IP parties.

[0061] A computer program for performing at least part of the steps of the inventive method could be used as an upgrade software, which is sold e.g. to service providers, which will upgrade one or more SIP proxy server thus enabling a usual SIP proxy server having the functionality of an SIP interception proxy server. The computer program includes at least the following steps:

adapting at least one connection parameter in the SDP (Session Description Protocol) of the received SIP invite message,

adapting at least one connection parameter in the SDP (Session Description Protocol) of the received SIP response message.

[0062] The computer program could also be programmed to perform all steps of the method as described above.

[0063] Within an IP network one, two, or more SIP proxy servers could be used, one, two, or more SIP interception proxy servers could be used, one, two, or more RTP proxy servers could be used, and one, two, or more RTP interception proxy servers could be used.

[0064] The IP network could be a wireline network, a wireless network, or a combination of both.

List of abbreviations:

[0065]

3G	Third Generation
API	Application Programmer Interface
AMR	Adaptive Multi-Rate
AMR-WB	AMR-Wideband
DTMF	Dual-Tone Multi-Frequency
FEC	Forward Error Correction
H248	ITU standard
H261	ITU standard
H320	ITU standard
H323	ITU standard
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISDN	Integrated Services Digital Network
LDAP	Lightweight Directory Access Protocol
MEGACO	Media Gateway Controller
MCU	Multipoint Control Unit
MPEG	Motion Picture Expert Group
MGC	Media Gateway Controller
NGN	Next Generation Network
PSTN	Public Switched Telephone Network
PLMN	Public Land Mobile Network
QoS	Quality of Service
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
TCP	Transmission Control Protocol
UAC	User Agent Client
UDP	User Datagram Protocol
UMTS	Universal Mobile Transmission System
VoIP	Voice over IP
XML	extensible Markup Language

Claims

1. Lawful interception device including a Session Initiation Protocol proxy server or a Media Gateway Controller adapted to detect information in the signalling information being transmitted between two Internet Protocol parties and **characterised by** being adapted to generate instructions out of the detected signalling information for instructing a Real-time Transport Protocol proxy server to create channels to bypass a media stream to be intercepted via an intermediate storage medium.

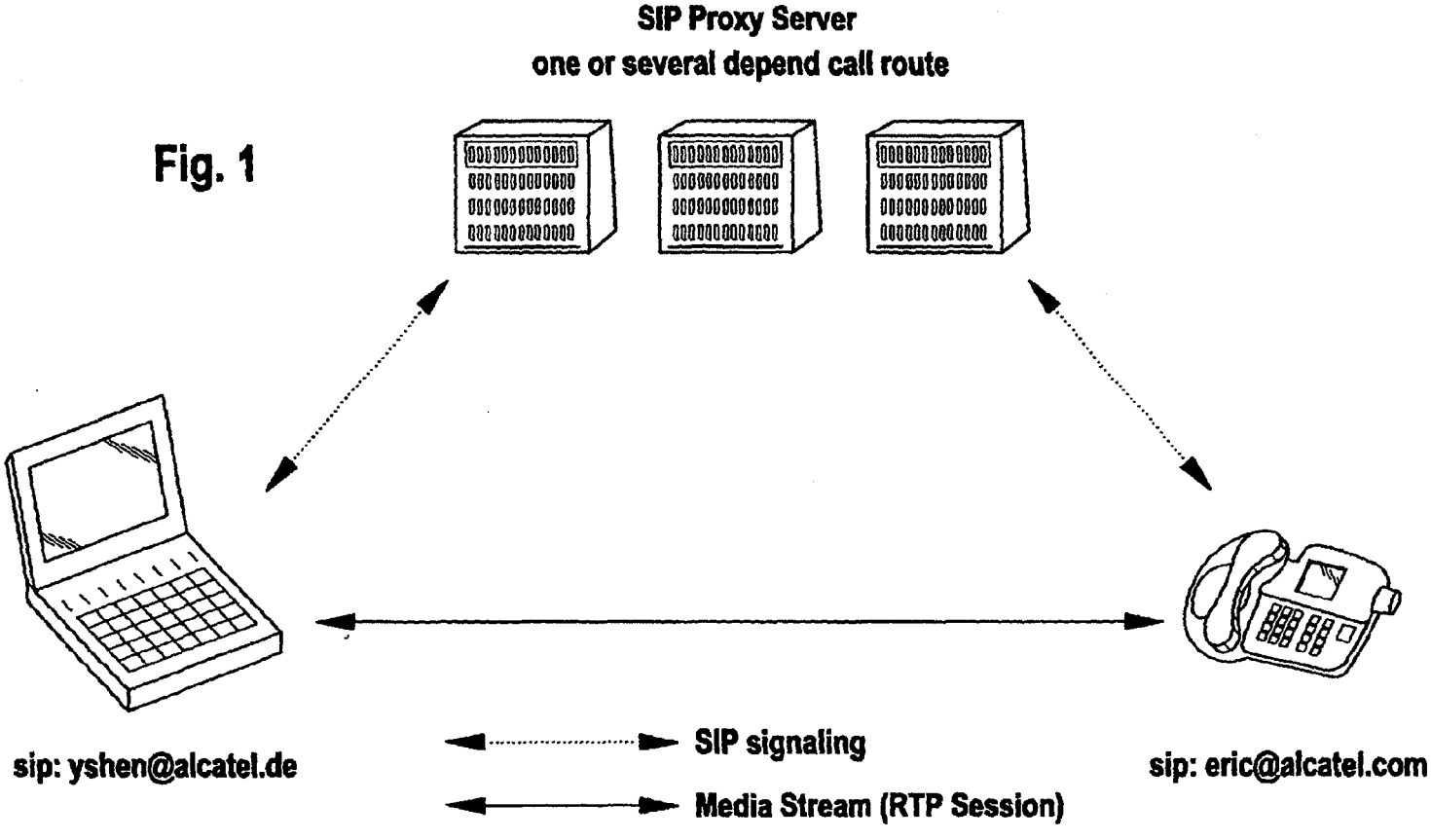
Patentansprüche

1. Legale Abfangeinrichtung mit einem Session-Initiation-Protocol-Proxy-Server oder einer Medien-Gateway-Steuerung, der bzw. die geeignet ist, Informationen in der zwischen zwei Internet-Protokoll-Teilnehmern übertragenen Zeichengabenachricht zu erfassen, und
dadurch gekennzeichnet ist, dass er bzw. sie geeignet ist, aus der erfassten Zeichengabeinformation Anweisungen für einen Echtzeit-Transportprotokoll-Proxy-Server zu erzeugen, Kanäle für die Umleitung eines abzufangenden Medienstroms über ein Zwischenspeichermedium einzurichten.

Revendications

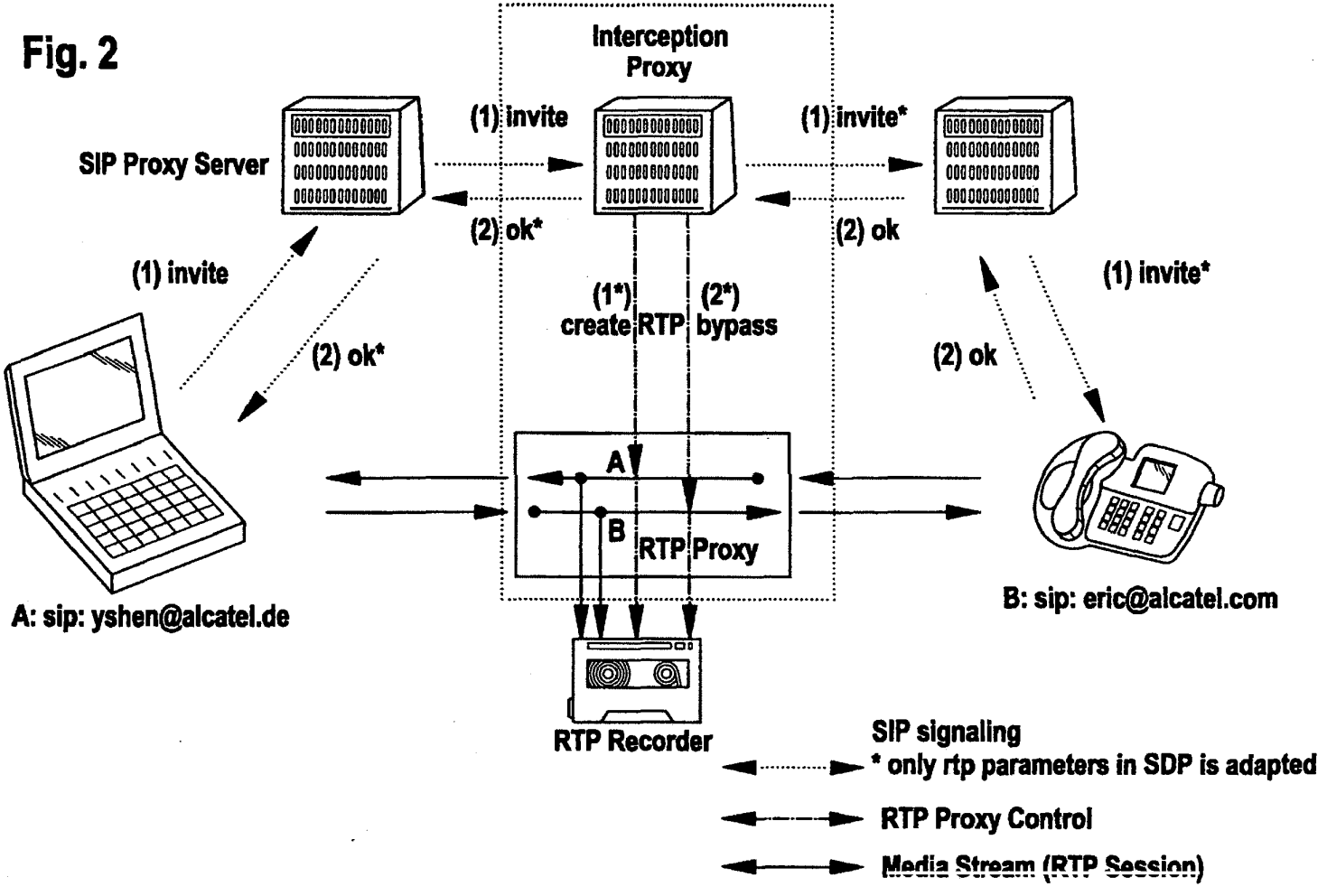
1. Dispositif d'interception licite comprenant un serveur mandataire de protocole de lancement de session ou un contrôleur de passerelle de média adaptés pour détecter des informations dans les informations de signalisation étant transmises entre deux correspondants IP (protocole internet) et **caractérisé en ce qu'**il est adapté pour générer des instructions parmi les informations de signalisation de signalisation détectées pour donner pour instruction à un serveur mandataire de protocole de transport en temps réel de créer des voies pour contourner un train de données de média à intercepter, via un support de stockage intermédiaire.

Fig. 1



T-Mobile Exhibit 1002
Part 4 of 6

Fig. 2


Interception Proxy
SIP Proxy Server
(1) invite
(2) ok*
(1) invite*
(2) ok
(1) invite
(2) ok*
(1*) create RTP
(2*) bypass
(2) ok
(1) invite*
A
B
RTP Proxy
A: sip: yshen@alcatel.de
B: sip: eric@alcatel.com
RTP Recorder
SIP signaling
* only rtp parameters in SDP is adapted
RTP Proxy Control
Media Stream (RTP Session)

PATENT COOPERATION TREATY

PCT

INTERNATIONAL PRELIMINARY REPORT ON PATENTABILITY
(Chapter I of the Patent Cooperation Treaty)

(PCT Rule 44*bis*)

Applicant's or agent's file reference 83636-18	**FOR FURTHER ACTION**	See item 4 below
International application No. PCT/CA2008/000545	International filing date *(day/month/year)* 20 March 2008 (20.03.2008)	Priority date *(day/month/year)* 26 March 2007 (26.03.2007)
International Patent Classification (8th edition unless older edition indicated) See relevant information in Form PCT/ISA/237		
Applicant DIGIFONICA (INTERNATIONAL) LIMITED		

1. This international preliminary report on patentability (Chapter I) is issued by the International Bureau on behalf of the International Searching Authority under Rule 44 *bis*.1(a).

2. This REPORT consists of a total of 6 sheets, including this cover sheet.

   In the attached sheets, any reference to the written opinion of the International Searching Authority should be read as a reference to the international preliminary report on patentability (Chapter I) instead.

3. This report contains indications relating to the following items:

☒	Box No. I	Basis of the report
☐	Box No. II	Priority
☐	Box No. III	Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
☐	Box No. IV	Lack of unity of invention
☒	Box No. V	Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
☐	Box No. VI	Certain documents cited
☒	Box No. VII	Certain defects in the international application
☒	Box No. VIII	Certain observations on the international application

4. The International Bureau will communicate this report to designated Offices in accordance with Rules 44*bis*.3(c) and 93*bis*.1 but not, except where the applicant makes an express request under Article 23(2), before the expiration of 30 months from the priority date (Rule 44*bis* .2).

	Date of issuance of this report 29 September 2009 (29.09.2009)
The International Bureau of WIPO 34, chemin des Colombettes 1211 Geneva 20, Switzerland Facsimile No. +41 22 338 82 70	Authorized officer Athina Nickitas-Etienne e-mail: pt04.pct@wipo.int

Form PCT/IB/373 (January 2004)

T-Mobile Exhibit 1002
Part 4 of 6

PATENT COOPERATION TREATY

From the
INTERNATIONAL SEARCHING AUTHORITY

To:
SMART & BIGGAR
Box 11560 Vancouver Centre
2200 - 650 W. Georgia Street
VANCOUVER, British Columbia
Canada, V6B 4N8

PCT

WRITTEN OPINION OF THE
INTERNATIONAL SEARCHING AUTHORITY

(PCT Rule 43*bis*.1)

Date of mailing *(day/month/year)* 20 June 2008 (20-06-2008)

Applicant's or agent's file reference
83636-18

FOR FURTHER ACTION
See paragraph 2 below

International application No.	International filing date *(day/month/year)*	Priority date *(day/month/year)*
PCT/CA2008/000545	20 March 2008 (20-03-2008)	26 March 2007 (26-03-2007)

International Patent Classification (IPC) or both national classification and IPC
IPC: ***H04L 12/66*** (2006.01) , ***H04M 11/06*** (2006.01) , ***H04M 3/42*** (2006.01) , ***H04Q 3/00*** (2006.01) , ***H04Q 3/64*** (2006.01)

Applicant
DIGIFONICA (INTERNATIONAL) LIMITED ET AL

1. This opinion contains indications relating to the following items :

- [X] Box No. I Basis of the opinion
- [ ] Box No. II Priority
- [ ] Box No. III Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
- [ ] Box No. IV Lack of unity of invention
- [X] Box No. V Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
- [ ] Box No. VI Certain documents cited
- [X] Box No. VII Certain defects in the international application
- [X] Box No. VIII Certain observations on the international application

2. **FURTHER ACTION**
If a demand for international preliminary examination is made, this opinion will be considered to be a written opinion of the International Preliminary Examining Authority ("IPEA") except that this does not apply where the applicant chooses an Authority other than this one to be the IPEA and the chosen IPEA has notified the International Bureau under Rule 66.1*bis*(b) that written opinions of this International Searching Authority will not be so considered.

If this opinion is, as provided above, considered to be a written opinion of the IPEA, the applicant is invited to submit to the IPEA a written reply together, where appropriate, with amendments, before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later.

For further options, see Form PCT/ISA/220.

3. For further details, see notes to Form PCT/ISA/220.

Name and mailing address of the ISA/CA	Date of completion of this opinion	Authorized officer
Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	13 June 2008 (13-06-2008)	Arthur Smith 819-953-1360

Box No. I Basis of this opinion

1. With regard to the **language**, this opinion has been established on the basis of:

   [X] the international application in the language in which it was filed

   [ ] a translation of the international application into , which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b)).

2. [ ] This opinion has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43*bis*.1(a))

3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application and necessary to the claimed invention, this opinion has been established on the basis of :

   a. type of material

      [ ] a sequence listing

      [ ] table(s) related to the sequence listing

   b. format of material

      [ ] on paper

      [ ] in electronic form

   c. time of filing/furnishing

      [ ] contained in the international application as filed.

      [ ] filed together with the international application in electronic form

      [ ] furnished subsequently to this Authority for the purposes of search.

4. [ ] In addition, in the case that more than one version or copy of a sequence listing and/or table(s) relating thereto has been filed or furnished, the required statements that the information in the subsequent or additional copies is identical to that in the application as filed or does not go beyond the application as filed, as appropriate, were furnished.

5. Additional comments :

Box No. V **Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement**

1. Statement

Novelty (N)	Claims	1-43	YES
	Claims	None	NO
Inventive step (IS)	Claims	1-43	YES
	Claims	None	NO
Industrial applicability (IA)	Claims	1-43	YES
	Claims	None	NO

2. Citations and explanations :

The following document is referred to in this communication:
D1 US2005/0083911 A1

D1 is considered to form the closest prior art. D1 describes providing E911 emergency services to an IP telephony-based PBX or similar system by establishing a 911 Location Server Database comprising an Emergency Response Location (ERL) database and a Phone Location database. The location of a caller stored in the ERL database is used to route an emergency call to an end-office switch corresponding to the nearest Public Safety Answering Point (PSAP). The Phone Location database stores one record for each registered phone in the system.

Novelty
D1 fails to disclose all the elements of independent claims 1, 15, 29, and 43. In particular, D1 fails to teach a direct inward dialing (DID) identifier or producing a DID identifier for a caller identifier that has no pre-associated DID identifier. Claims 2-14, 16-28 and 30-42 depend on the above independent claims, respectively.

Thus, claims 1-43 **are considered to be novel** in accordance with Article 33(2) PCT.

Inventive Step
D1 fails to teach or suggest producing a routing message including an emergency response center identifier and temporary DID identifier for establishing a route between a caller and an emergency response center as claimed in independent claims 1, 15, 29, and 43. Claims 2-14, 16-28, and 30-42 depend on the above independent claims, respectively.

Thus, claims 1-43 **are considered to have inventive step** in accordance with Article 33(3) PCT.

Industrial Applicability
Including location information for emergency VoIP callers located behind a VoIP PBX finds application in IP telephony networks. Thus, claims 1-43 **are industrially applicable** in accordance with Article 33(4) PCT.

Box No. VII Certain defects in the international application

The following defects in the form or contents of the international application have been noted :

Description-Related Deficiencies

On page 1, line 19, of the description, the term "presended" (sic) is misspelled. Applicant may have intended "presented to".

Box No. VIII Certain observations on the international application

The following observations on the clarity of the claims, description, and drawings or on the question whether the claims are fully supported by the description, are made :

Claims-Related Deficiencies

Claims 1 and 22 do not comply with PCT Article 6. In claim 1, the term "the method" (line 3) lacks an antecedent and in claim 22, the term "said pool" (line 15) lacks an antecedent.

PATENT COOPERATION TREATY

PCT

INTERNATIONAL PRELIMINARY REPORT ON PATENTABILITY
(Chapter I of the Patent Cooperation Treaty)

(PCT Rule 44*bis*)

Applicant's or agent's file reference 83636-56	**FOR FURTHER ACTION**	See item 4 below
International application No. PCT/CA2009/001317	International filing date *(day/month/year)* 17 September 2009 (17.09.2009)	Priority date *(day/month/year)*
International Patent Classification (8th edition unless older edition indicated) See relevant information in Form PCT/ISA/237		
Applicant DIGIFONICA (INTERNATIONAL) LIMITED		

1. This international preliminary report on patentability (Chapter I) is issued by the International Bureau on behalf of the International Searching Authority under Rule 44 *bis*.1(a).

2. This REPORT consists of a total of 5 sheets, including this cover sheet.

   In the attached sheets, any reference to the written opinion of the International Searching Authority should be read as a reference to the international preliminary report on patentability (Chapter I) instead.

3. This report contains indications relating to the following items:

☒	Box No. I	Basis of the report
☐	Box No. II	Priority
☐	Box No. III	Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
☐	Box No. IV	Lack of unity of invention
☒	Box No. V	Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
☐	Box No. VI	Certain documents cited
☐	Box No. VII	Certain defects in the international application
☐	Box No. VIII	Certain observations on the international application

4. The International Bureau will communicate this report to designated Offices in accordance with Rules 44*bis*.3(c) and 93*bis*.1 but not, except where the applicant makes an express request under Article 23(2), before the expiration of 30 months from the priority date (Rule 44*bis* .2).

	Date of issuance of this report 20 March 2012 (20.03.2012)
The International Bureau of WIPO 34, chemin des Colombettes 1211 Geneva 20, Switzerland Facsimile No. +41 22 338 82 70	Authorized officer Athina Nickitas-Etienne e-mail: pt04.pct@wipo.int

Form PCT/IB/373 (January 2004)

T-Mobile Exhibit 1002
Part 4 of 6

PATENT COOPERATION TREATY

From the
INTERNATIONAL SEARCHING AUTHORITY

To:
SMART & BIGGAR
Box 11560 Vancouver Centre
2200 - 650 W. Georgia Street
VANCOUVER, British Columbia
Canada, V6B 4N8

PCT

WRITTEN OPINION OF THE
INTERNATIONAL SEARCHING AUTHORITY

(PCT Rule 43*bis*.1)

Date of mailing *(day/month/year)* 18 June 2010 (18-06-2010)

Applicant's or agent's file reference
83636-56

FOR FURTHER ACTION
See paragraph 2 below

International application No.	International filing date *(day/month/year)*	Priority date *(day/month/year)*
PCT/CA2009/001317	17 September 2009 (17-09-2009)	

International Patent Classification (IPC) or both national classification and IPC
IPC: ***H04L 12/66*** (2006.01) , ***H04L 29/06*** (2006.01) , ***H04W 36/02*** (2009.01) , ***H04W 36/18*** (2009.01)

Applicant
DIGIFONICA (INTERNATIONAL) LIMITED ET AL

1. This opinion contains indications relating to the following items :

- [X] Box No. I Basis of the opinion
- [ ] Box No. II Priority
- [ ] Box No. III Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
- [ ] Box No. IV Lack of unity of invention
- [X] Box No. V Reasoned statement under Rule 43*bis*.1(a)(I) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
- [ ] Box No. VI Certain documents cited
- [ ] Box No. VII Certain defects in the international application
- [ ] Box No. VIII Certain observations on the international application

2. **FURTHER ACTION**
If a demand for international preliminary examination is made, this opinion will be considered to be a written opinion of the International Preliminary Examining Authority ("IPEA") except that this does not apply where the applicant chooses an Authority other than this one to be the IPEA and the chosen IPEA has notified the International Bureau under Rule 66.1*bis*(b) that written opinions of this International Searching Authority will not be so considered.

If this opinion is, as provided above, considered to be a written opinion of the IPEA, the applicant is invited to submit to the IPEA a written reply together, where appropriate, with amendments, before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later.

For further options, see Form PCT/ISA/220.

3. For further details, see notes to Form PCT/ISA/220.

Name and mailing address of the ISA/CA	Date of completion of this opinion	Authorized officer
Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	17 June 2010 (17-06-2010)	Salvatore Ginese (819) 934-4888

Box No. I Basis of this opinion

1. With regard to the **language**, this opinion has been established on the basis of:

   [X] the international application in the language in which it was filed

   [ ] a translation of the international application into , which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b)).

2. [ ] This opinion has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43*bis*.1(a))

3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, this opinion has been established on the basis of a sequence listing filed or furnished:

   a. (means)

      [ ] on paper

      [ ] in electronic form

   b. (time)

      [ ] the international application as filed.

      [ ] together with the international application in electronic form

      [ ] subsequently to this Authority for the purposes of search

4. [ ] In addition, in the case that more than one version or copy of a sequence listing has been filed or furnished, the required statements that the information in the subsequent or additional copies is identical to that in the application as filed or does not go beyond the application as filed, as appropriate, were furnished.

5. Additional comments :

Box No. V **Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement**

1. Statement

Novelty (N)	Claims	1 to 16	YES
	Claims	None	NO
Inventive step (IS)	Claims	1 to 16	YES
	Claims	None	NO
Industrial applicability (IA)	Claims	1 to 16	YES
	Claims	None	NO

2. Citations and explanations :

Reference is made to the following documents:

D1: US 2009/0028146 A1	**KLEYMAN et al.**	**29 January 2009**	**(29-01-2009)**
D2: US 7454510 B2	**KLEYMAN et al.**	**18 November 2008**	**(18-11-2008)**
D3: US 2007/0253418 A1	**SHIRI et al.**	**01 November 2007**	**(01-11-2007)**
D4: US 2007/0036143 A1	**ALT et al.**	**15 February 2007**	**(15-02-2007)**

Novelty

None of D1 to D4 individually teaches explicitly a method for facilitating uninterrrupted transmission of internet protocol (IP) transmissions containing real time transport protocol (RTP) data during endpoint changes, the method comprising: maintaining records, each record associating session information, caller information and callee information for IP communication sessions; said session information including caller and callee RTP port identifiers identifying caller and callee RTP ports respectively of a media relay; said caller information including a caller IP address identifier and a caller port identifier to which IP transmission received at said callee RTP port are transmitted form the media relay, and a caller synchronization source (SSRC) identifier; and said callee information including a callee IP address identifier and a callee port identifier to which IP transmissions received at said caller RTP port are transmitted from the media relay, and a callee SSRC identifier; and when an IP transmission is received at said caller RTP port or said callee RTP port; locating one of said records having said caller RTP port identifier or said callee RTP port identifier matching a destination port identifier in said IP transmission; when said one of said records is located and when said destination port identifier in said IP transmission matches the (caller or callee) RTP port identifier of said one of said records; setting a source IP address identifier and source port identifier from said IP transmission as the (caller or callee) IP address identifier and (caller/callee) IP address identifier and(caller/callee) port identifier respectively of said one of said records when said (caller/callee) IP address identifier and (caller/callee) port identifier do not match said source IP address identifier and source port identifier respectively; and a received SSRC identifier in said IP transmission matches said (caller/callee) SSRC identifier.

The subject matter of claims 1 to 16 is therefore considered to be novel, satisfying the requirements of **Article 33(2).**

(Continued in supplemental box 1 of 1)

Supplemental Box

In case **the space in any of the preceding boxes is not sufficient.**

Continuation of : Box No. V (This is supplemental box 1 of 1)

Inventive step

None of D1 to D4 teaches or suggests in combination the aforementioned features.

The subject matter of claims 1 to 16 is therefore considered to contain an inventive step, satisfying the requirements of **Article 33(3).**

Industrial applicability

The subject matter of claims 1 to 16 is considered to be industrially applicable, thus fulfilling the requirements of **Article 33(4)**.

Network Working Group
Request for Comments: 2661
Category: Standards Track

W. Townsley
A. Valencia
cisco Systems
A. Rubens
Ascend Communications
C. Pall
G. Zorn
Microsoft Corporation
B. Palter
Redback Networks

August 1999

Layer Two Tunneling Protocol "L2TP"

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract

This document describes the Layer Two Tunneling Protocol (L2TP). STD 51, RFC 1661 specifies multi-protocol access via PPP [RFC1661]. L2TP facilitates the tunneling of PPP packets across an intervening network in a way that is as transparent as possible to both end-users and applications.

Table of Contents

1.0 Introduction..... 3
1.1 Specification of Requirements..... 4
1.2 Terminology..... 4
2.0 Topology..... 8
3.0 Protocol Overview..... 9
3.1 L2TP Header Format..... 9
3.2 Control Message Types..... 11
4.0 Control Message Attribute Value Pairs..... 12
4.1 AVP Format..... 13
4.2 Mandatory AVPs..... 14
4.3 Hiding of AVP Attribute Values..... 14

4.4 AVP Summary..... 17
4.4.1 AVPs Applicable To All Control Messages..... 17

4.4.2 Result and Error Codes..... 18
4.4.3 Control Connection Management AVPs..... 20
4.4.4 Call Management AVPs..... 27
4.4.5 Proxy LCP and Authentication AVPs..... 34
4.4.6 Call Status AVPs..... 39
5.0 Protocol Operation..... 41
5.1 Control Connection Establishment..... 41
5.1.1 Tunnel Authentication..... 42
5.2 Session Establishment..... 42
5.2.1 Incoming Call Establishment..... 42
5.2.2 Outgoing Call Establishment..... 43
5.3 Forwarding PPP Frames..... 43
5.4 Using Sequence Numbers on the Data Channel..... 44
5.5 Keepalive (Hello)..... 44
5.6 Session Teardown..... 45
5.7 Control Connection Teardown..... 45
5.8 Reliable Delivery of Control Messages..... 46
6.0 Control Connection Protocol Specification..... 48
6.1 Start-Control-Connection-Request (SCCRQ)..... 48
6.2 Start-Control-Connection-Reply (SCCRP)..... 48
6.3 Start-Control-Connection-Connected (SCCCN)..... 49
6.4 Stop-Control-Connection-Notification (StopCCN)..... 49
6.5 Hello (HELLO)..... 49
6.6 Incoming-Call-Request (ICRQ)..... 50
6.7 Incoming-Call-Reply (ICRP)..... 51
6.8 Incoming-Call-Connected (ICCN)..... 51
6.9 Outgoing-Call-Request (OCRQ)..... 52
6.10 Outgoing-Call-Reply (OCRP)..... 53
6.11 Outgoing-Call-Connected (OCCN)..... 53
6.12 Call-Disconnect-Notify (CDN)..... 53
6.13 WAN-Error-Notify (WEN)..... 54
6.14 Set-Link-Info (SLI)..... 54
7.0 Control Connection State Machines..... 54
7.1 Control Connection Protocol Operation..... 55
7.2 Control Connection States..... 56
7.2.1 Control Connection Establishment..... 56
7.3 Timing considerations..... 58
7.4 Incoming calls..... 58
7.4.1 LAC Incoming Call States..... 60
7.4.2 LNS Incoming Call States..... 62
7.5 Outgoing calls..... 63
7.5.1 LAC Outgoing Call States..... 64
7.5.2 LNS Outgoing Call States..... 66
7.6 Tunnel Disconnection..... 67
8.0 L2TP Over Specific Media..... 67
8.1 L2TP over UDP/IP..... 68

8.2 IP..... 69
9.0 Security Considerations..... 69
9.1 Tunnel Endpoint Security..... 70
9.2 Packet Level Security..... 70
9.3 End to End Security..... 70
9.4 L2TP and IPsec..... 71
9.5 Proxy PPP Authentication..... 71
10.0 IANA Considerations..... 71
10.1 AVP Attributes..... 71
10.2 Message Type AVP Values..... 72
10.3 Result Code AVP Values..... 72

10.3.1 Result Code Field Values..... 72
10.3.2 Error Code Field Values..... 72
10.4 Framing Capabilities & Bearer Capabilities..... 72
10.5 Proxy Authen Type AVP Values..... 72
10.6 AVP Header Bits..... 73
11.0 References..... 73
12.0 Acknowledgments..... 74
13.0 Authors' Addresses..... 75
Appendix A: Control Channel Slow Start and Congestion Avoidance..... 76
Appendix B: Control Message Examples..... 77
Appendix C: Intellectual Property Notice..... 79
Full Copyright Statement..... 80

1.0 Introduction

PPP [RFC1661] defines an encapsulation mechanism for transporting multiprotocol packets across layer 2 (L2) point-to-point links. Typically, a user obtains a L2 connection to a Network Access Server (NAS) using one of a number of techniques (e.g., dialup POTS, ISDN, ADSL, etc.) and then runs PPP over that connection. In such a configuration, the L2 termination point and PPP session endpoint reside on the same physical device (i.e., the NAS).

L2TP extends the PPP model by allowing the L2 and PPP endpoints to reside on different devices interconnected by a packet-switched network. With L2TP, a user has an L2 connection to an access concentrator (e.g., modem bank, ADSL DSLAM, etc.), and the concentrator then tunnels individual PPP frames to the NAS. This allows the actual processing of PPP packets to be divorced from the termination of the L2 circuit.

One obvious benefit of such a separation is that instead of requiring the L2 connection terminate at the NAS (which may require a long-distance toll charge), the connection may terminate at a (local) circuit concentrator, which then extends the logical PPP session over

a shared infrastructure such as frame relay circuit or the Internet. From the user's perspective, there is no functional difference between having the L2 circuit terminate in a NAS directly or using L2TP.

L2TP may also solve the multilink hunt-group splitting problem. Multilink PPP [RFC1990] requires that all channels composing a multilink bundle be grouped at a single Network Access Server (NAS). Due to its ability to project a PPP session to a location other than the point at which it was physically received, L2TP can be used to make all channels terminate at a single NAS. This allows multilink operation even when the calls are spread across distinct physical NASs.

This document defines the necessary control protocol for on-demand creation of tunnels between two nodes and the accompanying encapsulation for multiplexing multiple, tunneled PPP sessions.

1.1 Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",

"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2 Terminology

Analog Channel

A circuit-switched communication path which is intended to carry 3.1 kHz audio in each direction.

Attribute Value Pair (AVP)

The variable length concatenation of a unique Attribute (represented by an integer) and a Value containing the actual value identified by the attribute. Multiple AVPs make up Control Messages which are used in the establishment, maintenance, and teardown of tunnels.

Call

A connection (or attempted connection) between a Remote System and LAC. For example, a telephone call through the PSTN. A Call (Incoming or Outgoing) which is successfully established between a Remote System and LAC results in a corresponding L2TP Session within a previously established Tunnel between the LAC and LNS. (See also: Session, Incoming Call, Outgoing Call).

Called Number

An indication to the receiver of a call as to what telephone number the caller used to reach it.

Calling Number

An indication to the receiver of a call as to the telephone number of the caller.

CHAP

Challenge Handshake Authentication Protocol [RFC1994], a PPP cryptographic challenge/response authentication protocol in which the cleartext password is not passed over the line.

Control Connection

A control connection operates in-band over a tunnel to control the establishment, release, and maintenance of sessions and of the tunnel itself.

Control Messages

Control messages are exchanged between LAC and LNS pairs, operating in-band within the tunnel protocol. Control messages govern aspects of the tunnel and sessions within the tunnel.

Digital Channel

A circuit-switched communication path which is intended to carry digital information in each direction.

DSLAM

Digital Subscriber Line (DSL) Access Module. A network device used in the deployment of DSL service. This is typically a concentrator of individual DSL lines located in a central office (CO) or local exchange.

Incoming Call

A Call received at an LAC to be tunneled to an LNS (see Call, Outgoing Call).

L2TP Access Concentrator (LAC)

A node that acts as one side of an L2TP tunnel endpoint and is a peer to the L2TP Network Server (LNS). The LAC sits between an LNS and a remote system and forwards packets to and from each. Packets sent from the LAC to the LNS requires tunneling with the L2TP protocol as defined in this document. The connection from the LAC to the remote system is either local (see: Client LAC) or a PPP link.

L2TP Network Server (LNS)

A node that acts as one side of an L2TP tunnel endpoint and is a peer to the L2TP Access Concentrator (LAC). The LNS is the logical termination point of a PPP session that is being tunneled from the remote system by the LAC.

Management Domain (MD)

A network or networks under the control of a single administration, policy or system. For example, an LNS's Management Domain might be the corporate network it serves. An LAC's Management Domain might be the Internet Service Provider that owns and manages it.

Network Access Server (NAS)

A device providing local network access to users across a remote access network such as the PSTN. An NAS may also serve as an LAC, LNS or both.

Outgoing Call

A Call placed by an LAC on behalf of an LNS (see Call, Incoming Call).

Peer

When used in context with L2TP, peer refers to either the LAC or LNS. An LAC's Peer is an LNS and vice versa. When used in context with PPP, a peer is either side of the PPP connection.

POTS

Plain Old Telephone Service.

Remote System

An end-system or router attached to a remote access network (i.e. a PSTN), which is either the initiator or recipient of a call. Also referred to as a dial-up or virtual dial-up client.

Session

L2TP is connection-oriented. The LNS and LAC maintain state for each Call that is initiated or answered by an LAC. An L2TP Session is created between the LAC and LNS when an end-to-end PPP connection is established between a Remote System and the LNS. Datagrams related to the PPP connection are sent over the Tunnel between the LAC and LNS. There is a one to one relationship between established L2TP Sessions and their associated Calls. (See also: Call).

Tunnel

A Tunnel exists between a LAC-LNS pair. The Tunnel consists of a Control Connection and zero or more L2TP Sessions. The Tunnel carries encapsulated PPP datagrams and Control Messages between the LAC and the LNS.

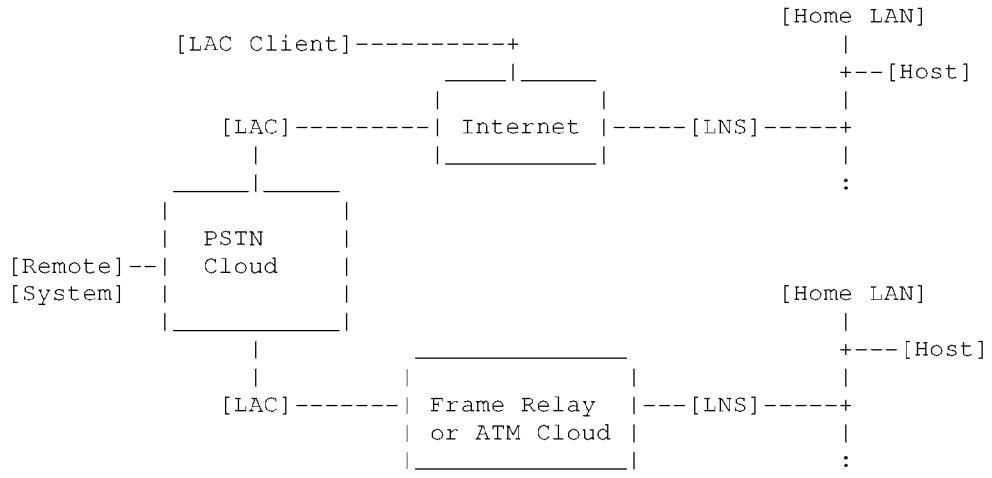
Zero-Length Body (ZLB) Message

A control packet with only an L2TP header. ZLB messages are used for explicitly acknowledging packets on the reliable control channel.

http://www.ietf.org/rfc/rfc2661.txt

2.0 Topology

The following diagram depicts a typical L2TP scenario. The goal is to tunnel PPP frames between the Remote System or LAC Client and an LNS located at a Home LAN.



The Remote System initiates a PPP connection across the PSTN Cloud to an LAC. The LAC then tunnels the PPP connection across the Internet, Frame Relay, or ATM Cloud to an LNS whereby access to a Home LAN is obtained. The Remote System is provided addresses from the HOME LAN

via PPP NCP negotiation. Authentication, Authorization and Accounting may be provided by the Home LAN's Management Domain as if the user were connected to a Network Access Server directly.

A LAC Client (a Host which runs L2TP natively) may also participate in tunneling to the Home LAN without use of a separate LAC. In this case, the Host containing the LAC Client software already has a connection to the public Internet. A "virtual" PPP connection is then created and the local L2TP LAC Client software creates a tunnel to the LNS. As in the above case, Addressing, Authentication, Authorization and Accounting will be provided by the Home LAN's Management Domain.

3.0 Protocol Overview

L2TP utilizes two types of messages, control messages and data messages. Control messages are used in the establishment, maintenance and clearing of tunnels and calls. Data messages are used to encapsulate PPP frames being carried over the tunnel. Control messages utilize a reliable Control Channel within L2TP to guarantee delivery (see section 5.1 for details). Data messages are not retransmitted when packet loss occurs.

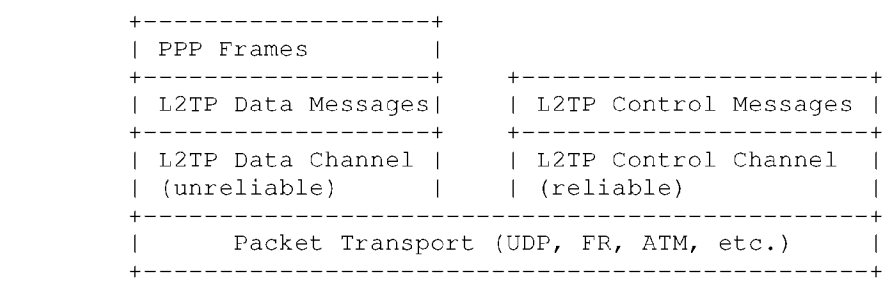


Figure 3.0 L2TP Protocol Structure

Figure 3.0 depicts the relationship of PPP frames and Control Messages over the L2TP Control and Data Channels. PPP Frames are passed over an unreliable Data Channel encapsulated first by an L2TP header and then a Packet Transport such as UDP, Frame Relay, ATM, etc. Control messages are sent over a reliable L2TP Control Channel which transmits packets in-band over the same Packet Transport.

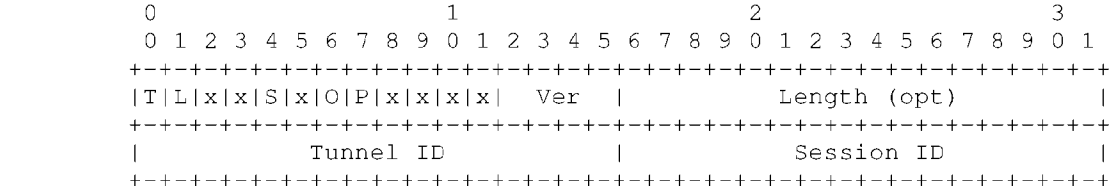
Sequence numbers are required to be present in all control messages and are used to provide reliable delivery on the Control Channel. Data Messages may use sequence numbers to reorder packets and detect lost packets.

All values are placed into their respective fields and sent in network order (high order octets first).

3.1 L2TP Header Format

L2TP packets for the control channel and data channel share a common header format. In each case where a field is optional, its space does not exist in the message if the field is marked not present. Note that while optional on data messages, the Length, Ns, and Nr fields marked as optional below, are required to be present on all control messages.

This header is formatted:



Page 1282
T-Mobile Exhibit 1002
Part 4 of 6

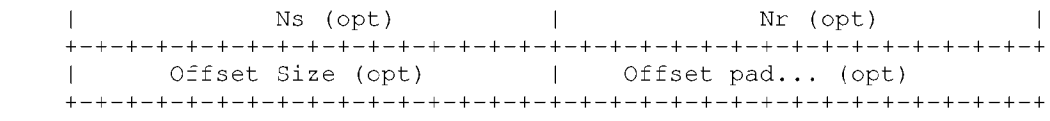


Figure 3.1 L2TP Message Header

The Type (T) bit indicates the type of message. It is set to 0 for a data message and 1 for a control message.

If the Length (L) bit is 1, the Length field is present. This bit MUST be set to 1 for control messages.

The x bits are reserved for future extensions. All reserved bits MUST be set to 0 on outgoing messages and ignored on incoming messages.

If the Sequence (S) bit is set to 1 the Ns and Nr fields are present. The S bit MUST be set to 1 for control messages.

If the Offset (O) bit is 1, the Offset Size field is present. The O bit MUST be set to 0 (zero) for control messages.

If the Priority (P) bit is 1, this data message should receive preferential treatment in its local queuing and transmission. LCP echo requests used as a keepalive for the link, for instance, should generally be sent with this bit set to 1. Without it, a temporary interval of local congestion could result in interference with keepalive messages and unnecessary loss of the link. This feature is only for use with data messages. The P bit MUST be set to 0 for all control messages.

Ver MUST be 2, indicating the version of the L2TP data message header described in this document. The value 1 is reserved to permit detection of L2F [RFC2341] packets should they arrive intermixed with L2TP packets. Packets received with an unknown Ver field MUST be discarded.

The Length field indicates the total length of the message in octets.

Tunnel ID indicates the identifier for the control connection. L2TP tunnels are named by identifiers that have local significance only. That is, the same tunnel will be given different Tunnel IDs by each end of the tunnel. Tunnel ID in each message is that of the intended recipient, not the sender. Tunnel IDs are selected and exchanged as Assigned Tunnel ID AVPs during the creation of a tunnel.

Session ID indicates the identifier for a session within a tunnel. L2TP sessions are named by identifiers that have local significance only. That is, the same session will be given different Session IDs by each end of the session. Session ID in each message is that of the intended recipient, not the sender. Session IDs are selected and exchanged as Assigned Session ID AVPs during the creation of a session.

Ns indicates the sequence number for this data or control message, beginning at zero and incrementing by one (modulo 2**16) for each message sent. See Section 5.8 and 5.4 for more information on using

this field.

Nr indicates the sequence number expected in the next control message to be received. Thus, Nr is set to the Ns of the last in-order message received plus one (modulo 2**16). In data messages, Nr is reserved and, if present (as indicated by the S-bit), MUST be ignored upon receipt. See section 5.8 for more information on using this field in control messages.

The Offset Size field, if present, specifies the number of octets past the L2TP header at which the payload data is expected to start. Actual data within the offset padding is undefined. If the offset field is present, the L2TP header ends after the last octet of the offset padding.

3.2 Control Message Types

The Message Type AVP (see section 4.4.1) defines the specific type of control message being sent. Recall from section 3.1 that this is only for control messages, that is, messages with the T-bit set to 1.

This document defines the following control message types (see Section 6.1 through 6.14 for details on the construction and use of each message):

Control Connection Management

0	(reserved)	
1	(SCCRQ)	Start-Control-Connection-Request
2	(SCCRP)	Start-Control-Connection-Reply
3	(SCCCN)	Start-Control-Connection-Connected
4	(StopCCN)	Stop-Control-Connection-Notification
5	(reserved)	
6	(HELLO)	Hello

Call Management

7	(OCRQ)	Outgoing-Call-Request
8	(OCRP)	Outgoing-Call-Reply
9	(OCCN)	Outgoing-Call-Connected
10	(ICRQ)	Incoming-Call-Request
11	(ICRP)	Incoming-Call-Reply
12	(ICCN)	Incoming-Call-Connected
13	(reserved)	
14	(CDN)	Call-Disconnect-Notify

Error Reporting

T-Mobile Exhibit 1002
Part 4 of 6

15 (WEN) WAN-Error-Notify

PPP Session Control

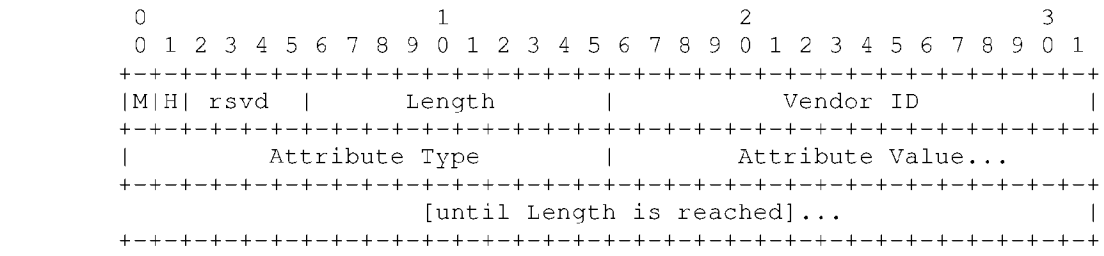
16 (SLI) Set-Link-Info

4.0 Control Message Attribute Value Pairs

To maximize extensibility while still permitting interoperability, a uniform method for encoding message types and bodies is used throughout L2TP. This encoding will be termed AVP (Attribute-Value Pair) in the remainder of this document.

4.1 AVP Format

Each AVP is encoded as:



The first six bits are a bit mask, describing the general attributes of the AVP.

Two bits are defined in this document, the remaining are reserved for future extensions. Reserved bits MUST be set to 0. An AVP received with a reserved bit set to 1 MUST be treated as an unrecognized AVP.

Mandatory (M) bit: Controls the behavior required of an implementation which receives an AVP which it does not recognize. If the M bit is set on an unrecognized AVP within a message associated with a particular session, the session associated with this message MUST be terminated. If the M bit is set on an unrecognized AVP within a message associated with the overall tunnel, the entire tunnel (and all sessions within) MUST be terminated. If the M bit is not set, an unrecognized AVP MUST be ignored. The control message must then continue to be processed as if the AVP had not been present.

Hidden (H) bit: Identifies the hiding of data in the Attribute Value field of an AVP. This capability can be used to avoid the passing of sensitive data, such as user passwords, as cleartext in an AVP. Section 4.3 describes the procedure for performing AVP hiding.

Length: Encodes the number of octets (including the Overall Length and bitmask fields) contained in this AVP. The Length may be calculated as 6 + the length of the Attribute Value field in octets. The field itself is 10 bits, permitting a maximum of 1023 octets of data in a single AVP. The minimum Length of an AVP is 6. If the length is 6, then the Attribute Value field is absent.

Vendor ID: The IANA assigned "SMI Network Management Private Enterprise Codes" [RFC1700] value. The value 0, corresponding to IETF adopted attribute values, is used for all AVPs defined within this document. Any vendor wishing to implement their own L2TP extensions can use their own Vendor ID along with private Attribute

values, guaranteeing that they will not collide with any other vendor's extensions, nor with future IETF extensions. Note that there are 16 bits allocated for the Vendor ID, thus limiting this feature to the first 65,535 enterprises.

Attribute Type: A 2 octet value with a unique interpretation across all AVPs defined under a given Vendor ID.

Attribute Value: This is the actual value as indicated by the Vendor ID and Attribute Type. It follows immediately after the Attribute Type field, and runs for the remaining octets indicated in the Length (i.e., Length minus 6 octets of header). This field is absent if the Length is 6.

4.2 Mandatory AVPs

Receipt of an unknown AVP that has the M-bit set is catastrophic to the session or tunnel it is associated with. Thus, the M bit should only be defined for AVPs which are absolutely crucial to proper operation of the session or tunnel. Further, in the case where the LAC or LNS receives an unknown AVP with the M-bit set and shuts down the session or tunnel accordingly, it is the full responsibility of the peer sending the Mandatory AVP to accept fault for causing an non-interoperable situation. Before defining an AVP with the M-bit set, particularly a vendor-specific AVP, be sure that this is the intended consequence.

When an adequate alternative exists to use of the M-bit, it should be utilized. For example, rather than simply sending an AVP with the M-bit set to determine if a specific extension exists, availability may be identified by sending an AVP in a request message and expecting a corresponding AVP in a reply message.

Use of the M-bit with new AVPs (those not defined in this document) MUST provide the ability to configure the associated feature off, such that the AVP is either not sent, or sent with the M-bit not set.

4.3 Hiding of AVP Attribute Values

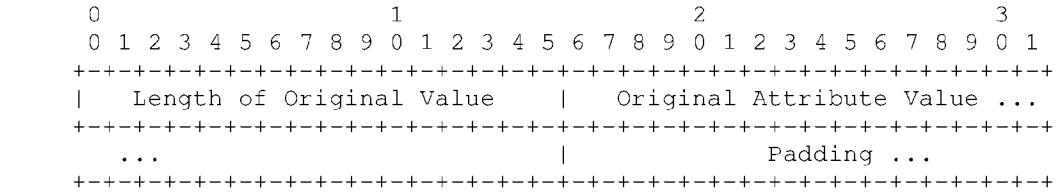
The H bit in the header of each AVP provides a mechanism to indicate to the receiving peer whether the contents of the AVP are hidden or present in cleartext. This feature can be used to hide sensitive control message data such as user passwords or user IDs.

The H bit MUST only be set if a shared secret exists between the LAC

and LNS. The shared secret is the same secret that is used for tunnel authentication (see Section 5.1.1). If the H bit is set in any

AVP(s) in a given control message, a Random Vector AVP must also be present in the message and MUST precede the first AVP having an H bit of 1.

Hiding an AVP value is done in several steps. The first step is to take the length and value fields of the original (cleartext) AVP and encode them into a Hidden AVP Subformat as follows:



Length of Original Attribute Value: This is length of the Original Attribute Value to be obscured in octets. This is necessary to determine the original length of the Attribute Value which is lost when the additional Padding is added.

Original Attribute Value: Attribute Value that is to be obscured.

Padding: Random additional octets used to obscure length of the Attribute Value that is being hidden.

To mask the size of the data being hidden, the resulting subformat MAY be padded as shown above. Padding does NOT alter the value placed in the Length of Original Attribute Value field, but does alter the length of the resultant AVP that is being created. For example, If an Attribute Value to be hidden is 4 octets in length, the unhidden AVP length would be 10 octets (6 + Attribute Value length). After hiding, the length of the AVP will become 6 + Attribute Value length + size of the Length of Original Attribute Value field + Padding. Thus, if Padding is 12 octets, the AVP length will be 6 + 4 + 2 + 12 = 24 octets.

Next, An MD5 hash is performed on the concatenation of:

+ the 2 octet Attribute number of the AVP
+ the shared secret
+ an arbitrary length random vector

The value of the random vector used in this hash is passed in the value field of a Random Vector AVP. This Random Vector AVP must be placed in the message by the sender before any hidden AVPs. The same random vector may be used for more than one hidden AVP in the same

message. If a different random vector is used for the hiding of subsequent AVPs then a new Random Vector AVP must be placed in the command message before the first AVP to which it applies.

The MD5 hash value is then XORed with the first 16 octet (or less) segment of the Hidden AVP Subformat and placed in the Attribute Value field of the Hidden AVP. If the Hidden AVP Subformat is less than 16 octets, the Subformat is transformed as if the Attribute Value field had been padded to 16 octets before the XOR, but only the actual octets present in the Subformat are modified, and the length of the AVP is not altered.

If the Subformat is longer than 16 octets, a second one-way MD5 hash is calculated over a stream of octets consisting of the shared secret followed by the result of the first XOR. That hash is XORed with the second 16 octet (or less) segment of the Subformat and placed in the corresponding octets of the Value field of the Hidden AVP.

If necessary, this operation is repeated, with the shared secret used along with each XOR result to generate the next hash to XOR the next segment of the value with.

The hiding method was adapted from RFC 2138 [RFC2138] which was taken from the "Mixing in the Plaintext" section in the book "Network Security" by Kaufman, Perlman and Speciner [KPS]. A detailed explanation of the method follows:

Call the shared secret S, the Random Vector RV, and the Attribute Value AV. Break the value field into 16-octet chunks p1, p2, etc. with the last one padded at the end with random data to a 16-octet boundary. Call the ciphertext blocks c(1), c(2), etc. We will also define intermediate values b1, b2, etc.

```
b1 = MD5(AV + S + RV)   c(1) = p1 xor b1
b2 = MD5(S  + c(1))     c(2) = p2 xor b2
            .                       .
            .                       .
            .                       .
bi = MD5(S  + c(i-1))   c(i) = pi xor bi
```

The String will contain c(1)+c(2)+...+c(i) where + denotes concatenation.

On receipt, the random vector is taken from the last Random Vector AVP encountered in the message prior to the AVP to be unhidden. The above process is then reversed to yield the original value.

4.4 AVP Summary

The following sections contain a list of all L2TP AVPs defined in this document.

Following the name of the AVP is a list indicating the message types that utilize each AVP. After each AVP title follows a short

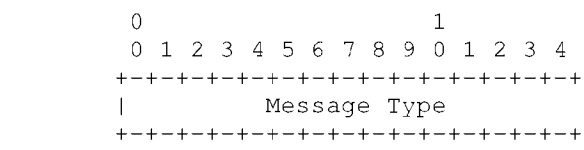
description of the purpose of the AVP, a detail (including a graphic) of the format for the Attribute Value, and any additional information needed for proper use of the avp.

4.4.1 AVPs Applicable To All Control Messages

Message Type (All Messages)

The Message Type AVP, Attribute Type 0, identifies the control message herein and defines the context in which the exact meaning of the following AVPs will be determined.

The Attribute Value field for this AVP has the following format:



The Message Type is a 2 octet unsigned integer.

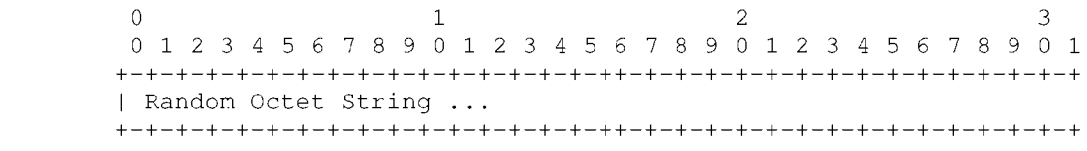
The Message Type AVP MUST be the first AVP in a message, immediately following the control message header (defined in section 3.1). See Section 3.2 for the list of defined control message types and their identifiers.

The Mandatory (M) bit within the Message Type AVP has special meaning. Rather than an indication as to whether the AVP itself should be ignored if not recognized, it is an indication as to whether the control message itself should be ignored. Thus, if the M-bit is set within the Message Type AVP and the Message Type is unknown to the implementation, the tunnel MUST be cleared. If the M-bit is not set, then the implementation may ignore an unknown message type. The M-bit MUST be set to 1 for all message types defined in this document. This AVP may not be hidden (the H-bit MUST be 0). The Length of this AVP is 8.

Random Vector (All Messages)

The Random Vector AVP, Attribute Type 36, is used to enable the hiding of the Attribute Value of arbitrary AVPs.

The Attribute Value field for this AVP has the following format:



The Random Octet String may be of arbitrary length, although a random vector of at least 16 octets is recommended. The string contains the random vector for use in computing the MD5 hash to

retrieve or hide the Attribute Value of a hidden AVP (see Section 4.2).

More than one Random Vector AVP may appear in a message, in which case a hidden AVP uses the Random Vector AVP most closely preceding it. This AVP MUST precede the first AVP with the H bit set.

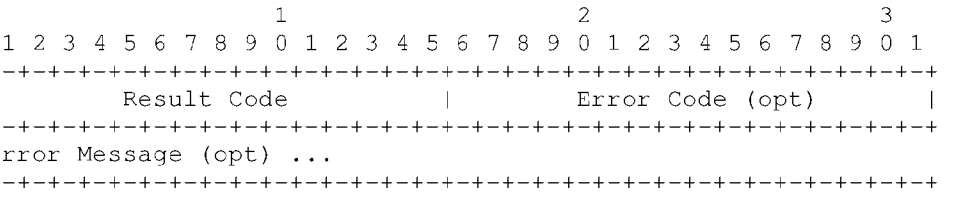
The M-bit for this AVP MUST be set to 1. This AVP MUST NOT be hidden (the H-bit MUST be 0). The Length of this AVP is 6 plus the length of the Random Octet String.

4.4.2 Result and Error Codes

Result Code (CDN, StopCCN)

The Result Code AVP, Attribute Type 1, indicates the reason for terminating the control channel or session.

The Attribute Value field for this AVP has the following format:



The Result Code is a 2 octet unsigned integer. The optional Error Code is a 2 octet unsigned integer. An optional Error Message can follow the Error Code field. Presence of the Error Code and

Message are indicated by the AVP Length field. The Error Message contains an arbitrary string providing further (human readable) text associated with the condition. Human readable text in all error messages MUST be provided in the UTF-8 charset using the Default Language [RFC2277].

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 1. The Length is 8 if there is no Error Code or Message, 10 if there is an Error Code and no Error Message or 10 + the length of the Error Message if there is an Error Code and Message.

Defined Result Code values for the StopCCN message are:

- 0 - Reserved
- 1 - General request to clear control connection
- 2 - General error--Error Code indicates the problem
- 3 - Control channel already exists
- 4 - Requester is not authorized to establish a control channel
- 5 - The protocol version of the requester is not supported
  Error Code indicates highest version supported
- 6 - Requester is being shut down
- 7 - Finite State Machine error

Defined Result Code values for the CDN message are:

0 - Reserved
1 - Call disconnected due to loss of carrier
2 - Call disconnected for the reason indicated in error code
3 - Call disconnected for administrative reasons
4 - Call failed due to lack of appropriate facilities being available (temporary condition)
5 - Call failed due to lack of appropriate facilities being available (permanent condition)
6 - Invalid destination
7 - Call failed due to no carrier detected
8 - Call failed due to detection of a busy signal
9 - Call failed due to lack of a dial tone
10 - Call was not established within time allotted by LAC
11 - Call was connected but no appropriate framing was detected

The Error Codes defined below pertain to types of errors that are not specific to any particular L2TP request, but rather to protocol or message format errors. If an L2TP reply indicates in

its Result Code that a general error occurred, the General Error value should be examined to determine what the error was. The currently defined General Error codes and their meanings are:

0 - No general error
1 - No control connection exists yet for this LAC-LNS pair
2 - Length is wrong
3 - One of the field values was out of range or reserved field was non-zero
4 - Insufficient resources to handle this operation now
5 - The Session ID is invalid in this context
6 - A generic vendor-specific error occurred in the LAC
7 - Try another. If LAC is aware of other possible LNS destinations, it should try one of them. This can be used to guide an LAC based on LNS policy, for instance, the existence of multilink PPP bundles.
8 - Session or tunnel was shutdown due to receipt of an unknown AVP with the M-bit set (see section 4.2). The Error Message SHOULD contain the attribute of the offending AVP in (human readable) text form.

When a General Error Code of 6 is used, additional information about the error SHOULD be included in the Error Message field.

4.4.3 Control Connection Management AVPs

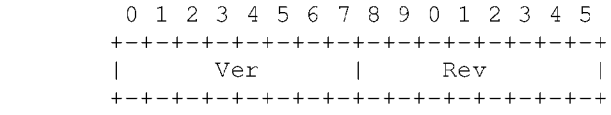
Protocol Version (SCCRP, SCCRQ)

The Protocol Version AVP, Attribute Type 2, indicates the L2TP protocol version of the sender.

The Attribute Value field for this AVP has the following format:

0 1

Page 1291
T-Mobile Exhibit 1002
Part 4 of 6



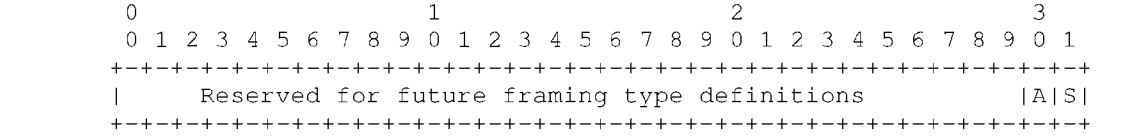
The Ver field is a 1 octet unsigned integer containing the value 1. Rev field is a 1 octet unsigned integer containing 0. This pertains to L2TP protocol version 1, revision 0. Note this is not the same version number that is included in the header of each message.

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 1. The Length of this AVP is 8.

Framing Capabilities (SCCRP, SCCRQ)

The Framing Capabilities AVP, Attribute Type 3, provides the peer with an indication of the types of framing that will be accepted or requested by the sender.

The Attribute Value field for this AVP has the following format:



The Attribute Value field is a 32-bit mask, with two bits defined. If bit A is set, asynchronous framing is supported. If bit S is set, synchronous framing is supported.

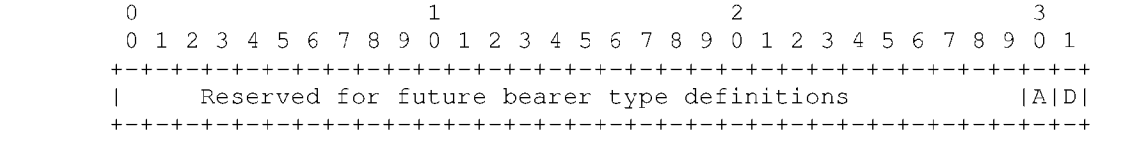
A peer MUST NOT request an incoming or outgoing call with a Framing Type AVP specifying a value not advertised in the Framing Capabilities AVP it received during control connection establishment. Attempts to do so will result in the call being rejected.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) is 10.

Bearer Capabilities (SCCRP, SCCRQ)

The Bearer Capabilities AVP, Attribute Type 4, provides the peer with an indication of the bearer device types supported by the hardware interfaces of the sender for outgoing calls.

The Attribute Value field for this AVP has the following format:



This is a 32-bit mask, with two bits defined. If bit A is set, analog access is supported. If bit D is set, digital access is

supported.

An LNS should not request an outgoing call specifying a value in the Bearer Type AVP for a device type not advertised in the Bearer Capabilities AVP it received from the LAC during control connection establishment. Attempts to do so will result in the call being rejected.

This AVP MUST be present if the sender can place outgoing calls when requested.

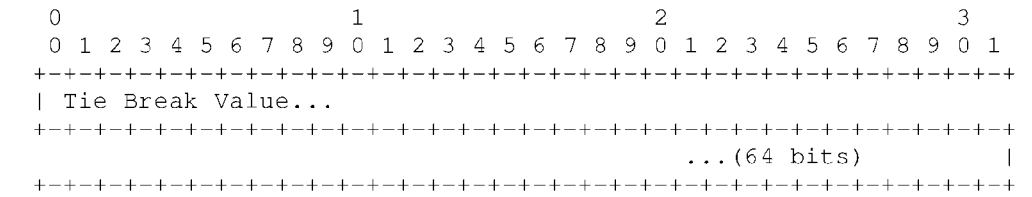
Note that an LNS that cannot act as an LAC as well will not support hardware devices for handling incoming and outgoing calls and should therefore set the A and D bits of this AVP to 0, or should not send the AVP at all. An LNS that can also act as an LAC and place outgoing calls should set A or D as appropriate. Presence of this message is not a guarantee that a given outgoing call will be placed by the sender if requested, just that the physical capability exists.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) is 10.

Tie Breaker (SCCRQ)

The Tie Breaker AVP, Attribute Type 5, indicates that the sender wishes a single tunnel to exist between the given LAC-LNS pair.

The Attribute Value field for this AVP has the following format:



The Tie Breaker Value is an 8 octet value that is used to choose a single tunnel where both LAC and LNS request a tunnel concurrently. The recipient of a SCCRQ must check to see if a SCCRQ has been sent to the peer, and if so, must compare its Tie Breaker value with the received one. The lower value "wins", and the "loser" MUST silently discard its tunnel. In the case where a tie breaker is present on both sides, and the value is equal, both sides MUST discard their tunnels.

Page 1293
T-Mobile Exhibit 1002
Part 4 of 6

http://www.ietf.org/rfc/rfc2661.txt

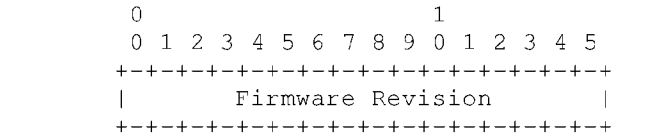
If a tie breaker is received, and an outstanding SCCRQ had no tie breaker value, the initiator which included the Tie Breaker AVP "wins". If neither side issues a tie breaker, then two separate tunnels are opened.

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 0. The Length of this AVP is 14.

Firmware Revision (SCCRP, SCCRQ)

The Firmware Revision AVP, Attribute Type 6, indicates the firmware revision of the issuing device.

The Attribute Value field for this AVP has the following format:



The Firmware Revision is a 2 octet unsigned integer encoded in a vendor specific format.

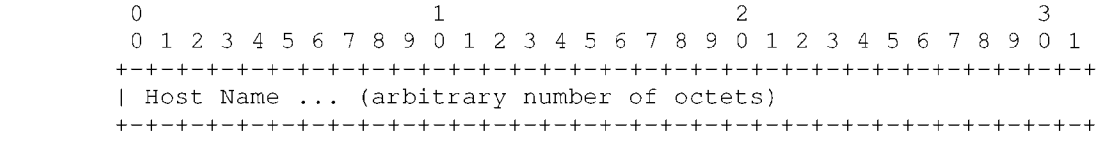
For devices which do not have a firmware revision (general purpose computers running L2TP software modules, for instance), the revision of the L2TP software module may be reported instead.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) is 8.

Host Name (SCCRP, SCCRQ)

The Host Name AVP, Attribute Type 7, indicates the name of the issuing LAC or LNS.

The Attribute Value field for this AVP has the following format:



The Host Name is of arbitrary length, but MUST be at least 1 octet.

This name should be as broadly unique as possible; for hosts participating in DNS [RFC1034], a hostname with fully qualified domain would be appropriate.

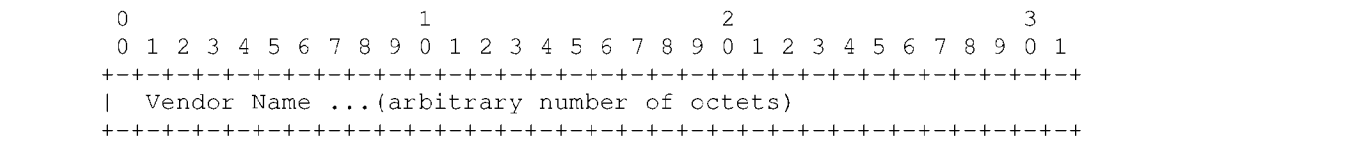
This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for

this AVP MUST be set to 1. The Length of this AVP is 6 plus the length of the Host Name.

Vendor Name (SCCRP, SCCRQ)

The Vendor Name AVP, Attribute Type 8, contains a vendor specific (possibly human readable) string describing the type of LAC or LNS being used.

The Attribute Value field for this AVP has the following format:



The Vendor Name is the indicated number of octets representing the vendor string. Human readable text for this AVP MUST be provided in the UTF-8 charset using the Default Language [RFC2277].

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6 plus the length of the Vendor Name.

Assigned Tunnel ID (SCCRP, SCCRQ, StopCCN)

The Assigned Tunnel ID AVP, Attribute Type 9, encodes the ID being assigned to this tunnel by the sender.

The Attribute Value field for this AVP has the following format:



The Assigned Tunnel ID is a 2 octet non-zero unsigned integer.

The Assigned Tunnel ID AVP establishes a value used to multiplex and demultiplex multiple tunnels between the LNS and LAC. The L2TP peer MUST place this value in the Tunnel ID header field of all

control and data messages that it subsequently transmits over the associated tunnel. Before the Assigned Tunnel ID AVP is received from a peer, messages MUST be sent to that peer with a Tunnel ID value of 0 in the header of all control messages.

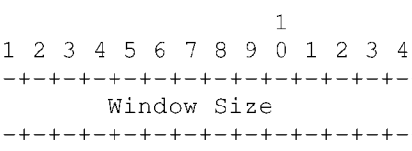
In the StopCCN control message, the Assigned Tunnel ID AVP MUST be the same as the Assigned Tunnel ID AVP first sent to the receiving peer, permitting the peer to identify the appropriate tunnel even if a StopCCN is sent before an Assigned Tunnel ID AVP is received.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 8.

Receive Window Size (SCCRQ, SCCRP)

The Receive Window Size AVP, Attribute Type 10, specifies the receive window size being offered to the remote peer.

The Attribute Value field for this AVP has the following format:



The Window Size is a 2 octet unsigned integer.

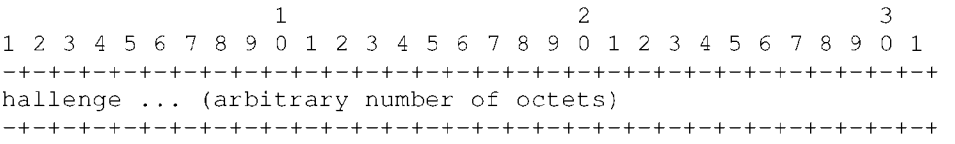
If absent, the peer must assume a Window Size of 4 for its transmit window. The remote peer may send the specified number of control messages before it must wait for an acknowledgment.

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 1. The Length of this AVP is 8.

Challenge (SCCRP, SCCRQ)

The Challenge AVP, Attribute Type 11, indicates that the issuing peer wishes to authenticate the tunnel endpoints using a CHAP-style authentication mechanism.

The Attribute Value field for this AVP has the following format:



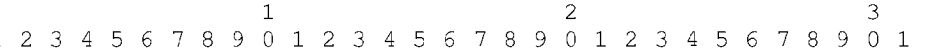
The Challenge is one or more octets of random data.

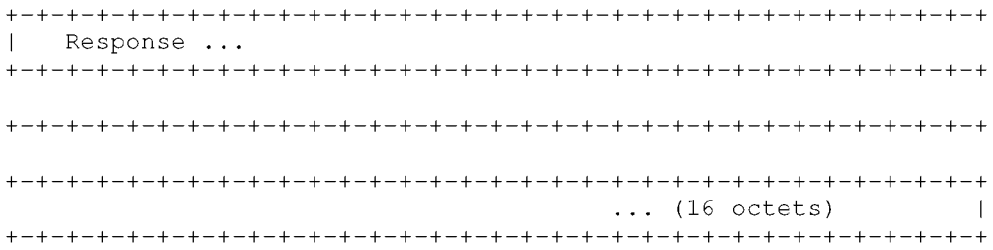
This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 6 plus the length of the Challenge.

Challenge Response (SCCCN, SCCRP)

The Response AVP, Attribute Type 13, provides a response to a challenge received.

The Attribute Value field for this AVP has the following format:





The Response is a 16 octet value reflecting the CHAP-style [RFC1994] response to the challenge.

This AVP MUST be present in an SCCRP or SCCCN if a challenge was received in the preceding SCCRQ or SCCRP. For purposes of the ID value in the CHAP response calculation, the value of the Message Type AVP for this message is used (e.g. 2 for an SCCRP, and 3 for an SCCCN).

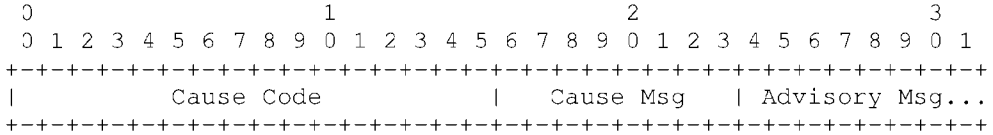
This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 22.

4.4.4 Call Management AVPs

Q.931 Cause Code (CDN)

The Q.931 Cause Code AVP, Attribute Type 12, is used to give additional information in case of unsolicited call disconnection.

The Attribute Value field for this AVP has the following format:



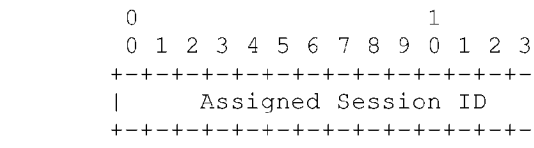
Cause Code is the returned Q.931 Cause code, and Cause Msg is the returned Q.931 message code (e.g., DISCONNECT) associated with the Cause Code. Both values are returned in their native ITU encodings [DSS1]. An additional ASCII text Advisory Message may also be included (presence indicated by the AVP Length) to further explain the reason for disconnecting.

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 1. The Length of this AVP is 9, plus the size of the Advisory Message.

Assigned Session ID (CDN, ICRP, ICRQ, OCRP, OCRQ)

The Assigned Session ID AVP, Attribute Type 14, encodes the ID being assigned to this session by the sender.

The Attribute Value field for this AVP has the following format:



The Assigned Session ID is a 2 octet non-zero unsigned integer.

The Assigned Session ID AVP is establishes a value used to multiplex and demultiplex data sent over a tunnel between the LNS and LAC. The L2TP peer MUST place this value in the Session ID header field of all control and data messages that it subsequently transmits over the tunnel that belong to this session. Before the

Assigned Session ID AVP is received from a peer, messages MUST be sent to that peer with a Session ID of 0 in the header of all control messages.

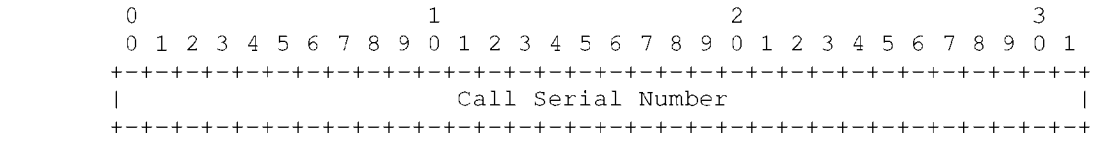
In the CDN control message, the same Assigned Session ID AVP first sent to the receiving peer is used, permitting the peer to identify the appropriate tunnel even if CDN is sent before an Assigned Session ID is received.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 8.

Call Serial Number (ICRQ, OCRQ)

The Call Serial Number AVP, Attribute Type 15, encodes an identifier assigned by the LAC or LNS to this call.

The Attribute Value field for this AVP has the following format:



The Call Serial Number is a 32 bit value.

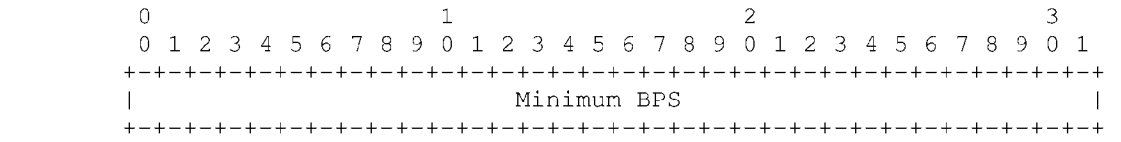
The Call Serial Number is intended to be an easy reference for administrators on both ends of a tunnel to use when investigating call failure problems. Call Serial Numbers should be set to progressively increasing values, which are likely to be unique for a significant period of time across all interconnected LNSs and LACs.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 10.

Minimum BPS (OCRQ)

The Minimum BPS AVP, Attribute Type 16, encodes the lowest acceptable line speed for this call.

The Attribute Value field for this AVP has the following format:



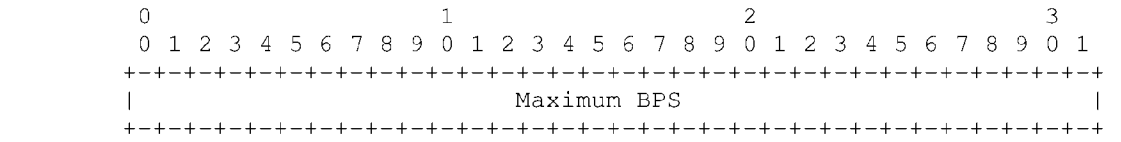
The Minimum BPS is a 32 bit value indicates the speed in bits per second.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 10.

Maximum BPS (OCRQ)

The Maximum BPS AVP, Attribute Type 17, encodes the highest acceptable line speed for this call.

The Attribute Value field for this AVP has the following format:



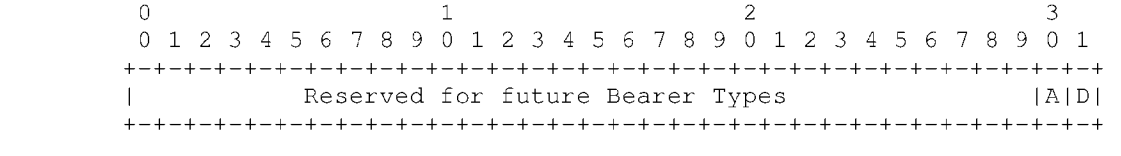
The Maximum BPS is a 32 bit value indicates the speed in bits per second.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 10.

Bearer Type (ICRQ, OCRQ)

The Bearer Type AVP, Attribute Type 18, encodes the bearer type for the incoming or outgoing call.

The Attribute Value field for this AVP has the following format:



The Bearer Type is a 32-bit bit mask, which indicates the bearer capability of the call (ICRQ) or required for the call (OCRQ). If set, bit A indicates that the call refers to an analog channel. If set, bit D indicates that the call refers to a digital channel. Both may be set, indicating that the call was either indistinguishable, or can be placed on either type of channel.

Bits in the Value field of this AVP MUST only be set by the LNS for an OCRQ if it was set in the Bearer Capabilities AVP received from the LAC during control connection establishment.

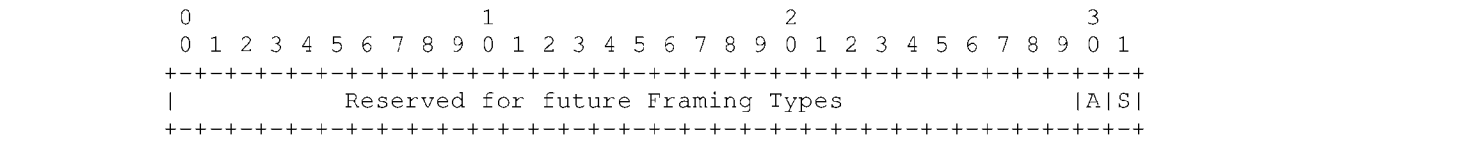
It is valid to set neither the A nor D bits in an ICRQ. Such a setting may indicate that the call was not received over a physical link (e.g if the LAC and PPP are located in the same subsystem).

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 10.

Framing Type (ICCN, OCCN, OCRQ)

The Framing Type AVP, Attribute Type 19, encodes the framing type for the incoming or outgoing call.

The Attribute Value field for this AVP has the following format:



The Framing Type is a 32-bit mask, which indicates the type of PPP framing requested for an OCRQ, or the type of PPP framing negotiated for an OCCN or ICCN. The framing type MAY be used as an indication to PPP on the LNS as to what link options to use for LCP negotiation [RFC1662].

Bit A indicates asynchronous framing. Bit S indicates synchronous framing. For an OCRQ, both may be set, indicating that either type of framing may be used.

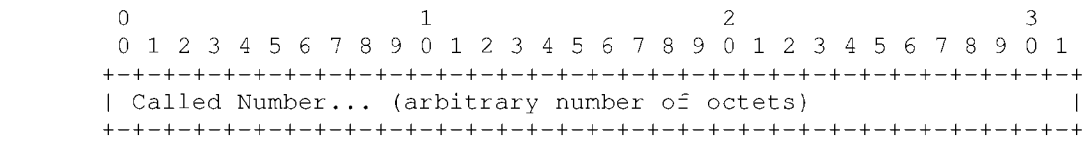
Bits in the Value field of this AVP MUST only be set by the LNS for an OCRQ if it was set in the Framing Capabilities AVP received from the LAC during control connection establishment.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 10.

Called Number (ICRQ, OCRQ)

The Called Number AVP, Attribute Type 21, encodes the telephone number to be called for an OCRQ, and the Called number for an ICRQ.

The Attribute Value field for this AVP has the following format:



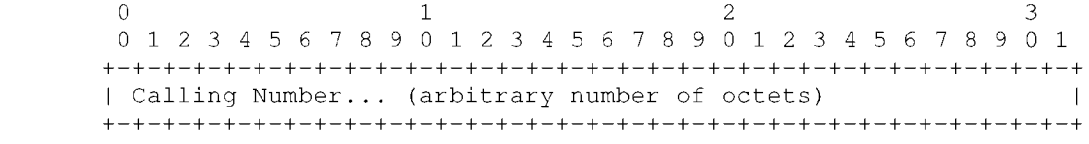
The Called Number is an ASCII string. Contact between the administrator of the LAC and the LNS may be necessary to coordinate interpretation of the value needed in this AVP.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 6 plus the length of the Called Number.

Calling Number (ICRQ)

The Calling Number AVP, Attribute Type 22, encodes the originating number for the incoming call.

The Attribute Value field for this AVP has the following format:



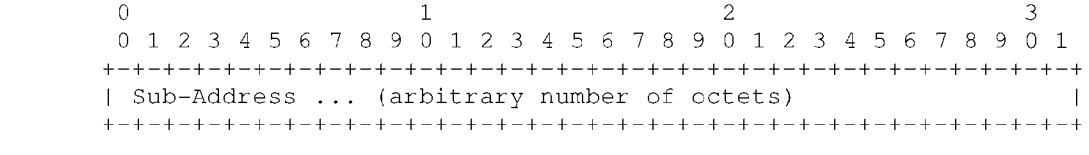
Calling Number is an ASCII string. Contact between the administrator of the LAC and the LNS may be necessary to coordinate interpretation of the value in this AVP.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 6 plus the length of the Calling Number.

Sub-Address (ICRQ, OCRQ)

The Sub-Address AVP, Attribute Type 23, encodes additional dialing information.

The Attribute Value field for this AVP has the following format:



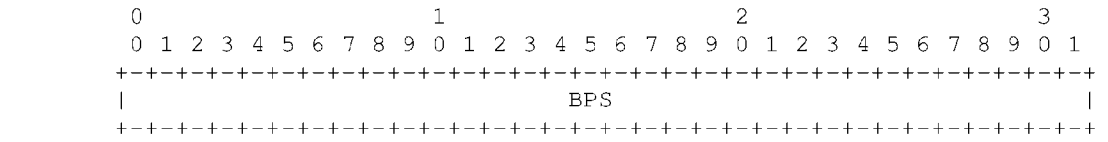
The Sub-Address is an ASCII string. Contact between the administrator of the LAC and the LNS may be necessary to coordinate interpretation of the value in this AVP.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 6 plus the length of the Sub-Address.

(Tx) Connect Speed (ICCN, OCCN)

The (Tx) Connect Speed BPS AVP, Attribute Type 24, encodes the speed of the facility chosen for the connection attempt.

The Attribute Value field for this AVP has the following format:



The (Tx) Connect Speed BPS is a 4 octet value indicating the speed in bits per second.

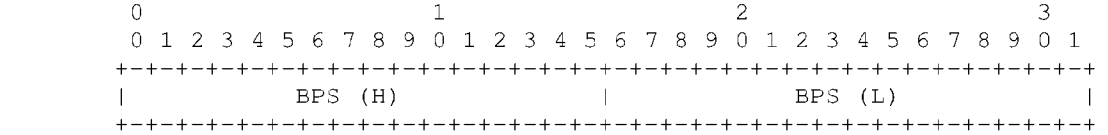
When the optional Rx Connect Speed AVP is present, the value in this AVP represents the transmit connect speed, from the perspective of the LAC (e.g. data flowing from the LAC to the remote system). When the optional Rx Connect Speed AVP is NOT present, the connection speed between the remote system and LAC is assumed to be symmetric and is represented by the single value in this AVP.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 10.

Rx Connect Speed (ICCN, OCCN)

The Rx Connect Speed AVP, Attribute Type 38, represents the speed of the connection from the perspective of the LAC (e.g. data flowing from the remote system to the LAC).

The Attribute Value field for this AVP has the following format:



BPS is a 4 octet value indicating the speed in bits per second.

Presence of this AVP implies that the connection speed may be asymmetric with respect to the transmit connect speed given in the (Tx) Connect Speed AVP.

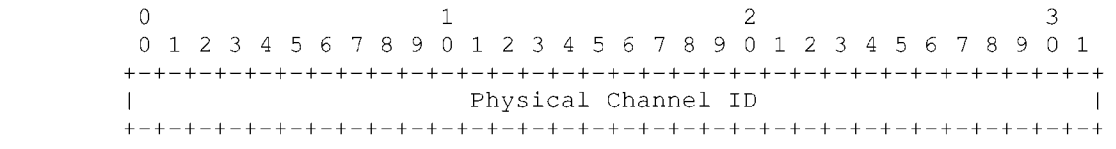
This AVP may be hidden (the H-bit MAY be 1 or 0). The M-bit for

this AVP MUST be set to 0. The Length (before hiding) of this AVP is 10.

Physical Channel ID (ICRQ, OCRP)

The Physical Channel ID AVP, Attribute Type 25, encodes the vendor specific physical channel number used for a call.

The Attribute Value field for this AVP has the following format:



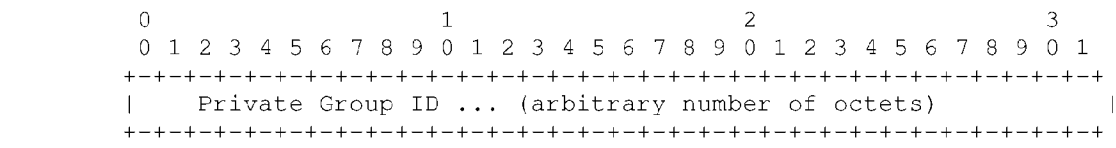
Physical Channel ID is a 4 octet value intended to be used for logging purposes only.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 10.

Private Group ID (ICCN)

The Private Group ID AVP, Attribute Type 37, is used by the LAC to indicate that this call is to be associated with a particular customer group.

The Attribute Value field for this AVP has the following format:



The Private Group ID is a string of octets of arbitrary length.

The LNS MAY treat the PPP session as well as network traffic through this session in a special manner determined by the peer. For example, if the LNS is individually connected to several private networks using unregistered addresses, this AVP may be included by the LAC to indicate that a given call should be associated with one of the private networks.

The Private Group ID is a string corresponding to a table in the LNS that defines the particular characteristics of the selected group. A LAC MAY determine the Private Group ID from a RADIUS response, local configuration, or some other source.

This AVP may be hidden (the H-bit MAY be 1 or 0). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP

is 6 plus the length of the Private Group ID.

Sequencing Required (ICCN, OCCN)

The Sequencing Required AVP, Attribute Type 39, indicates to the LNS that Sequence Numbers MUST always be present on the data channel.

This AVP has no Attribute Value field.

This AVP MUST NOT be hidden (the H-bit MUST be 0). The M-bit for this AVP MUST be set to 1. The Length of this AVP is 6.

4.4.5 Proxy LCP and Authentication AVPs

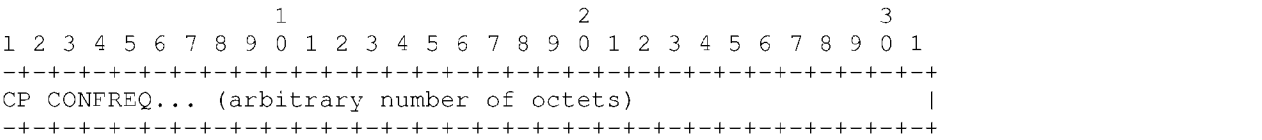
The LAC may have answered the call and negotiated LCP with the remote system, perhaps in order to establish the system's apparent identity. In this case, these AVPs may be included to indicate the

link properties the remote system initially requested, properties the remote system and LAC ultimately negotiated, as well as PPP authentication information sent and received by the LAC. This information may be used to initiate the PPP LCP and authentication systems on the LNS, allowing PPP to continue without renegotiation of LCP. Note that the LNS policy may be to enter an additional round of LCP negotiation and/or authentication if the LAC is not trusted.

Initial Received LCP CONFREQ (ICCN)

In the Initial Received LCP CONFREQ AVP, Attribute Type 26, provides the LNS with the Initial CONFREQ received by the LAC from the PPP Peer.

The Attribute Value field for this AVP has the following format:



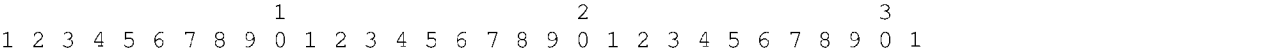
LCP CONFREQ is a copy of the body of the initial CONFREQ received, starting at the first option within the body of the LCP message.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6 plus the length of the CONFREQ.

Last Sent LCP CONFREQ (ICCN)

In the Last Sent LCP CONFREQ AVP, Attribute Type 27, provides the LNS with the Last CONFREQ sent by the LAC to the PPP Peer.

The Attribute Value field for this AVP has the following format:



```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| LCP CONFREQ... (arbitrary number of octets)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The LCP CONFREQ is a copy of the body of the final CONFREQ sent to the client to complete LCP negotiation, starting at the first option within the body of the LCP message.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6 plus the length of the CONFREQ.

Last Received LCP CONFREQ (ICCN)

The Last Received LCP CONFREQ AVP, Attribute Type 28, provides the LNS with the Last CONFREQ received by the LAC from the PPP Peer.

The Attribute Value field for this AVP has 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| LCP CONFREQ... (arbitrary number of octets)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The LCP CONFREQ is a copy of the body of the final CONFREQ received from the client to complete LCP negotiation, starting at the first option within the body of the LCP message.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6 plus the length of the CONFREQ.

Proxy Authen Type (ICCN)

The Proxy Authen Type AVP, Attribute Type 29, determines if proxy authentication should be used.

The Attribute Value field for this AVP has the following format:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Authen Type          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Authen Type is a 2 octet unsigned integer, holding:

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 8.

Defined Authen Type values are:
- 0 - Reserved
- 1 - Textual username/password exchange
- 2 - PPP CHAP
- 3 - PPP PAP
- 4 - No Authentication
- 5 - Microsoft CHAP Version 1 (MSCHAPv1)

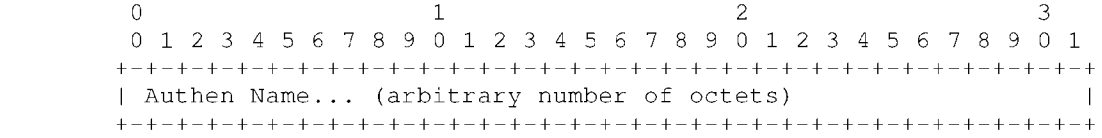
This AVP MUST be present if proxy authentication is to be utilized. If it is not present, then it is assumed that this peer cannot perform proxy authentication, requiring a restart of the authentication phase at the LNS if the client has already entered this phase with the LAC (which may be determined by the Proxy LCP AVP if present).

Associated AVPs for each type of authentication follow.

Proxy Authen Name (ICCN)

The Proxy Authen Name AVP, Attribute Type 30, specifies the name of the authenticating client when using proxy authentication.

The Attribute Value field for this AVP has the following format:



Authen Name is a string of octets of arbitrary length. It contains the name specified in the client's authentication response.

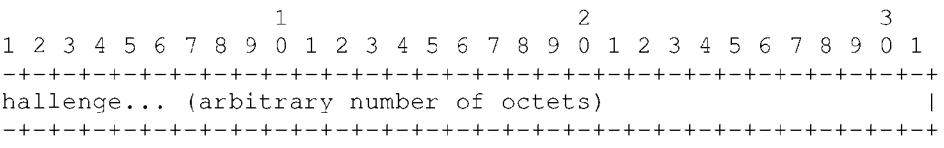
This AVP MUST be present in messages containing a Proxy Authen Type AVP with an Authen Type of 1, 2, 3 or 5. It may be desirable to employ AVP hiding for obscuring the cleartext name.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) is 6 plus the length of the cleartext name.

Proxy Authen Challenge (ICCN)

The Proxy Authen Challenge AVP, Attribute Type 31, specifies the challenge sent by the LAC to the PPP Peer, when using proxy authentication.

The Attribute Value field for this AVP has the following format:



The Challenge is a string of one or more octets.

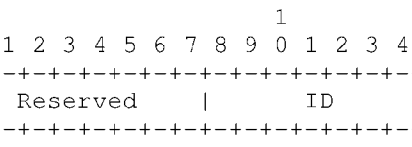
This AVP MUST be present for Proxy Authen Types 2 and 5. The Challenge field contains the CHAP challenge presented to the client by the LAC.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6, plus the length of the Challenge.

Proxy Authen ID (ICCN)

The Proxy Authen ID AVP, Attribute Type 32, specifies the ID value of the PPP Authentication that was started between the LAC and the PPP Peer, when proxy authentication is being used.

The Attribute Value field for this AVP has the following format:



ID is a 2 octet unsigned integer, the most significant octet MUST be 0.

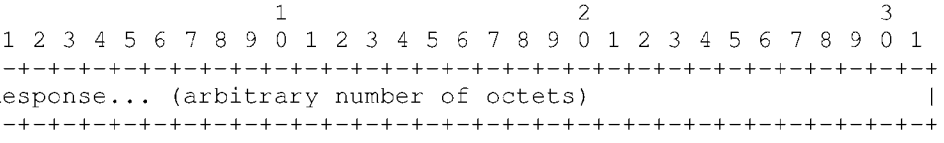
The Proxy Authen ID AVP MUST be present for Proxy authen types 2, 3 and 5. For 2 and 5, the ID field contains the byte ID value presented to the client by the LAC in its Challenge. For 3, it is the Identifier value of the Authenticate-Request.

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0.

Proxy Authen Response (ICCN)

The Proxy Authen Response AVP, Attribute Type 33, specifies the PPP Authentication response received by the LAC from the PPP Peer, when proxy authentication is used.

The Attribute Value field for this AVP has the following format:



The Response is a string of octets.

This AVP MUST be present for Proxy authen types 1, 2, 3 and 5. The Response field contains the client's response to the challenge. For Proxy authen types 2 and 5, this field contains the response value received by the LAC. For types 1 or 3, it contains the clear text password received from the client by the LAC. In the case of cleartext passwords, AVP hiding is recommended.

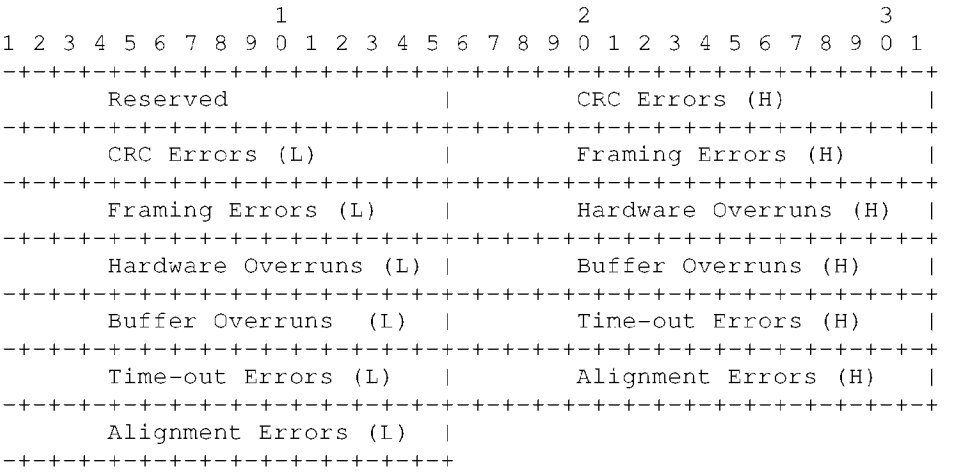
This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 0. The Length (before hiding) of this AVP is 6 plus the length of the Response.

4.4.6 Call Status AVPs

Call Errors (WEN)

The Call Errors AVP, Attribute Type 34, is used by the LAC to send error information to the LNS.

The Attribute Value field for this AVP has the following format:



The following fields are defined:

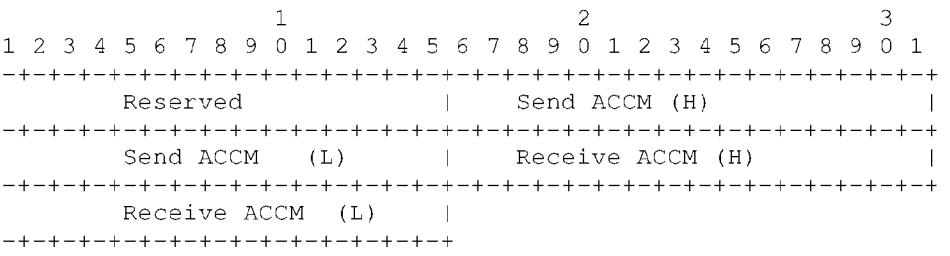
- Reserved - Not used, MUST be 0
- CRC Errors - Number of PPP frames received with CRC errors since call was established
- Framing Errors - Number of improperly framed PPP packets received
- Hardware Overruns - Number of receive buffer over-runs since call was established
- Buffer Overruns - Number of buffer over-runs detected since call was established
- Time-out Errors - Number of time-outs since call was established
- Alignment Errors - Number of alignment errors since call was established

This AVP may be hidden (the H-bit may be 0 or 1). The M-bit for this AVP MUST be set to 1. The Length (before hiding) of this AVP is 32.

ACCM (SLI)

The ACCM AVP, Attribute Type 35, is used by the LNS to inform LAC of the ACCM negotiated with the PPP Peer by the LNS.

The Attribute Value field for this AVP has the following format:



Send ACCM and Receive ACCM are each 4 octet values preceded by a 2 octet reserved quantity. The send ACCM value should be used by the LAC to process packets it sends on the connection. The receive ACCM value should be used by the LAC to process incoming packets on the connection. The default values used by the LAC for both these fields are 0xFFFFFFFF. The LAC should honor these fields unless it has specific configuration information to indicate that the requested mask must be modified to permit operation.

This AVP may be hidden (the H-bit MAY be 1 or 0). The M-bit for this AVP MUST be set to 1. The Length of this AVP is 16.

5.0 Protocol Operation

The necessary setup for tunneling a PPP session with L2TP consists of two steps, (1) establishing the Control Connection for a Tunnel, and (2) establishing a Session as triggered by an incoming or outgoing call request. The Tunnel and corresponding Control Connection MUST be established before an incoming or outgoing call is initiated. An L2TP Session MUST be established before L2TP can begin to tunnel PPP frames. Multiple Sessions may exist across a single Tunnel and multiple Tunnels may exist between the same LAC and LNS.

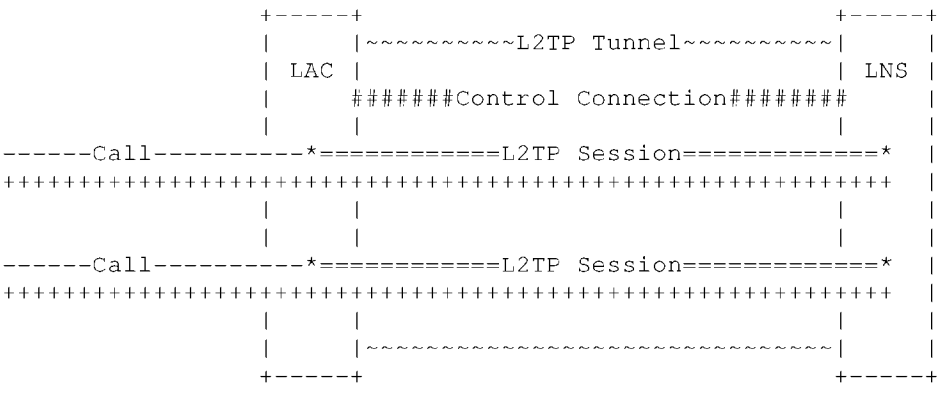


Figure 5.1 Tunneling PPP

5.1 Control Connection Establishment

The Control Connection is the initial connection that must be achieved between an LAC and LNS before sessions may be brought up. Establishment of the control connection includes securing the identity of the peer, as well as identifying the peer's L2TP version, framing, and bearer capabilities, etc.

A three message exchange is utilized to setup the control connection. Following is a typical message exchange:

```
   LAC or LNS  LAC or LNS
   ----------  ----------
   SCCRQ ->
               <- SCCRP
   SCCCN ->
               <- ZLB ACK
```

The ZLB ACK is sent if there are no further messages waiting in queue for that peer.

5.1.1 Tunnel Authentication

L2TP incorporates a simple, optional, CHAP-like [RFC1994] tunnel authentication system during control connection establishment. If an LAC or LNS wishes to authenticate the identity of the peer it is contacting or being contacted by, a Challenge AVP is included in the SCCRQ or SCCRP message. If a Challenge AVP is received in an SCCRQ or SCCRP, a Challenge Response AVP MUST be sent in the following SCCRP or SCCCN, respectively. If the expected response and response received from a peer does not match, establishment of the tunnel MUST be disallowed.

To participate in tunnel authentication, a single shared secret MUST exist between the LAC and LNS. This is the same shared secret used for AVP hiding (see Section 4.3). See Section 4.4.3 for details on construction of the Challenge and Response AVPs.

5.2 Session Establishment

After successful control connection establishment, individual sessions may be created. Each session corresponds to single PPP stream between the LAC and LNS. Unlike control connection establishment, session establishment is directional with respect to the LAC and LNS. The LAC requests the LNS to accept a session for an incoming call, and the LNS requests the LAC to accept a session for placing an outgoing call.

5.2.1 Incoming Call Establishment

A three message exchange is employed to setup the session. Following is a typical sequence of events:

```
   LAC         LNS
   ---         ---
   (Call
    Detected)
```

```
   ICRQ ->
            <- ICRP
   ICCN ->
            <- ZLB ACK
```

The ZLB ACK is sent if there are no further messages waiting in queue for that peer.

5.2.2 Outgoing Call Establishment

A three message exchange is employed to setup the session. Following is a typical sequence of events:

```
   LAC          LNS
   ---          ---
            <- OCRQ
   OCRP ->

   (Perform
    Call
    Operation)

   OCCN ->
            <- ZLB ACK
```

The ZLB ACK is sent if there are no further messages waiting in queue for that peer.

5.3 Forwarding PPP Frames

Once tunnel establishment is complete, PPP frames from the remote system are received at the LAC, stripped of CRC, link framing, and transparency bytes, encapsulated in L2TP, and forwarded over the appropriate tunnel. The LNS receives the L2TP packet, and processes the encapsulated PPP frame as if it were received on a local PPP interface.

The sender of a message associated with a particular session and tunnel places the Session ID and Tunnel ID (specified by its peer) in the Session ID and Tunnel ID header for all outgoing messages. In this manner, PPP frames are multiplexed and demultiplexed over a single tunnel between a given LNS-LAC pair. Multiple tunnels may exist between a given LNS-LAC pair, and multiple sessions may exist within a tunnel.

The value of 0 for Session ID and Tunnel ID is special and MUST NOT be used as an Assigned Session ID or Assigned Tunnel ID. For the cases where a Session ID has not yet been assigned by the peer (i.e., during establishment of a new session or tunnel), the Session ID field MUST be sent as 0, and the Assigned Session ID AVP within the message MUST be used to identify the session. Similarly, for cases where the Tunnel ID has not yet been assigned from the peer, the Tunnel ID MUST be sent as 0 and Assigned Tunnel ID AVP used to identify the tunnel.

http://www.ietf.org/rfc/rfc2661.txt

5.4 Using Sequence Numbers on the Data Channel

Sequence numbers are defined in the L2TP header for control messages and optionally for data messages (see Section 3.1). These are used to provide a reliable control message transport (see Section 5.8) and optional data message sequencing. Each peer maintains separate sequence numbers for the control connection and each individual data session within a tunnel.

Unlike the L2TP control channel, the L2TP data channel does not use sequence numbers to retransmit lost data messages. Rather, data messages may use sequence numbers to detect lost packets and/or restore the original sequence of packets that may have been reordered during transport. The LAC may request that sequence numbers be present in data messages via the Sequencing Required AVP (see Section 4.4.6). If this AVP is present during session setup, sequence numbers MUST be present at all times. If this AVP is not present, sequencing presence is under control of the LNS. The LNS controls enabling and disabling of sequence numbers by sending a data message with or without sequence numbers present at any time during the life of a session. Thus, if the LAC receives a data message without sequence numbers present, it MUST stop sending sequence numbers in future data messages. If the LAC receives a data message with sequence numbers present, it MUST begin sending sequence numbers in future outgoing data messages. If the LNS enables sequencing after disabling it earlier in the session, the sequence number state picks up where it left off before.

The LNS may initiate disabling of sequencing at any time during the session (including the first data message sent). It is recommended that for connections where reordering or packet loss may occur, sequence numbers always be enabled during the initial negotiation stages of PPP and disabled only when and if the risk is considered acceptable. For example, if the PPP session being tunneled is not utilizing any stateful compression or encryption protocols and is only carrying IP (as determined by the PPP NCPs that are established), then the LNS might decide to disable sequencing as IP is tolerant to datagram loss and reordering.

5.5 Keepalive (Hello)

A keepalive mechanism is employed by L2TP in order to differentiate tunnel outages from extended periods of no control or data activity on a tunnel. This is accomplished by injecting Hello control messages (see Section 6.5) after a specified period of time has elapsed since the last data or control message was received on a tunnel. As for any other control message, if the Hello message is not reliably delivered then the tunnel is declared down and is reset. The transport reset

mechanism along with the injection of Hello messages ensures that a connectivity failure between the LNS and the LAC will be detected at both ends of a tunnel.

5.6 Session Teardown

Session teardown may be initiated by either the LAC or LNS and is accomplished by sending a CDN control message. After the last session is cleared, the control connection MAY be torn down as well (and typically is). Following is an example of a typical control message exchange:

```
   LAC or LNS  LAC or LNS

   CDN ->
   (Clean up)

               <- ZLB ACK
                  (Clean up)
```

5.7 Control Connection Teardown

Control connection teardown may be initiated by either the LAC or LNS and is accomplished by sending a single StopCCN control message. The receiver of a StopCCN MUST send a ZLB ACK to acknowledge receipt of the message and maintain enough control connection state to properly accept StopCCN retransmissions over at least a full retransmission cycle (in case the ZLB ACK is lost). The recommended time for a full retransmission cycle is 31 seconds (see section 5.8). Following is an example of a typical control message exchange:

```
   LAC or LNS  LAC or LNS

   StopCCN ->
   (Clean up)

               <- ZLB ACK
                  (Wait)
                  (Clean up)
```

An implementation may shut down an entire tunnel and all sessions on the tunnel by sending the StopCCN. Thus, it is not necessary to clear each session individually when tearing down the whole tunnel.

5.8 Reliable Delivery of Control Messages

L2TP provides a lower level reliable transport service for all control messages. The Nr and Ns fields of the control message header (see section 3.1) belong to this transport. The upper level functions of L2TP are not concerned with retransmission or ordering of control messages. The reliable control message is a sliding window transport that provides control message retransmission and congestion

control. Each peer maintains separate sequence number state for the control connection within a tunnel.

The message sequence number, Ns, begins at 0. Each subsequent message is sent with the next increment of the sequence number. The sequence number is thus a free running counter represented modulo 65536. The sequence number in the header of a received message is considered less than or equal to the last received number if its value lies in the range of the last received number and the preceding 32767 values, inclusive. For example, if the last received sequence number was 15, then messages with sequence numbers 0 through 15, as well as 32784 through 65535, would be considered less than or equal. Such a message would be considered a duplicate of a message already received and ignored from processing. However, in order to ensure that all messages are acknowledged properly (particularly in the case of a lost ZLB ACK message), receipt of duplicate messages MUST be acknowledged by the reliable transport. This acknowledgement may either piggybacked on a message in queue, or explicitly via a ZLB ACK.

All control messages take up one slot in the control message sequence number space, except the ZLB acknowledgement. Thus, Ns is not incremented after a ZLB message is sent.

The last received message number, Nr, is used to acknowledge messages received by an L2TP peer. It contains the sequence number of the message the peer expects to receive next (e.g. the last Ns of a non-ZLB message received plus 1, modulo 65536). While the Nr in a received ZLB is used to flush messages from the local retransmit queue (see below), Nr of the next message sent is not be updated by the Ns of the ZLB.

The reliable transport at a receiving peer is responsible for making sure that control messages are delivered in order and without duplication to the upper level. Messages arriving out of order may be queued for in-order delivery when the missing messages are received, or they may be discarded requiring a retransmission by the peer.

Each tunnel maintains a queue of control messages to be transmitted to its peer. The message at the front of the queue is sent with a given Ns value, and is held until a control message arrives from the peer in which the Nr field indicates receipt of this message. After a period of time (a recommended default is 1 second) passes without acknowledgement, the message is retransmitted. The retransmitted message contains the same Ns value, but the Nr value MUST be updated with the sequence number of the next expected message.

Each subsequent retransmission of a message MUST employ an exponential backoff interval. Thus, if the first retransmission occurred after 1 second, the next retransmission should occur after 2 seconds has elapsed, then 4 seconds, etc. An implementation MAY place a cap upon the maximum interval between retransmissions. This cap MUST be no less than 8 seconds per retransmission. If no peer response is detected after several retransmissions, (a recommended default is 5, but SHOULD be configurable), the tunnel and all

sessions within MUST be cleared.

When a tunnel is being shut down for reasons other than loss of connectivity, the state and reliable delivery mechanisms MUST be maintained and operated for the full retransmission interval after the final message exchange has occurred.

A sliding window mechanism is used for control message transmission. Consider two peers A & B. Suppose A specifies a Receive Window Size AVP with a value of N in the SCCRQ or SCCRP messages. B is now allowed to have up to N outstanding control messages. Once N have been sent, it must wait for an acknowledgment that advances the window before sending new control messages. An implementation may support a receive window of only 1 (i.e., by sending out a Receive Window Size AVP with a value of 1), but MUST accept a window of up to 4 from its peer (e.g. have the ability to send 4 messages before backing off). A value of 0 for the Receive Window Size AVP is invalid.

When retransmitting control messages, a slow start and congestion avoidance window adjustment procedure SHOULD be utilized. The recommended procedure for this is described in Appendix A.

A peer MUST NOT withhold acknowledgment of messages as a technique for flow controlling control messages. An L2TP implementation is expected to be able to keep up with incoming control messages, possibly responding to some with errors reflecting an inability to honor the requested action.

Appendix B contains examples of control message transmission, acknowledgement, and retransmission.

6.0 Control Connection Protocol Specification

The following control connection messages are used to establish, clear and maintain L2TP tunnels. All data is sent in network order (high order octets first). Any "reserved" or "empty" fields MUST be sent as 0 values to allow for protocol extensibility.

6.1 Start-Control-Connection-Request (SCCRQ)

Start-Control-Connection-Request (SCCRQ) is a control message used to initialize a tunnel between an LNS and an LAC. It is sent by either the LAC or the LNS to being the tunnel establishment process.

The following AVPs MUST be present in the SCCRQ:

- Message Type AVP
- Protocol Version
- Host Name
- Framing Capabilities
- Assigned Tunnel ID

The Following AVPs MAY be present in the SCCRQ:

- Bearer Capabilities
- Receive Window Size
- Challenge

Page 1315
T-Mobile Exhibit 1002
Part 4 of 6

Tie Breaker
Firmware Revision
Vendor Name

6.2 Start-Control-Connection-Reply (SCCRP)

Start-Control-Connection-Reply (SCCRP) is a control message sent in reply to a received SCCRQ message. SCCRP is used to indicate that the SCCRQ was accepted and establishment of the tunnel should continue.

The following AVPs MUST be present in the SCCRP:

Message Type
Protocol Version
Framing Capabilities
Host Name
Assigned Tunnel ID

The following AVPs MAY be present in the SCCRP:

Bearer Capabilities
Firmware Revision
Vendor Name
Receive Window Size
Challenge
Challenge Response

6.3 Start-Control-Connection-Connected (SCCCN)

Start-Control-Connection-Connected (SCCCN) is a control message sent in reply to an SCCRP. SCCCN completes the tunnel establishment process.

The following AVP MUST be present in the SCCCN:

Message Type

The following AVP MAY be present in the SCCCN:

Challenge Response

6.4 Stop-Control-Connection-Notification (StopCCN)

Stop-Control-Connection-Notification (StopCCN) is a control message sent by either the LAC or LNS to inform its peer that the tunnel is being shutdown and the control connection should be closed. In addition, all active sessions are implicitly cleared (without sending any explicit call control messages). The reason for issuing this request is indicated in the Result Code AVP. There is no explicit reply to the message, only the implicit ACK that is received by the reliable control message transport layer.

The following AVPs MUST be present in the StopCCN:

Message Type
Assigned Tunnel ID
Result Code

6.5 Hello (HELLO)

The Hello (HELLO) message is an L2TP control message sent by either peer of a LAC-LNS control connection. This control message is used as a "keepalive" for the tunnel.

The sending of HELLO messages and the policy for sending them are left up to the implementation. A peer MUST NOT expect HELLO messages at any time or interval. As with all messages sent on the control connection, the receiver will return either a ZLB ACK or an (unrelated) message piggybacking the necessary acknowledgement information.

Since a HELLO is a control message, and control messages are reliably sent by the lower level transport, this keepalive function operates by causing the transport level to reliably deliver a message. If a media interruption has occurred, the reliable transport will be unable to deliver the HELLO across, and will clean up the tunnel.

Keepalives for the tunnel MAY be implemented by sending a HELLO if a period of time (a recommended default is 60 seconds, but SHOULD be configurable) has passed without receiving any message (data or control) from the peer.

HELLO messages are global to the tunnel. The Session ID in a HELLO message MUST be 0.

The Following AVP MUST be present in the HELLO message:

Message Type

6.6 Incoming-Call-Request (ICRQ)

Incoming-Call-Request (ICRQ) is a control message sent by the LAC to the LNS when an incoming call is detected. It is the first in a three message exchange used for establishing a session within an L2TP tunnel.

ICRQ is used to indicate that a session is to be established between the LAC and LNS for this call and provides the LNS with parameter information for the session. The LAC may defer answering the call until it has received an ICRP from the LNS indicating that the session should be established. This mechanism allows the LNS to obtain sufficient information about the call before determining whether it should be answered or not. Alternatively, the LAC may answer the call, negotiate LCP and PPP authentication, and use the information gained to choose the LNS. In this case, the call has already been answered by the time the ICRP message is received; the LAC simply spoofs the "call indication" and "call answer" steps in this case.

The following AVPs MUST be present in the ICRQ:

- Message Type
- Assigned Session ID
- Call Serial Number

The following AVPs MAY be present in the ICRQ:

- Bearer Type
- Physical Channel ID
- Calling Number
- Called Number
- Sub-Address

6.7 Incoming-Call-Reply (ICRP)

Incoming-Call-Reply (ICRP) is a control message sent by the LNS to the LAC in response to a received ICRQ message. It is the second in the three message exchange used for establishing sessions within an L2TP tunnel.

ICRP is used to indicate that the ICRQ was successful and for the LAC to answer the call if it has not already done so. It also allows the LNS to indicate necessary parameters for the L2TP session.

The following AVPs MUST be present in the ICRP:

- Message Type
- Assigned Session ID

6.8 Incoming-Call-Connected (ICCN)

Incoming-Call-Connected (ICCN) is a control message sent by the LAC to the LNS in response to a received ICRP message. It is the third message in the three message exchange used for establishing sessions within an L2TP tunnel.

ICCN is used to indicate that the ICRP was accepted, the call has been answered, and that the L2TP session should move to the established state. It also provides additional information to the LNS about parameters used for the answered call (parameters that may not always available at the time the ICRQ is issued).

The following AVPs MUST be present in the ICCN:

- Message Type
- (Tx) Connect Speed
- Framing Type

The following AVPs MAY be present in the ICCN:

- Initial Received LCP CONFREQ
- Last Sent LCP CONFREQ
- Last Received LCP CONFREQ
- Proxy Authen Type
- Proxy Authen Name
- Proxy Authen Challenge
- Proxy Authen ID
- Proxy Authen Response
- Private Group ID
- Rx Connect Speed
- Sequencing Required

6.9 Outgoing-Call-Request (OCRQ)

Outgoing-Call-Request (OCRQ) is a control message sent by the LNS to the LAC to indicate that an outbound call from the LAC is to be established. It is the first in a three message exchange used for establishing a session within an L2TP tunnel.

OCRQ is used to indicate that a session is to be established between the LNS and LAC for this call and provides the LAC with parameter information for both the L2TP session, and the call that is to be placed

An LNS MUST have received a Bearer Capabilities AVP during tunnel establishment from an LAC in order to request an outgoing call to that LAC.

The following AVPs MUST be present in the OCRQ:

- Message Type
- Assigned Session ID
- Call Serial Number
- Minimum BPS
- Maximum BPS
- Bearer Type
- Framing Type
- Called Number

The following AVPs MAY be present in the OCRQ:

- Sub-Address

6.10 Outgoing-Call-Reply (OCRP)

Outgoing-Call-Reply (OCRP) is a control message sent by the LAC to the LNS in response to a received OCRQ message. It is the second in a three message exchange used for establishing a session within an L2TP tunnel.

OCRP is used to indicate that the LAC is able to attempt the outbound call and returns certain parameters regarding the call attempt.

The following AVPs MUST be present in the OCRP:

- Message Type
- Assigned Session ID

The following AVPs MAY be present in the OCRP:

- Physical Channel ID

6.11 Outgoing-Call-Connected (OCCN)

Outgoing-Call-Connected (OCCN) is a control message sent by the LAC to the LNS following the OCRP and after the outgoing call has been completed. It is the final message in a three message exchange used for establishing a session within an L2TP tunnel.

OCCN is used to indicate that the result of a requested outgoing call was successful. It also provides information to the LNS about the particular parameters obtained after the call was established.

The following AVPs MUST be present in the OCCN:

- Message Type
- (Tx) Connect Speed
- Framing Type

The following AVPs MAY be present in the OCCN:

- Rx Connect Speed
- Sequencing Required

6.12 Call-Disconnect-Notify (CDN)

The Call-Disconnect-Notify (CDN) message is an L2TP control message sent by either the LAC or LNS to request disconnection of a specific call within the tunnel. Its purpose is to inform the peer of the

disconnection and the reason why the disconnection occurred. The peer MUST clean up any resources, and does not send back any indication of success or failure for such cleanup.

The following AVPs MUST be present in the CDN:

- Message Type
- Result Code
- Assigned Session ID

The following AVPs MAY be present in the CDN:

- Q.931 Cause Code

6.13 WAN-Error-Notify (WEN)

The WAN-Error-Notify message is an L2TP control message sent by the LAC to the LNS to indicate WAN error conditions (conditions that occur on the interface supporting PPP). The counters in this message are cumulative. This message should only be sent when an error occurs, and not more than once every 60 seconds. The counters are reset when a new call is established.

The following AVPs MUST be present in the WEN:

Message Type
Call Errors

6.14 Set-Link-Info (SLI)

The Set-Link-Info message is an L2TP control message sent by the LNS to the LAC to set PPP-negotiated options. These options can change at any time during the life of the call, thus the LAC MUST be able to update its internal call information and behavior on an active PPP session.

The following AVPs MUST be present in the SLI:

Message Type
ACCM

7.0 Control Connection State Machines

The control messages defined in section 6 are exchanged by way of state tables defined in this section. Tables are defined for incoming call placement, outgoing call placement, as well as for initiation of

the tunnel itself. The state tables do not encode timeout and retransmission behavior, as this is handled in the underlying semantics defined in Section 5.8.

7.1 Control Connection Protocol Operation

This section describes the operation of various L2TP control connection functions and the Control Connection messages which are used to support them.

Receipt of an invalid or unrecoverable malformed control message should be logged appropriately and the control connection cleared to ensure recovery to a known state. The control connection may then be restarted by the initiator.

An invalid control message is defined as a message which contains a Message Type that is marked mandatory (see Section 4.4.1) and is unknown to the implementation, or a control message that is received in an improper sequence (e.g. an SCCCN sent in reply to an SCCRQ).

Examples of a malformed control message include one that has an invalid value in its header, contains an AVP that is formatted incorrectly or whose value is out of range, or a message that is missing a required AVP. A control message with a malformed header

should be discarded. A control message with an invalid AVP should look to the M-bit for that AVP to determine whether the error is recoverable or not.

A malformed yet recoverable non-mandatory (M-bit is not set) AVP within a control message should be treated in a similar manner as an unrecognized non-mandatory AVP. Thus, if a malformed AVP is received with the M-bit set, the session or tunnel should be terminated with a proper Result or Error Code sent. If the M-bit is not set, the AVP should be ignored (with the exception of logging a local error message) and the message accepted.

This MUST NOT be considered a license to send malformed AVPs, but simply a guide towards how to handle an improperly formatted message if one is received. It is impossible to list all potential malformations of a given message and give advice for each. That said, one example of a recoverable, malformed AVP might be if the Rx Connect Speed AVP, attribute 38, is received with a length of 8 rather than 10 and the BPS given in 2 octets rather than 4. Since the Rx Connect Speed is non-mandatory, this condition should not be considered catastrophic. As such, the control message should be accepted as if the AVP had not been received (with the exception of a local error message being logged).

In several cases in the following tables, a protocol message is sent, and then a "clean up" occurs. Note that regardless of the initiator of the tunnel destruction, the reliable delivery mechanism must be allowed to run (see Section 5.8) before destroying the tunnel. This permits the tunnel management messages to be reliably delivered to the peer.

Appendix B.1 contains an example of lock-step tunnel establishment.

7.2 Control Connection States

The L2TP control connection protocol is not distinguishable between the LNS and LAC, but is distinguishable between the originator and receiver. The originating peer is the one which first initiates establishment of the tunnel (in a tie breaker situation, this is the winner of the tie). Since either LAC or LNS can be the originator, a collision can occur. See the Tie Breaker AVP in Section 4.4.3 for a description of this and its resolution.

7.2.1 Control Connection Establishment

State	Event	Action	New State
idle	Local Open request	Send SCCRQ	wait-ctl-reply
idle	Receive SCCRQ, acceptable	Send SCCRP	wait-ctl-conn
idle	Receive SCCRQ, not acceptable	Send StopCCN, Clean up	idle
idle	Receive SCCRP	Send StopCCN	idle

		Clean up	
idle	Receive SCCCN	Clean up	idle
wait-ctl-reply	Receive SCCRP, acceptable	Send SCCCN, Send tunnel-open event to waiting sessions	established
wait-ctl-reply	Receive SCCRP, not acceptable	Send StopCCN, Clean up	idle
wait-ctl-reply	Receive SCCRQ, lose tie-breaker	Clean up, Re-queue SCCRQ for idle state	idle

wait-ctl-reply	Receive SCCCN	Send StopCCN Clean up	idle
wait-ctl-conn	Receive SCCCN, acceptable	Send tunnel-open event to waiting sessions	established
wait-ctl-conn	Receive SCCCN, not acceptable	Send StopCCN, Clean up	idle
wait-ctl-conn	Receive SCCRP, SCCRQ	Send StopCCN, Clean up	idle
established	Local Open request (new call)	Send tunnel-open event to waiting sessions	established
established	Admin Tunnel Close	Send StopCCN Clean up	idle
established	Receive SCCRQ, SCCRP, SCCCN	Send StopCCN Clean up	idle
idle wait-ctl-reply, wait-ctl-conn, established	Receive StopCCN	Clean up	idle

The states associated with the LNS or LAC for control connection establishment are:

idle
: Both initiator and recipient start from this state. An initiator transmits an SCCRQ, while a recipient remains in the idle state until receiving an SCCRQ.

wait-ctl-reply
: The originator checks to see if another connection has been requested from the same peer, and if so, handles the collision situation described in Section 5.8.

  When an SCCRP is received, it is examined for a compatible

Page 1323
T-Mobile Exhibit 1002
Part 4 of 6

version. If the version of the reply is lower than the version sent in the request, the older (lower) version should be used provided it is supported. If the version in the reply is earlier and supported, the originator moves to the established state. If

the version is earlier and not supported, a StopCCN MUST be sent to the peer and the originator cleans up and terminates the tunnel.

wait-ctl-conn
This is where an SCCCN is awaited; upon receipt, the challenge response is checked. The tunnel either is established, or is torn down if an authorization failure is detected.

established
An established connection may be terminated by either a local condition or the receipt of a Stop-Control-Connection-Notification. In the event of a local termination, the originator MUST send a Stop-Control-Connection-Notification and clean up the tunnel.

If the originator receives a Stop-Control-Connection-Notification it MUST also clean up the tunnel.

7.3 Timing considerations

Due to the real-time nature of telephone signaling, both the LNS and LAC should be implemented with multi-threaded architectures such that messages related to multiple calls are not serialized and blocked. The call and connection state figures do not specify exceptions caused by timers. These are addressed in Section 5.8.

7.4 Incoming calls

An Incoming-Call-Request message is generated by the LAC when an incoming call is detected (for example, an associated telephone line rings). The LAC selects a Session ID and serial number and indicates the call bearer type. Modems should always indicate analog call type. ISDN calls should indicate digital when unrestricted digital service or rate adaption is used and analog if digital modems are involved. Calling Number, Called Number, and Subaddress may be included in the message if they are available from the telephone network.

Once the LAC sends the Incoming-Call-Request, it waits for a response from the LNS but it does not necessarily answer the call from the telephone network yet. The LNS may choose not to accept the call if:

- No resources are available to handle more sessions
- The dialed, dialing, or subaddress fields do not correspond to an authorized user
- The bearer service is not authorized or supported

If the LNS chooses to accept the call, it responds with an Incoming-Call-Reply. When the LAC receives the Incoming-Call-Reply, it attempts to connect the call. A final call connected message from the LAC to the LNS indicates that the call states for both the LAC and the LNS should enter the established state. If the call terminated before the LNS could accept it, a Call-Disconnect-Notify is sent by the LAC to indicate this condition.

When the dialed-in client hangs up, the call is cleared normally and the LAC sends a Call-Disconnect-Notify message. If the LNS wishes to clear a call, it sends a Call-Disconnect-Notify message and cleans up its session.

7.4.1 LAC Incoming Call States

State	Event	Action	New State

idle	Bearer Ring or Ready to indicate incoming conn.	Initiate local tunnel open	wait-tunnel
idle	Receive ICCN, ICRP, CDN	Clean up	idle
wait-tunnel	Bearer line drop or local close request	Clean up	idle
wait-tunnel	tunnel-open	Send ICRQ	wait-reply
wait-reply	Receive ICRP, acceptable	Send ICCN	established
wait-reply	Receive ICRP, Not acceptable	Send CDN, Clean up	idle
wait-reply	Receive ICRQ	Send CDN Clean up	idle
wait-reply	Receive CDN ICCN	Clean up	idle
wait-reply	Local close request or Bearer line drop	Send CDN, Clean up	idle
established	Receive CDN	Clean up	idle
established	Receive ICRQ, ICRP, ICCN	Send CDN, Clean up	idle
established	Bearer line drop or local close request	Send CDN, Clean up	idle

The states associated with the LAC for incoming calls are:

idle
The LAC detects an incoming call on one of its interfaces. Typically this means an analog line is ringing or an ISDN TE has detected an incoming Q.931 SETUP message. The LAC initiates its tunnel establishment state machine, and moves to a state waiting for confirmation of the existence of a tunnel.

wait-tunnel
In this state the session is waiting for either the control connection to be opened or for verification that the tunnel is already open. Once an indication that the tunnel has/was opened,

session control messages may be exchanged. The first of these is the Incoming-Call-Request.

wait-reply
The LAC receives either a CDN message indicating the LNS is not willing to accept the call (general error or don't accept) and moves back into the idle state, or an Incoming-Call-Reply message indicating the call is accepted, the LAC sends an Incoming-Call-Connected message and enters the established state.

established
Data is exchanged over the tunnel. The call may be cleared following:
+ An event on the connected interface: The LAC sends a Call-Disconnect-Notify message
+ Receipt of a Call-Disconnect-Notify message: The LAC cleans up, disconnecting the call.
+ A local reason: The LAC sends a Call-Disconnect-Notify message.

7.4.2 LNS Incoming Call States

State	Event	Action	New State
idle	Receive ICRQ, acceptable	Send ICRP	wait-connect
idle	Receive ICRQ, not acceptable	Send CDN, Clean up	idle
idle	Receive ICRP	Send CDN Clean up	idle
idle	Receive ICCN	Clean up	idle
wait-connect	Receive ICCN acceptable	Prepare for data	established
wait-connect	Receive ICCN not acceptable	Send CDN, Clean up	idle
wait-connect	Receive ICRQ,	Send CDN	idle

Page 1327
T-Mobile Exhibit 1002
Part 4 of 6

	ICRP	Clean up	
idle, wait-connect, established	Receive CDN	Clean up	idle
wait-connect	Local	Send CDN,	idle
established	Close request	Clean up	
established	Receive ICRQ, ICRP, ICCN	Send CDN Clean up	idle

The states associated with the LNS for incoming calls are:

idle
An Incoming-Call-Request message is received. If the request is not acceptable, a Call-Disconnect-Notify is sent back to the LAC and the LNS remains in the idle state. If the Incoming-Call-Request message is acceptable, an Incoming-Call-Reply is sent. The session moves to the wait-connect state.

wait-connect
If the session is still connected on the LAC, the LAC sends an Incoming-Call-Connected message to the LNS which then moves into established state. The LAC may send a Call-Disconnect-Notify to indicate that the incoming caller could not be connected. This

could happen, for example, if a telephone user accidentally places a standard voice call to an LAC resulting in a handshake failure on the called modem.

established
The session is terminated either by receipt of a Call-Disconnect-Notify message from the LAC or by sending a Call-Disconnect-Notify. Clean up follows on both sides regardless of the initiator.

7.5 Outgoing calls

Outgoing calls are initiated by an LNS and instruct an LAC to place a call. There are three messages for outgoing calls: Outgoing-Call-Request, Outgoing-Call-Reply, and Outgoing-Call-Connected. The LNS sends an Outgoing-Call-Request specifying the dialed party phone number, subaddress and other parameters. The LAC MUST respond to the Outgoing-Call-Request message with an Outgoing-Call-Reply message once the LAC determines that the proper facilities exist to place the call and the call is administratively authorized. For example, is this LNS allowed to dial an international call? Once the outbound call is connected, the LAC sends an Outgoing-Call-Connected message to the LNS indicating the final result of the call attempt:

7.5.1 LAC Outgoing Call States

State	Event	Action	New State
idle	Receive OCRQ, acceptable	Send OCRP, Open bearer	wait-cs-answer
idle	Receive OCRQ, not acceptable	Send CDN, Clean up	idle
idle	Receive OCRP	Send CDN Clean up	idle
idle	Receive OCCN, CDN	Clean up	idle
wait-cs-answer	Bearer answer, framing detected	Send OCCN	established
wait-cs-answer	Bearer failure	Send CDN, Clean up	idle
wait-cs-answer	Receive OCRQ, OCRP, OCCN	Send CDN Clean up	idle
established	Receive OCRQ, OCRP, OCCN	Send CDN Clean up	idle
wait-cs-answer, established	Receive CDN	Clean up	idle
established	Bearer line drop, Local close request	Send CDN, Clean up	idle

The states associated with the LAC for outgoing calls are:

idle
If Outgoing-Call-Request is received in error, respond with a Call-Disconnect-Notify. Otherwise, allocate a physical channel and

send an Outgoing-Call-Reply. Place the outbound call and move to the wait-cs-answer state.

wait-cs-answer
If the call is not completed or a timer expires waiting for the call to complete, send a Call-Disconnect-Notify with the appropriate error condition set and go to idle state. If a circuit

switched connection is established and framing is detected, send an Outgoing-Call-Connected indicating success and go to established state.

established
If a Call-Disconnect-Notify is received by the LAC, the telco call MUST be released via appropriate mechanisms and the session cleaned up. If the call is disconnected by the client or the called interface, a Call-Disconnect-Notify message MUST be sent to the LNS. The sender of the Call-Disconnect-Notify message returns to the idle state after sending of the message is complete.

7.5.2 LNS Outgoing Call States

State	Event	Action	New State
idle	Local open request	Initiate local tunnel-open	wait-tunnel
idle	Receive OCCN, OCRP, CDN	Clean up	idle
wait-tunnel	tunnel-open	Send OCRQ	wait-reply
wait-reply	Receive OCRP, acceptable	none	wait-connect
wait-reply	Receive OCRP, not acceptable	Send CDN Clean up	idle
wait-reply	Receive OCCN, OCRQ	Send CDN Clean up	idle
wait-connect	Receive OCCN	none	established
wait-connect	Receive OCRQ, OCRP	Send CDN Clean up	idle
idle, wait-reply, wait-connect, established	Receive CDN,	Clean up	idle
established	Receive OCRQ, OCRP, OCCN	Send CDN Clean up	idle
wait-reply, wait-connect, established	Local Close request	Send CDN Clean up	idle
wait-tunnel	Local Close request	Clean up	idle

The states associated with the LNS for outgoing calls are:

idle, wait-tunnel
   When an outgoing call is initiated, a tunnel is first created, much as the idle and wait-tunnel states for an LAC incoming call. Once a tunnel is established, an Outgoing-Call-Request message is sent to the LAC and the session moves into the wait-reply state.

wait-reply
   If a Call-Disconnect-Notify is received, an error occurred, and

the session is cleaned up and returns to idle. If an Outgoing-Call-Reply is received, the call is in progress and the session moves to the wait-connect state.

wait-connect
If a Call-Disconnect-Notify is received, the call failed; the session is cleaned up and returns to idle. If an Outgoing-Call-Connected is received, the call has succeeded and the session may now exchange data.

established
If a Call-Disconnect-Notify is received, the call has been terminated for the reason indicated in the Result and Cause Codes; the session moves back to the idle state. If the LNS chooses to terminate the session, it sends a Call-Disconnect-Notify to the LAC and then cleans up and idles its session.

7.6 Tunnel Disconnection

The disconnection of a tunnel consists of either peer issuing a Stop-Control-Connection-Notification. The sender of this Notification should wait a finite period of time for the acknowledgment of this message before releasing the control information associated with the tunnel. The recipient of this Notification should send an acknowledgment of the Notification and then release the associated control information.

When to release a tunnel is an implementation issue and is not specified in this document. A particular implementation may use whatever policy is appropriate for determining when to release a control connection. Some implementations may leave a tunnel open for a period of time or perhaps indefinitely after the last session for that tunnel is cleared. Others may choose to disconnect the tunnel immediately after the last user connection on the tunnel disconnects.

8.0 L2TP Over Specific Media

L2TP is self-describing, operating at a level above the media over which it is carried. However, some details of its connection to media are required to permit interoperable implementations. The following sections describe details needed to permit interoperability over specific media.

8.1 L2TP over UDP/IP

L2TP uses the registered UDP port 1701 [RFC1700]. The entire L2TP packet, including payload and L2TP header, is sent within a UDP datagram. The initiator of an L2TP tunnel picks an available source UDP port (which may or may not be 1701), and sends to the desired destination address at port 1701. The recipient picks a free port on its own system (which may or may not be 1701), and sends its reply to the initiator's UDP port and address, setting its own source port to the free port it found. Once the source and destination ports and addresses are established, they MUST remain static for the life of

the tunnel.

It has been suggested that having the recipient choose an arbitrary source port (as opposed to using the destination port in the packet initiating the tunnel, i.e., 1701) may make it more difficult for L2TP to traverse some NAT devices. Implementors should consider the potential implication of this before before choosing an arbitrary source port.

IP fragmentation may occur as the L2TP packet travels over the IP substrate. L2TP makes no special efforts to optimize this. A LAC implementation MAY cause its LCP to negotiate for a specific MRU, which could optimize for LAC environments in which the MTU's of the path over which the L2TP packets are likely to travel have a consistent value.

The default for any L2TP implementation is that UDP checksums MUST be enabled for both control and data messages. An L2TP implementation MAY provide an option to disable UDP checksums for data messages. It is recommended that UDP checksums always be enabled on control packets.

Port 1701 is used for both L2F [RFC2341] and L2TP packets. The Version field in each header may be used to discriminate between the two packet types (L2F uses a value of 1, and the L2TP version described in this document uses a value of 2). An L2TP implementation running on a system which does not support L2F MUST silently discard all L2F packets.

To the PPP clients using an L2TP-over-UDP/IP tunnel, the PPP link has the characteristic of being able to reorder or silently drop packets. The former may break non-IP protocols being carried by PPP, especially LAN-centric ones such as bridging. The latter may break protocols which assume per-packet indication of error, such as TCP header compression. Sequencing may be handled by using L2TP data message sequence numbers if any protocol being transported by the PPP

tunnel cannot tolerate reordering. The sequence dependency characteristics of individual protocols are outside the scope of this document.

Allowing packets to be dropped silently is perhaps more problematic with some protocols. If PPP reliable delivery [RFC1663] is enabled, no upper PPP protocol will encounter lost packets. If L2TP sequence numbers are enabled, L2TP can detect the packet loss. In the case of an LNS, the PPP and L2TP stacks are both present within the LNS, and packet loss signaling may occur precisely as if a packet was received with a CRC error. Where the LAC and PPP stack are co-resident, this technique also applies. Where the LAC and PPP client are physically distinct, the analogous signaling MAY be accomplished by sending a packet with a CRC error to the PPP client. Note that this would greatly increase the complexity of debugging client line problems, since the client statistics could not distinguish between true media errors and LAC-initiated ones. Further, this technique is not possible on all hardware.

If VJ compression is used, and neither PPP reliable delivery nor

sequence numbers are enabled, each lost packet results in a 1 in 2**16 chance of a TCP segment being forwarded with incorrect contents [RFC1144]. Where the combination of the packet loss rate with this statistical exposure is unacceptable, TCP header compression SHOULD NOT be used.

In general, it is wise to remember that the L2TP/UDP/IP transport is an unreliable transport. As with any PPP media that is subject to loss, care should be taken when using protocols that are particularly loss-sensitive. Such protocols include compression and encryption protocols that employ history.

8.2 IP

When operating in IP environments, L2TP MUST offer the UDP encapsulation described in 8.1 as its default configuration for IP operation. Other configurations (perhaps corresponding to a compressed header format) MAY be defined and made available as a configurable option.

9.0 Security Considerations

L2TP encounters several security issues in its operation. The general approach of L2TP to these issues is documented here.

9.1 Tunnel Endpoint Security

The tunnel endpoints may optionally perform an authentication procedure of one another during tunnel establishment. This authentication has the same security attributes as CHAP, and has reasonable protection against replay and snooping during the tunnel establishment process. This mechanism is not designed to provide any authentication beyond tunnel establishment; it is fairly simple for a malicious user who can snoop the tunnel stream to inject packets once an authenticated tunnel establishment has been completed successfully.

For authentication to occur, the LAC and LNS MUST share a single secret. Each side uses this same secret when acting as authenticatee as well as authenticator. Since a single secret is used, the tunnel authentication AVPs include differentiating values in the CHAP ID fields for each message digest calculation to guard against replay attacks.

The Assigned Tunnel ID and Assigned Session ID (See Section 4.4.3) SHOULD be selected in an unpredictable manner rather than sequentially or otherwise. Doing so will help deter hijacking of a session by a malicious user who does not have access to packet traces between the LAC and LNS.

9.2 Packet Level Security

Securing L2TP requires that the underlying transport make available encryption, integrity and authentication services for all L2TP

traffic. This secure transport operates on the entire L2TP packet and is functionally independent of PPP and the protocol being carried by PPP. As such, L2TP is only concerned with confidentiality, authenticity, and integrity of the L2TP packets between its tunnel endpoints (the LAC and LNS), not unlike link-layer encryption being concerned only about protecting the confidentiality of traffic between its physical endpoints.

9.3 End to End Security

Protecting the L2TP packet stream via a secure transport does, in turn, also protect the data within the tunneled PPP packets while transported from the LAC to the LNS. Such protection should not be considered a substitution for end-to-end security between communicating hosts or applications.

9.4 L2TP and IPsec

When running over IP, IPsec provides packet-level security via ESP and/or AH. All L2TP control and data packets for a particular tunnel appear as homogeneous UDP/IP data packets to the IPsec system.

In addition to IP transport security, IPsec defines a mode of operation that allows tunneling of IP packets. The packet level encryption and authentication provided by IPsec tunnel mode and that provided by L2TP secured with IPsec provide an equivalent level of security for these requirements.

IPsec also defines access control features that are required of a compliant IPsec implementation. These features allow filtering of packets based upon network and transport layer characteristics such as IP address, ports, etc. In the L2TP tunneling model, analogous filtering is logically performed at the PPP layer or network layer above L2TP. These network layer access control features may be handled at the LNS via vendor-specific authorization features based upon the authenticated PPP user, or at the network layer itself by using IPsec transport mode end-to-end between the communicating hosts. The requirements for access control mechanisms are not a part of the L2TP specification and as such are outside the scope of this document.

9.5 Proxy PPP Authentication

L2TP defines AVPs that MAY be exchanged during session establishment to provide forwarding of PPP authentication information obtained at the LAC to the LNS for validation (see Section 4.4.5). This implies a direct trust relationship of the LAC on behalf of the LNS. If the LNS chooses to implement proxy authentication, it MUST be able to be configured off, requiring a new round a PPP authentication initiated by the LNS (which may or may not include a new round of LCP negotiation).

10.0 IANA Considerations

This document defines a number of "magic" numbers to be maintained by the IANA. This section explains the criteria to be used by the IANA to assign additional numbers in each of these lists. The following subsections describe the assignment policy for the namespaces defined elsewhere in this document.

10.1 AVP Attributes

As defined in Section 4.1, AVPs contain vendor ID, Attribute and Value fields. For vendor ID value of 0, IANA will maintain a registry

of assigned Attributes and in some case also values. Attributes 0-39 are assigned as defined in Section 4.4. The remaining values are available for assignment through IETF Consensus [RFC 2434].

10.2 Message Type AVP Values

As defined in Section 4.4.1, Message Type AVPs (Attribute Type 0) have an associated value maintained by IANA. Values 0-16 are defined in Section 3.2, the remaining values are available for assignment via IETF Consensus [RFC 2434]

10.3 Result Code AVP Values

As defined in Section 4.4.2, Result Code AVPs (Attribute Type 1) contain three fields. Two of these fields (the Result Code and Error Code fields) have associated values maintained by IANA.

10.3.1 Result Code Field Values

The Result Code AVP may be included in CDN and StopCCN messages. The allowable values for the Result Code field of the AVP differ depending upon the value of the Message Type AVP. For the StopCCN message, values 0-7 are defined in Section 4.4.2; for the StopCCN message, values 0-11 are defined in the same section. The remaining values of the Result Code field for both messages are available for assignment via IETF Consensus [RFC 2434].

10.3.2 Error Code Field Values

Values 0-7 are defined in Section 4.4.2. Values 8-32767 are available for assignment via IETF Consensus [RFC 2434]. The remaining values of the Error Code field are available for assignment via First Come First Served [RFC 2434].

10.4 Framing Capabilities & Bearer Capabilities

The Framing Capabilities AVP and Bearer Capabilities AVPs (defined in Section 4.4.3) both contain 32-bit bitmasks. Additional bits should only be defined via a Standards Action [RFC 2434].

10.5 Proxy Authen Type AVP Values

The Proxy Authen Type AVP (Attribute Type 29) has an associated value maintained by IANA. Values 0-5 are defined in Section 4.4.5, the remaining values are available for assignment via First Come First Served [RFC 2434].

10.6 AVP Header Bits

There are four remaining reserved bits in the AVP header. Additional bits should only be assigned via a Standards Action [RFC 2434].

11.0 References

[DSS1] ITU-T Recommendation, "Digital subscriber Signaling System No. 1 (DSS 1) - ISDN user-network interface layer 3 specification for basic call control", Rec. Q.931(I.451), May 1998

[KPS] Kaufman, C., Perlman, R., and Speciner, M., "Network Security: Private Communications in a Public World", Prentice Hall, March 1995, ISBN 0-13-061466-1

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987.

[RFC1144] Jacobson, V., "Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, February 1990.

[RFC1661] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC1662] Simpson, W., "PPP in HDLC-like Framing", STD 51, RFC 1662, July 1994.

[RFC1663] Rand, D., "PPP Reliable Transmission", RFC 1663, July 1994.

[RFC1700] Reynolds, J. and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994. See also: http://www.iana.org/numbers.html

[RFC1990] Sklower, K., Lloyd, B., McGregor, G., Carr, D. and T. Coradetti, "The PPP Multilink Protocol (MP)", RFC 1990, August 1996.

[RFC1994] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2138] Rigney, C., Rubens, A., Simpson, W. and S. Willens, "Remote Authentication Dial In User Service (RADIUS)", RFC 2138, April 1997.

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC2341] Valencia, A., Littlewood, M. and T. Kolar, "Cisco Layer Two Forwarding (Protocol) L2F", RFC 2341, May 1998.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W. and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, July 1999.

[STEVENS] Stevens, W. Richard, "TCP/IP Illustrated, Volume I The Protocols", Addison-Wesley Publishing Company, Inc., March 1996, ISBN 0-201-63346-9

12.0 Acknowledgments

The basic concept for L2TP and many of its protocol constructs were adopted from L2F [RFC2341] and PPTP [PPTP]. Authors of these are A. Valencia, M. Littlewood, T. Kolar, K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn.

Dory Leifer made valuable refinements to the protocol definition of L2TP and contributed to the editing of this document.

Steve Cobb and Evan Caves redesigned the state machine tables.

Barney Wolff provided a great deal of design input on the endpoint authentication mechanism.

John Bray, Greg Burns, Rich Garrett, Don Grosser, Matt Holdrege, Terry Johnson, Dory Leifer, and Rich Shea provided valuable input and review at the 43rd IETF in Orlando, FL., which led to improvement of the overall readability and clarity of this document.

13.0 Authors' Addresses

Gurdeep Singh Pall
Microsoft Corporation
Redmond, WA

EMail: gurdeep@microsoft.com

Bill Palter
RedBack Networks, Inc
1389 Moffett Park Drive
Sunnyvale, CA 94089

EMail: palter@zev.net

Allan Rubens
Ascend Communications
1701 Harbor Bay Parkway
Alameda, CA 94502

EMail: acr@del.com

W. Mark Townsley
cisco Systems
7025 Kit Creek Road
PO Box 14987
Research Triangle Park, NC 27709

EMail: townsley@cisco.com

Andrew J. Valencia
cisco Systems
170 West Tasman Drive
San Jose CA 95134-1706

EMail: vandys@cisco.com

Glen Zorn
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

EMail: gwz@acm.org

Appendix A: Control Channel Slow Start and Congestion Avoidance

Although each side has indicated the maximum size of its receive window, it is recommended that a slow start and congestion avoidance method be used to transmit control packets. The methods described here are based upon the TCP congestion avoidance algorithm as described in section 21.6 of TCP/IP Illustrated, Volume I, by W. Richard Stevens [STEVENS].

Slow start and congestion avoidance make use of several variables. The congestion window (CWND) defines the number of packets a sender may send before waiting for an acknowledgment. The size of CWND expands and contracts as described below. Note however, that CWND is never allowed to exceed the size of the advertised window obtained from the Receive Window AVP (in the text below, it is assumed any increase will be limited by the Receive Window Size). The variable SSTHRESH determines when the sender switches from slow start to congestion avoidance. Slow start is used while CWND is less than

SSHTRESH.

A sender starts out in the slow start phase. CWND is initialized to one packet, and SSHTRESH is initialized to the advertised window (obtained from the Receive Window AVP). The sender then transmits one packet and waits for its acknowledgement (either explicit or piggybacked). When the acknowledgement is received, the congestion window is incremented from one to two. During slow start, CWND is increased by one packet each time an ACK (explicit ZLB or piggybacked) is received. Increasing CWND by one on each ACK has the effect of doubling CWND with each round trip, resulting in an exponential increase. When the value of CWND reaches SSHTRESH, the slow start phase ends and the congestion avoidance phase begins.

During congestion avoidance, CWND expands more slowly. Specifically, it increases by 1/CWND for every new ACK received. That is, CWND is increased by one packet after CWND new ACKs have been received. Window expansion during the congestion avoidance phase is effectively linear, with CWND increasing by one packet each round trip.

When congestion occurs (indicated by the triggering of a retransmission) one half of the CWND is saved in SSTHRESH, and CWND is set to one. The sender then reenters the slow start phase.

Appendix B: Control Message Examples

B.1: Lock-step tunnel establishment

In this example, an LAC establishes a tunnel, with the exchange involving each side alternating in sending messages. This example shows the final acknowledgment explicitly sent within a ZLB ACK message. An alternative would be to piggyback the acknowledgement within a message sent as a reply to the ICRQ or OCRQ that will likely follow from the side that initiated the tunnel.

```
       LAC or LNS                LNS or LAC
       ----------                ----------

       SCCRQ     ->
       Nr: 0, Ns: 0
                                 <-     SCCRP
                                 Nr: 1, Ns: 0
       SCCCN     ->
       Nr: 1, Ns: 1
                                 <-       ZLB
                                 Nr: 2, Ns: 1
```

B.2: Lost packet with retransmission

An existing tunnel has a new session requested by the LAC. The ICRP is lost and must be retransmitted by the LNS. Note that loss of the

ICRP has two impacts: not only does it keep the upper level state machine from progressing, but it also keeps the LAC from seeing a timely lower level acknowledgment of its ICRQ.

```
         LAC                                  LNS
         ---                                  ---

     ICRQ      ->
     Nr: 1, Ns: 2

                       (packet lost)   <-      ICRP
                                       Nr: 3, Ns: 1

   (pause; LAC's timer started first, so fires first)

     ICRQ      ->
     Nr: 1, Ns: 2

     (Realizing that it has already seen this packet,
     the LNS discards the packet and sends a ZLB)
```



```
                                       <-       ZLB
                                       Nr: 3, Ns: 2

                     (LNS's retransmit timer fires)

                                       <-      ICRP
                                       Nr: 3, Ns: 1
     ICCN      ->
     Nr: 2, Ns: 3

                                       <-       ZLB
                                       Nr: 4, Ns: 2
```

Appendix C: Intellectual Property Notice

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF Secretariat."

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

Full Copyright Statement

Copyright (C) The Internet Society (1999). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

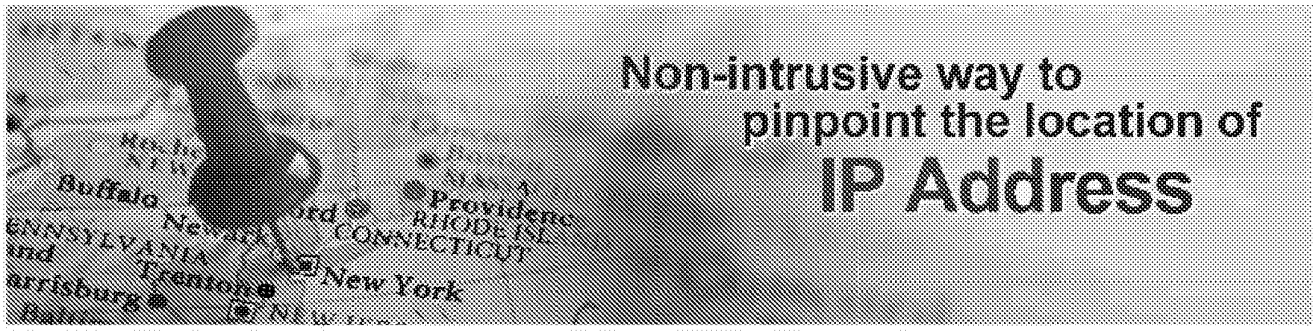
Funding for the RFC Editor function is currently provided by the Internet Society.

IP2LOCATION™

Bringing Location to the Internet

64.26.149.164

Home | Buy Online | Demo | Developers | Contact | Login My Account



Need to know which country, state or city your internet visitors are from? How to get their ISP and domain information? Or check their connection type, ZIP code, area code, weather station or perhaps their mobile carrier information?

0 item selected

$0.00

View Cart | Checkout

IP2Location™ is a non-intrusive geo IP solution to help you to identify visitor's geographical location, i.e. country, region, city, latitude, longitude, ZIP code, time zone, connection speed, ISP and domain name, IDD country code, area code, weather station code and name, and mobile carrier information using a proprietary IP address lookup database and technology without invading the Internet user's privacy. The solution is available as database, programming API and hosted solution. Request your free copy of the demo database today.

Product Picker

Pick a product on-demand.

	Field Name	Value
	IP Address	64.26.149.164
☒	Country	CANADA
☐	Region & City	ONTARIO, OTTAWA
☐	Latitude & Longitude	45.41117, -75.69812
☐	ZIP Code	K1P 6L5
☐	ISP	FETHERSTONHAUGH L.P.
☐	Domain	-
☐	Time Zone	-04:00
☐	Net Speed	COMP
☐	Area Code & IDD Code	+1

IP2LOCATION DATABASES

- DB1 IP-Country Database
- DB2 IP-Country-ISP Database
- DB3 IP-Country-Region-City Database
- DB4 IP-Country-Region-City-ISP Database
- DB5 IP-Country-Region-City-Latitude-Longitude Database
- DB6 IP-Country-Region-City-Latitude-Longitude-ISP Database
- DB7 IP-Country-Region-City-ISP-Domain Database
- DB8 IP-Country-Region-City-Latitude-Longitude-ISP-Domain Database
- DB9 IP-Country-Region-City-Latitude-Longitude-ZIPCode Database
- DB10 IP-Country-Region-City-Latitude-Longitude-ZIPCode-ISP-Domain Database
- DB11 IP-Country-Region-City-Latitude-Longitude-ZIPCode-TimeZone Database
- DB12 IP-Country-Region-City-Latitude-Longitude-ZIPCode-TimeZone-ISP-Domain Database
- DB13 IP-Country-Region-City-Latitude-Longitude-TimeZone-NetSpeed Database
- DB14 IP-Country-Region-City-Latitude-Longitude-ZIPCode-TimeZone-ISP-Domain-NetSpeed Database
- DB15 IP-Country-Region-City-Latitude-Longitude-ZIPCode-TimeZone-AreaCode Database

	Field Name	Value
☐	Weather Station	OTTAWA (CAXX0343)
☐	Mobile MCC, MNC, and Carrier name	-

Recommended Product:
[DB1] IP-Country Database

DB16 IP-Country-Region-City-Latitude-Longitude-ZIPCode-TimeZone-ISP-Domain-NetSpeed-AreaCode Database

DB17 IP-Country-Region-City-Latitude-Longitude-ZIPCode-TimeZone-NetSpeed-Weather Database

DB18 IP-Country-Region-City-Latitude-Longitude-ZIPCode-TimeZone-ISP-Domain-NetSpeed-AreaCode-Weather Database

DB19 IP-Country-Region-City-Latitude-Longitude-ISP-Domain-Mobile Database

DB20 IP-Country-Region-City-Latitude-Longitude-ZIPCode-TimeZone-ISP-Domain-NetSpeed-AreaCode-Weather-Mobile Database

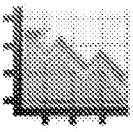
Developing Applications using IP2Location™



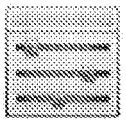
- Display native language and currency
- Redirect web pages based on visitor's location
- Digital rights management (DRM)



- Detect and prevent password sharing or abuse of service
- Detect and reduce credit card fraud



- Web traffics analysis
- Auto-selection of country on forms
- Filter access from countries you do not do business with



- Geo targeting for increased sales and click-through
- Spam filtering by location

Copyright © 2001-2012 IP2Location.com. All Rights Reserved.

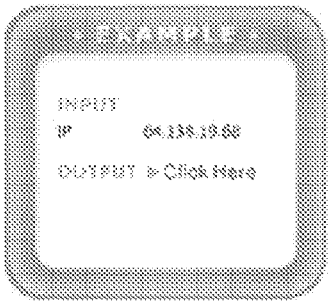
Home » Products » IP Address

DOTS IP Address Validation

Verify and Geo Target Website Visitors and Help Prevent Fraud

DOTS IP Address Validation reveals key pieces of information about your site visitors, such as: geographic location, the name of the Internet Service Provider (ISP) being used, and whether or not the IP address is going through a proxy server.

FREE TRIAL 30-DAY USE



Overview | Features & Benefits | Security | Pricing | Developer's Tools

What is it?

DOTS IP Address Validation (IPV) is a programmable XML Web Service hosted by Service Objects that allows businesses to integrate IP-based location information, distance calculation, and other geographic intelligence into Web-enabled applications and business processes. It offers wide-area coverage for Africa, Antarctica, Asia, Australia, Europe, North America, and South America.

Why use DOTS IP Validation?

The combination of multi-source triangulation of location, and outstanding anonymous proxy detection, make DOTS IP Validation among the top geolocation solutions.

What does it do?

DOTS IP Validation is the only geolocation service that calculates an IP's locations by triangulation of multiple data sources in real time. This technique yields outstanding coverage and leading accuracy for IP addresses worldwide.

In addition, the service collects anonymous proxy data from 165,000 computers, scattered across the world, acting as listening posts. These computers provide insight to other computers that are participating in malicious behavior.

©2001-2012 Service Objects, Inc. | 133 East De La Guerra St., Suite 10 | Santa Barbara, CA 93101 | 1.805.963.1700 | toll free: 1.8

List of North American Numbering Plan area codes

From Wikipedia, the free encyclopedia
(Redirected from List of NANP area codes)

This is a **list of North American telephone area codes** in effect for the North American Numbering Plan (NANP). The area to which an area code is officially assigned is known as a Numbering Plan Area (NPA).

Contents

- 1 200-299
- 2 300-399
- 3 400-499
- 4 500-599
- 5 600-699
- 6 700-799
- 7 800-899
- 8 900-999
- 9 By state, province or country
  - 9.1 United States
  - 9.2 Canada
  - 9.3 Caribbean and Bermuda
  - 9.4 U.S. Pacific Territories
  - 9.5 NANP non-geographic
- 10 See also
- 11 References

200-299

200: not used

201: New Jersey (Hackensack, Jersey City, Hoboken, Bayonne, Ridgewood, Union City, Teaneck, New Milford, and most of northeastern New Jersey)

- Originally covered all of New Jersey. Split to create **609** (1958), **908** (1991) and **973** (1997).
- Overlain by **551** in 2001.

202: Washington, D.C. (all)

203: Connecticut (Bridgeport, Danbury, New Haven, Waterbury, and southwestern Connecticut)

- Originally covered all of Connecticut; split in 1995 to create **860**.
- Overlain by **475** in December 2009.

204: Manitoba (all). 431 will overlay same area beginning November 2012.

205: Alabama (Birmingham; Tuscaloosa, and parts of western and central Alabama)

- Originally covered all of Alabama; split to create **334** (1995) and **256** (1998).

206: State of Washington (Seattle, all of Bainbridge, Mercer, and Vashon islands, Burien, Des Moines, Lake Forest Park, Normandy Park, Sea-Tac, Shoreline, Tukwila, and some small unincorporated areas adjacent to these. Also, parts of Woodway and Edmonds)

- Originally covered all of Washington; split to create **509** (1957), **360** (1995), **425** and **253** (1997).

207: Maine (all except Estcourt Station)

208: Idaho (all)

209: California (Stockton, Modesto, Merced, Tracy, San Andreas, and part of central California extending into central Yosemite National Park)

- Created by a split from **916** in 1958.
- Split in 1998 to create **559**.

210: Texas (San Antonio metropolitan area)

- Created by split from **512** in 1992.
- Three-way split in 1997 created **830** and **956**.

211: Community Services Hotline (e.g., crisis line, United Way, etc.)

212: New York City (Manhattan except for Marble Hill)

- Split in 1984 to create **718**.
- Overlain by **917** in 1992 and by **646** in 1999.

213: California (Downtown Los Angeles)

- Originally covered the southern third of California; split to create **714** (1951), **805** (1957), **818** (1984), **310** (1991), **323** (1998), and **424** (2006).

214: Texas (Dallas–Fort Worth metroplex)

- Split to create **817** (1953), **903** (1990) and **972** (1996). 214 and 972 were merged and overlain with **469** in 1999.

215: Pennsylvania (Philadelphia area including its suburbs in eastern Montgomery County and most of Bucks County as well as a very small portion of Berks County around the Hereford area, which is served by the Pennsburg 679 exchange.

- Split to create **610** in 1994.
- Overlain by **267** in 1997; a further overlay with **445** was proposed, but never implemented.

216: Ohio (Cleveland area)

- Three-way split in 1997 created **330** and **440**.

217: Illinois (Springfield, Champaign, Decatur, Urbana, Lincoln, and parts of central Illinois)

- Split in 1957 to create part of **309**.

218: Minnesota (Duluth, Moorhead, Thief River Falls, Bemidji, Brainerd, International Falls, and most of northern Minnesota)

- Split in 1954 to create **507**.

219: Indiana (northwest: Gary, Chesterton, Crown Point, East Chicago, Hammond, Hobart, Merrillville, Michigan City, Portage, Valparaiso)

- Created in 1948 by split from **317**.
- Split in 2002 to create **260** and **574**.

220–223: not used

224: Northeastern Illinois

- Overlain on **847** in 2002.

225: Louisiana (Baton Rouge, Donaldsonville, New Roads, White Castle, and east-central Louisiana)

- Created in 1998 by split from **504**.
- Mnemonic: *CAJun*

226: Ontario

- Overlain on **519** in 2006.
- Mnemonic: *CANada*

227 is assigned for numbering relief to **240** and **301** (Maryland), but no date has been scheduled for this to go into effect.

228: Mississippi (Gulfport, Biloxi, Pascagoula, Bay St. Louis, and southmost Mississippi)

- Created in 1997 by split from **601**.

229: Georgia (Albany, Valdosta, Bainbridge, Americus, Fitzgerald, and most of southwestern Georgia)

- Created in 2000 by split from **912**.

230: not used

231: Michigan (Muskegon, Traverse City, Ludington, Petoskey, and part of northwestern Michigan)

- Created in 1999 by split from **616**.

232–233: not used

234: Ohio—overlain on **330** in 2000.

235: not used

236: Overlay code assigned for numbering relief to **604**, **250**, and **778** in British Columbia, starting June 1,

2013.[1]

237–238: not used

239: Florida (southwest coast: all of Lee County, Collier County, the mainland part of Monroe County, including Cape Coral, Fort Myers, Naples, and Everglades)

- Created in 2003 by split from **941**.

240: Maryland

- Overlain on **301** in 1997.

241: not used

242: The Bahamas (all)

- Created in 1996 by split from **809**.
- Mnemonic: *BHA*.

243–245: not used

246: Barbados (all)

- Created in 1996 by split from **809**.
- Mnemonic: *BIM*

247: not used

248: Michigan (Oakland County)

- Created in 1997 by split from **810**.
- Overlain by **947**.

249: Ontario: (Northeastern Ontario and Central Ontario: Greater Sudbury, Sault Ste. Marie, North Bay, etc.)

- Overlain on **705** in 2011.

250: British Columbia (Victoria, Prince George, Prince Rupert, Kelowna, all areas except for Vancouver); also the town of Hyder, Alaska.

- Created in 1996 by split from **604**.
- Overlain by **778** (see **604**)

251: Alabama (Mobile County, Baldwin County, Bay Minette, Jackson, Brewton, Citronelle, and part of southwestern Alabama)

- Created in 2001 by split from **334**.

252: North Carolina (Greenville, New Bern, Kinston, Kitty Hawk, Rocky Mount)

- Created in 1998 by split from **919**.

253: State of Washington (Tacoma, Lakewood, Auburn, Puyallup, Enumclaw, Spanaway, and the southern suburbs of Seattle)

- Created in 1997 by split from **206**.

254: Texas (Waco, Killeen, Temple, Belton, and Stephenville)

- Created in 1997 by split from **817**.

255: not used

256: Alabama (Huntsville, Decatur, Cullman, Gadsden, Madison, Florence, Sheffield, Tuscumbia, Fort Payne, Scottsboro, and most of northern Alabama)

- Created in 1998 by split from **205**.
- Overlain by **938** in 2010.

257–259: not used

260: Indiana (Fort Wayne, New Haven, Decatur, Angola, Huntington, Wabash, and most of northeastern Indiana)

- Created in 2002 by split from **219**.

261: not used

262: Wisconsin (Racine, Kenosha, Menomonee Falls, Waukesha, and most of southeastern Wisconsin excluding Milwaukee County)

- Created in 1999 by split from **414**.

263: not used

264: Anguilla (all)

- Created in 1997 by split from **809**.
- Mnemonic: *ANG*

265–266: not used

267: Pennsylvania (Philadelphia and surrounding suburban area)

- Overlain on **215** in 1997.

268: Antigua and Barbuda (all)

- Created in 1996 by split from **809**.
- Mnemonic: *ANT*

269: Michigan (Battle Creek, Kalamazoo, Benton Harbor, Allegan, Hastings, St. Joseph, and most of southwestern Michigan)

- Created in 2002 by split from **616**.

270: Kentucky (Owensboro, Paducah, Bowling Green, Hopkinsville, Henderson, Elizabethtown, and most of western Kentucky)

- Created in 1999 by split from **502**.

271: not used

272: Pennsylvania (northeastern)—overlay of 570—effective 2012[2]

273: not used

274: Wisconsin—scheduled to overlay **920** in 2014

275: not used

276: Virginia (Bristol, Abingdon, Wytheville, Martinsville, Bluefield, and most of southwestern Virginia)

- Created in 2001 by split from **540**.

277–280: not used

281: Texas (Houston area)

- Created in 1996 by split from **713**; re-merged with 713 as an overlay in 1999.
- Overlain with **832** in 1999.

282: not used

283: proposed overlay of 513 (southwest Ohio) (Ten thousands blocks have been assigned to a switch in Cincinnati, Ohio.)[3]

284: the British Virgin Islands (all)

- Created in 1997 by split from **809**.
- Mnemonic: *BVI*

285–288: not used

289: Ontario

- Overlain on **905** in 2001.

290–299: Reserved for potential North American Numbering Plan expansion

300-399

300: not used

301: Maryland (Silver Spring, Washington, D.C., suburbs, all counties which touch the Potomac River, and the towns of Hagerstown, Frederick, Rockville, Cumberland, and land line telephones in all of western Maryland)

- Originally covered all of Maryland; split in 1991 to create **410**.

- Overlain by **240** in 1997.
- To be overlain with **227** in the future.

302: Delaware (all)

303: Colorado (Denver, Boulder, Longmont, Aurora, Golden, Limon, Centennial; central Colorado)

- Originally covered all of Colorado; split to create **719** (1988) and **970** (1995).
- Overlain by **720** in 1998.

304: West Virginia

- Overlain by **681** in 2009.

305: Florida (all of Miami-Dade County and the Florida Keys: Miami, Miami Beach, Homestead, Coral Gables, Marathon, Key West.

- Originally covered all of Florida; split to create **813** (1953), **904** (1965), **407** (1988), **954** (1995).
- Partial overlay by **786** in 1998, in Miami-Dade County only. Complete overlay including the Keys in 2008.

306: Saskatchewan (all)

- To be overlain by area code 639 in May, 2013.[4]

307: Wyoming (all)

308: Nebraska (North Platte, Scottsbluff, McCook, Kearney, Grand Island, and all of western Nebraska)

- Created in 1954 by split from **402**.

309: Illinois (Peoria, Bloomington, Moline, Rock Island, Galesburg, and west-central Illinois)

- Created in 1957 from parts of **217** and **815**.

310: California (Beverly Hills, Brentwood, Malibu, Pacific Palisades, Redondo Beach, Santa Monica, Torrance, Santa Catalina Island, and the coastal areas of Los Angeles County)

- Created in 1991 by split from **213**.
- Split in 1997 to create **562**.
- Overlain by **424** in 2006.

311: Local or government non-emergency use; City Hall in some localities.

312: Illinois (downtown Chicago)

- Split to create **708** (1989) and **773** (1996).
- Overlain by **872** in 2009.

313: Michigan (Dearborn, Detroit and its inner enclaves of Hamtramck and Highland Park)

- Split to create **810** (1993) and **734** (1997).

314: Missouri (St. Louis, St. Louis County, Florissant, Crestwood, Hazelwood, Kirwood, and surrounding sububs of St. Louis)

- Split in 1950 to create part of **417**.
- Split to create **573** (1996) and **636** (1999).

315: New York (Syracuse, Utica, Watertown, and north-central New York)

- Split in 1954 to create part of **607**.

316: Kansas (Wichita metropolitan area, McConnell Air Force Base, Augusta, El Dorado, Mulvane, and Hutchinson)

- Split in 2001 to create **620**.

317: Indiana (Indianapolis and immediate metro area including Carmel, Fishers, Noblesville, Greenwood, Mooresville, Beech Grove, Plainfield, Avon, and Brownsburg.)

- Split to create **219** (1948) and **765** (1997).

318: Louisiana (Shreveport, Monroe, Alexandria, Fisher, Tallulah, and most of northern Louisiana)

- Created in 1957 by split from **504**.
- Split in 1999 to create **337**.

319: Iowa (Cedar Rapids, Waterloo, Burlington, Iowa City, east-central, and southeastern Iowa)

- Split in 2001 to create **563**.

320: Minnesota (St. Cloud, Alexandria, Morris, Hutchinson, Sandstone, Appleton, Willmar and central Minnesota)

- Created in 1996 by split from **612**.

321: Florida (Orlando, Cocoa, Melbourne, Rockledge, Titusville, St Cloud, and east-central Florida). This Area partially overlies **407**, and this is the exclusive code for the Space Coast (Cape Canaveral, Titusville, Cocoa, Cocoa Beach, Satellite Beach, Melbourne, Merritt Island, Patrick Air Force Base, Palm Bay.)

- Created in 1999.
- Mnemonic: The final digits of a rocket countdown.

322: not used

323: California (part of the City of Los Angeles excluding Downtown Los Angeles, Western Los Angeles, Westwood, the San Fernando Valley, Florence, and Montebello)

- Created in 1998 by split from **213**.

324: not used

325: Texas (Abilene, San Angelo, Sweetwater, Snyder)

- Created in 2003 by split from **915**.

326: not used

327: Arkansas (Texarkana, Jonesboro, Pine Bluff, and most of southern, eastern, and northeastern Arkansas)

- Future overlay of 870, to be effective in 2013.

328–329: not used

330: Ohio (Akron, Canton, Youngstown, Warren, and most of northeastern Ohio)

- Created in 1997 by split from **216**.
- Overlain by **234** in 2000.

331: Illinois (Aurora, Naperville, Oswego, and other western suburbs of Chicago)

- Overlain on **630** in 2007.

332–333: not used

334: Alabama (Montgomery, Auburn, Dothan, Enterprise, Eufaula, Opelika, Phenix City, Selma, Tuskegee and most of southeastern Alabama)

- Created in 1995 by split from **205**.
- Split in 2001 to create **251**.

335: not used

336: North Carolina (Greensboro, Winston-Salem, Wilkesboro, Roxboro, and most of northwestern North Carolina)

- Created in 1997 by split from **910**.

337: Louisiana (Lafayette, Lake Charles, Leesville, New Iberia, Opelousas, and most of southwestern Louisiana)

- Created in 1999 by split from **318**.

338: not used

339: Massachusetts

- Overlain on **781** in 2001.

340: the U.S. Virgin Islands (all)

- Created in 1997 by split from **809**.

341 is assigned for numbering relief to **510** (east shore of San Francisco Bay, California), but no date has been scheduled for this to be implemented.

342: not used

343: Ontario: (Ottawa metropolitan area and southeastern Ontario)

Page 1355
T-Mobile Exhibit 1002
Part 4 of 6

- Overlain on **613** in 2010.

344: not used

345: the Cayman Islands (all)

- Created in 1996 by split from **809**.

346: not used

347: New York City

- Overlain on **718** (and part of **917**) in 1999.
- Overlain again by **929** in 2011.

348–350: not used

351: Massachusetts

- Overlain on **978** in 2001.

352: Florida (Gainesville, Ocala, Inverness, Dunnellon, and part of central Florida)

- Created in 1995 by split from **904**.
- Mnemonic: *FLA*

353–359: not used.

- **356** was authorized for use as a relief area code in New Jersey; however, **356** was replaced by 862 since this proposed code was condidered to be too similar to southern New Jersey's Area code 856

360: State of Washington (Olympia, Vancouver, Bellingham, Bremerton, Port Angeles, Aberdeen, and most of western Washington except the Seattle metropolitan area)

- Created in 1995 by split from **206**.

361: Texas (Corpus Christi, Victoria, George West, and much of South Texas)

- Created in 1999 by split from **512**.

362–363: not used

364: Kentucky—Planned relief of 270. Split plan canceled in December 2010, may be revived in the future as an overlay.

365: Ontario—overlay of **Area codes 905 and 289**, scheduled to be created on March 25, 2013[5]

- **Area code 742** reserved as a fourth code for the region.

366–368: not used

369 is assigned for numbering relief to **707** (northwest California), but no date has been scheduled for it to go into effect.

370–379: block reserved in case consecutive numbers are ever needed

380 is assigned for area code relief to **614** in Ohio but no date has been scheduled for this to be implemented. (Four thousands blocks are assigned to a switch in Palm Coast, Florida; and one thousands block to a switch in Shreveport, Louisiana.)[3]

381–384: not used

385: Utah

- Overlain on **801** in 2009.

386: Florida (Daytona Beach, Lake City, Live Oak, Crescent City, and parts of northeastern Florida)

- Created by split from **904** in 2001.
- Mnemonic: *FUN*

387: Reserved as a fourth area code to overlay **416** for the city of Toronto, Ontario, Canada with no set date for implementation

388-389: not used

390–399: Reserved for potential North American Numbering Plan expansion

400-499

400: not used

401: Rhode Island (all)

402: Nebraska (Omaha, Lincoln, Norfolk, Superior, and most of eastern Nebraska)

- Originally covered all of Nebraska; split in 1954 to create **308**.
- Overlain by **531** in 2011.

403: Alberta (Calgary, Banff, Red Deer, Medicine Hat, Lethbridge, Brooks, and most of southern Alberta)

- This area code formerly covered all of the Yukon Territory and part of the Northwest Territories. These were split off in 1997 to compose part of **867**.
- Split in 1999 to create **780**.
- All of Alberta (403 and 780) has now been overlain by **587**.

404: Georgia: Atlanta and the Atlanta metropolitan area inside of the Interstate 285 perimeter highway

- This area code originally covered all of Georgia, but it was split to create **Area code 912** in 1954; **Area code 706** in 1992; and **area code 770** in 1995. Area code 404 is now completely surrounded by area code 770, which forms an annulus around it.
- Overlain by **678** in 1998.

405: Oklahoma (Oklahoma City, Stillwater, Edmond, Norman, Shawnee, and most of central Oklahoma)

- Originally covered all of Oklahoma; split to create **918** in 1953 and **580** in 1997.

406: Montana (all)

407: Florida (Orlando, Sanford, St. Cloud, Kissimmee, and part of east-central Florida)

- Created in 1988 by split from **305**.
- Split in 1996 to create **561**.
- Split *and* overlain, partially, by **321** in 1999.

408: California (San Jose, Cupertino, Los Gatos, Milpitas, Mountain View, Santa Clara, Sunnyvale, and Silicon Valley)

- Created in 1959 by split from **415**.
- Split in 1998 to create **831**.

409: Texas (Beaumont, Galveston, Orange, Port Arthur, and Texas City.

- Created in 1983 by split from **713**.
- Three-way split in 2000 created **936** and **979**.

410: Maryland (except for St. Mary's County, all counties and cities which touch the Chesapeake Bay, the Atlantic Ocean, or Delaware, including Annapolis, metropolitan Baltimore, Berlin, Cambridge, Ocean City, Princess Anne, and Salisbury) as well as Howard County (Columbia).

- Created in 1991 by split from **301**.
- Overlain by **443** in 1997.

411: Directory Services; 1-areacode-555-1212 is also used to reach directory services in areas other than the caller's own area code.

412: Pennsylvania (Pittsburgh metropolitan area, Braddock, Duquesne, Johnstown, McKeesport)

- Split in 1998 to create **724**.
- Overlain by **878** in 2001.

413: Massachusetts (Springfield, Pittsfield, Holyoke, Greenfield, and most of Western Massachusetts)

414: Wisconsin (Milwaukee County and the city of Milwaukee)

- Split in 1955 to create part of **608**.
- Split to create **920** (1997) and **262** (1999).

415: California (San Francisco and most of Marin County)

- Split in 1959 to create **408** and **707**.
- Split in 1991 to create **510**.
- Split in 1997 to create **650**.

416: Ontario (the City of Toronto)

- Split in 1953 to create part of **519**.
- Split in 1993 to create **905**.
- Overlain by **647** in 2001.
- Overlain by **437** in 2013.

417: Missouri (Springfield, Joplin, Branson, Lamar, Lebanon, and most of southwestern Missouri)

- Created in 1950 from parts of the area codes **314** and **816**.

418: Quebec (Quebec City, Saguenay, the Gaspé Peninsula, Côte-Nord, Chibougamau, St-Georges), Maine (Estcourt Station)

- Overlain by **581** in 2008.

419: Ohio (Toledo, Sylvania, Mansfield, Lima, Findlay, Sandusky, Bowling Green, and most of northwestern Ohio)

- Overlain by **567** in 2002.

420–422: not used

423: Tennessee (two discontiguous portions of East Tennessee: Bristol, Johnson City, Kingsport, etc., in the northeast; and Chattanooga, Cleveland, Philadelphia, and McMinnville in the southeast

- Created in 1995 by split from **615**.
- Split in 1999 to create **865**.

424: California

- Overlain on **310** in Los Angeles County in 2006.

425: State of Washington (the northern and eastern suburbs of Seattle: Bellevue, Everett, Edmonds, Kirkland, Redmond, Renton, Sammamish, Issaquah, and Lynnwood.)

- Created in 1997 by split from **206**.

426–429: not used

430: Texas

- Overlain on **903** in 2003.

431: Manitoba. Will overlay **204** and be active beginning November 2012.

432: Texas (West Texas: Midland, Odessa, Big Spring, Alpine, Fort Stockton.

- Created in 2003 by split from **915**.

433: not used

434: Virginia (Charlottesville, Lynchburg, Danville, and south-central Virginia)

- Created in 2001 by split from **804**.

435: Utah (Cedar City, Logan, Moab, Park City, St. George, and all of Utah excluding the areas of Salt Lake City, Ogden, Provo, Morgan County, Davis County, and north-central Utah.)

- Created in 1997 by split from **801**.

436: not used

437: Province of Ontario: Toronto metropolitan area

- Reserved for use as an overlay on **416** and **647**. Scheduled to come into effect on March 25, 2013.

438: Province of Quebec: Montreal metropolitan area

- Overlain onto **514** in 2006.

439: not used

440: Ohio (surrounding Cleveland on three sides, including: Elyria, Lorain, Oberlin, Ashtabula, and most of north-central Ohio)

- Created in 1997 by split from **216**.

441: Bermuda (all)

- Created in 1995 by split from **809**.

442: California

- Overlain onto **760** in 2009.

443: Maryland

- Overlain onto **410** in 1997, mostly for the use of cellular telephones

444: not used

445 was formerly assigned to overlay **215** and **267** (Philadelphia, Pennsylvania), but this number is no longer assigned for any location. (One thousands block is assigned to a switch in Tucson Arizona.)[3]

446: not used

447: Illinois

- Proposed overlay onto **217**.

448–449: not used

450: Quebec (central southern Quebec including the city of Laval and other suburbs of Montreal but excluding the City of Montreal)

- Created in 1998 by split from **514**.
- Overlain by **579** in 2010.

451–455: not used

456: Inbound international calls for carrier-specific services

457: not used

458: Oregon (Eugene, Medford, Bend, Pendleton, Corvallis, Ontario, Burns, and all of Oregon except the metropolitan region of northwestern Oregon)

- Overlain on **541** in 2010.

459–463: not used

464: Illinois

- Proposed overlay for **708**.

465–468: not used

469: Texas

- Overlain on **214** and **972** in 1999.

470: Georgia: An overlay onto **Area code 404**, area code **678**, and **Area code 770**

- Activated in 2010.[6]

471–472: not used

473: Grenada (all)

- Created in 1997 by split from **809**.
- Mnemonic: *GRE* or *GRD*

474: not used.

475: Connecticut

- Overlain onto **203** in 2009.

476–477: not used

478: Georgia (Macon, Warner Robins, Swainsboro, Milledgeville, Perry, and part of central Georgia)

- Created in 2000 by split from **912**.

479: Arkansas (Fort Smith, Fayetteville, Rogers, and most of northwestern Arkansas)

- Created in 2002 by split from **501**.

480: Arizona (Mesa, Scottsdale, Chandler, Tempe, and the eastern Phoenix metropolitan area)

- Created in 1999 by split from **602**.

481–483: not used

484: Pennsylvania

- Overlain on **610** in 1999.

485–489: not used

490–499: Reserved for potential North American Numbering Plan expansion

500-599

500: Personal Communications Services

501: Arkansas (Little Rock, Hot Springs, and much of central Arkansas, but not Pine Bluff.)

- Originally covered all of Arkansas; split to create **870** (1997) and **479** (2002).

502: Kentucky (Louisiville, Frankfort, Shelbyville, Bardstown, and most of north-central Kentucky)

- Originally covered all of Kentucky; split to create **606** in 1954 and **270** in 1999.

503: Oregon (Portland, Salem, Hillsboro, St. Helens, Tillamook, Astoria and most of northwestern Oregon)

- Originally covered all of Oregon; split in 1995 to create **541**.
- Partially overlaid by **971** in 2000, excluding Clatsop County and Tillamook County. Complete overlay implemented in 2008.

504: Louisiana: New Orleans metropolitan area

- Originally covered all of Louisiana; split to create **318** (1957), **225** (1998) and **985** (2001). New Orleans and Baton Rouge are no longer in the same area code.

505: New Mexico: (Albuquerque, Santa Fe, Farmington, Gallup, and all of northwestern New Mexico, and part of central New Mexico).

- Originally covered all of New Mexico; split in 2007 to create **575**.

506: New Brunswick

- Created in 1955 by split from **902**.
- Split in 1962 to create **709**.

507: Minnesota (Rochester, Mankato, Austin, Marshall, Winona and most of southern Minnesota)

- Created in 1954 by split from **612**.

508: Massachusetts (Worcester, New Bedford, Fall River, Cape Cod, and most of southeastern Massachusetts)

- Created in 1988 by split from **617**.
- Split in 1997 to create **978**.
- Overlain by **774** in 2001.

509: State of Washington (all of eastern Washington, including Spokane, Ellensburg, Pullman, the Tri-Cities area, Walla Walla, Wenatchee, and Yakima.)

- Created in 1957 by split from **206**.

510: California (Oakland, Fremont, Hayward, Richmond, Berkeley and Alameda)

- Created in 1991 by split from **415**.
- Split in 1998 to create **925**.

511: Used as a state or local transportation or transit information hub. Otherwise used to direct local police non-emergency services in some communities.

512: Texas (Austin, San Marcos, and parts of central Texas)

- Split to create **210** (1992) and **361** (1999).

513: Ohio (Cincinnati, Middletown, Hamilton, Lebanon, and parts of southern and southwestern Ohio. This area code formerly also included Dayton.)

- Split in 1996 to create **937**.
- The proposed overlay with area code **283** has been postponed.

514: Province of Quebec: the entire Island of Montreal, Île Perrot, and Île Bizard

- Split to create **819** (1957) and **450** (1998).
- Overlain by **438** in 2006.

515: Iowa (Des Moines, Ames, Fort Dodge, Jefferson, Indianola and most of north-central Iowa)

- Split in 2000 to create **641**.

516: New York State (Nassau County, Hempstead, Long Beach, Great Neck, etc.)

- Created in 1951 by split from **914**.
- Split in 1999 to create **631**.

517: Michigan (Lansing, Jackson, Charlotte, Deerfield, Addison, and most of south-central Michigan)

- Split in 2000 to create **989**.

518: New York State (Albany, Schenectady, Plattsburgh, Saranac Lake, Lake George, Westport, and most of northeastern New York State)

519: Province of Ontario (London, Windsor, Kitchener-Waterloo, Cambridge, and most of southwestern Ontario)

- Created in 1953 from parts of **416** and **613**.
- Split in 1957 to create part of **705**.
- Overlain by **226** in 2006.

520: Arizona (Tucson, Nogales, Fort Huachuca, and most of southeastern Arizona)

- Created in 1995 by split from **602**.
- Split in 2001 to create **928**.

521: Reserved for Personal Communications Services-effective July 2011[7]

522: Reserved for Personal Communications Services

523-529: Reserved for Personal Communications Services-effective July 2011

530: California (Redding, Auburn, Chico, Davis, Lake Tahoe, Placerville, Susanville, Truckee, Yreka, and most of northeastern California)

- Created in 1997 by split from **916**.

531: Nebraska (Omaha, Lincoln, Norfolk, Superior, and most of eastern Nebraska)

- Overlain on **402** in 2011.

532: Reserved for Personal Communications Services-effective July 2011

533: Personal Communications Services

534: Wisconsin—an overlay for **715**

535 – 537: not used

538: Reserved for Personal Communications Services-effective July 2011

539: Oklahoma (Tulsa, Bartlesville, McAlester, Muskogee, Henryetta and northeastern Oklahoma)

- Overlain on **918** in 2011.

540: Virginia (Fredericksburg, Roanoke, Blacksburg, Harrisonburg, Lexington, Staunton, Winchester, and parts of north-central Virginia)

- Created in 1995 by split from **703**.
- Split in 2001 to create **276**.

541: Oregon (Eugene, Bend, Corvallis, Medford, Pendleton, and all of Oregon except metropolitan northwestern Oregon, including Portland, Salem, Astoria, etc.)

- Created in 1995 by split from **503**.
- Overlain in 2010 by **458**.

542-543: Reserved for Personal Communications Services-effective July 2011

544: Personal Communications Services

545: Reserved for Personal Communications Services-effective July 2011

546: not used

547: Reserved for Personal Communications Services-effective July 2011

548: not used

549: Reserved for Personal Communications Services-effective July 2011

550: not used

551: New Jersey

- Overlain onto **201** in 2001.

552-554: Reserved for Personal Communications Services-effective July 2011

555: Reserved for directory assistance applications.

- This is not the same as the 555 exchange, which is used for long-distance directory assistance or for fictitious purposes, such as on TV and in films.

556: Reserved for Personal Communications Services-effective July 2011

557: Missouri—planned overlay for **314**

558: not used

559: California (Fresno, Hanford (Calif.), Madera, Tulare, Visalia, and parts of the Central Valley of California.

- Created in 1998 by split from **209**.

560: not used

561: Florida (Palm Beach County, including Palm Beach, West Palm Beach, Boca Raton, Boynton Beach, etc.)

- Created in 1996 by split from **407**.
- Split to create **772**.

562: California (Long Beach, Whittier, Norwalk, La Habra, Lakewood, Pico Rivera, and most of southeastern Los Angeles County)

- Created in 1997 by split from **310**.

563: Iowa (Davenport, Dubuque, Clinton, Bettendorf, and most of eastern and northeastern Iowa)

- Created in 2001 by split from **319**.

564: State of Washington (Western Washington)—a proposed overlay area code for **206, 253, 360** and **425**

565: not used

566: Reserved for Personal Communications Services

567: Ohio

- Overlain on **419** in 2002.

568: not used

569: Reserved for Personal Communications Services-effective July 2011

570: Pennsylvania (Scranton, Wilkes-Barre, Bloomsburg, Danville, Nanticoke, Williamsport, and most of

northeastern Pennsylvania)

- Created in 1998 by split from **717**.

571: Virginia

- Overlain on **703** in 2000.

572: not used

573: Missouri (Columbia, Jefferson City, Hannibal, Cape Girardeau, Farmington, Lake of the Ozarks, Poplar Bluff, and most of eastern Missouri excluding St. Louis County and the City of St. Louis)

- Created in 1996 by split from **314**.

574: Indiana (South Bend, Elkhart, Goshen, and most of north-central Indiana)

- Created in 2002 by split from **219**.

575: New Mexico (Las Cruces, Roswell, Carlsbad Caverns National Park, Socorro, Taos, Truth or Consequences, and all areas of this state outside of central and northwestern New Mexico)

- Created in 2007 by split from **505**.

576: not used

577: Reserved for Personal Communications Services

578: Reserved for Personal Communications Services-effective July 2011

579: Province of Quebec (central southern Quebec including the city of Laval and other suburbs of Montreal but excluding the City of Montreal)

- Overlain on **450** in 2010.

580: Oklahoma (Ponca City, Ardmore, Enid, Lawton, Elk City, and most of southern and western Oklahoma)

- Created in 1997 by split from **405**.

581: Province of Quebec

- Overlain on **418** in 2008.

582: Pennsylvania—Relief of 814, effective 2012; currently proposed as a split including northwestern Pennsylvania around Erie

583–584: not used

585: New York State (Rochester, Batavia, and much of western New York State)

- Created in 2002 by split from **716**.

586: Michigan (Warren, Sterling Heights, and Macomb County)

- Created in 2001 by split from **810**.

587: Alberta

- Overlain on **403** and **780** in 2008.

588: Reserved for Personal Communications Services

589: Reserved for Personal Communications Services-effective July 2011

590–599: Reserved for potential North American Numbering Plan expansion

600-699

600: Special telecommunications needs in Canada

601: Mississippi (Jackson, Hattiesburg, Meridian, Natchez, Vicksburg, and most of central Mississippi)

- Originally covered all of Mississippi; split to create **228** (1997) and **662** (1999).
- Overlain by **769** in 2005.

602: Arizona: Downtown Phoenix only

- Originally covered all of Arizona; split to create **520** (1995), **480** and **623** (1999).

603: New Hampshire (all)

604: British Columbia (Greater Vancouver Regional District, Whistler, and remaining portion of **604** not part of ovcrlay complcx).

- Originally covered all of British Columbia; split in 1996 to create **250**.
- Partly overlain by **778** in 2001; the overlay was extended to all of **604** as well as **250** in 2008.

605: South Dakota (all)

606: Kentucky (Ashland, Pikeville, Hazard, Somerset, London, Corbin, Maysville, and much of eastern Kentucky)

- Created in 1954 by split from **502**.
- Split in 2000 to create **859**.

607: New York State (Binghamton, Elmira, Cornell, Norwich, Ithaca, and most of south-central New York)

- Created in 1954 from parts of **315** and **716**.

608: Wisconsin (Madison, La Crosse, Platteville, Beloit, and most of southwestern Wisconsin)

- Created in 1955 from parts of **414** and **715**.

609: New Jersey (Trenton, Atlantic City, Princeton, and most of central & southeastern New Jersey)

- Created in 1958 by split from **201**.
- Split in 1999 to create **856**.

610: Pennsylvania (Allentown, Reading, Norristown, Chester and parts of southeastern Pennsylvania)

- Created in 1994 by split from **215**.
- Overlain by **484** (1999).

611: Repair service for land-line telephones; customer service for most wireless carriers

612: Minnesota (Minneapolis, Fort Snelling, St. Anthony, and Richfield)

- Split in 1954 to create **507**.
- Split to create **320** (1996), **651** (1998), **763** and **952** (2000).

613: Province of Ontario (Ottawa and eastern Ontario)

- Split to create part of **519** (1953)
- Split to create part of **705** (1957)
- Overlain by **343** in 2010.

614: Ohio Columbus and Franklin County

- Split in 1997 to create **740**.

615: Tennessee (Nashville, Cookeville, Lebanon, Murfreesboro, Sparta, Smyrna, Springfield, and most of Middle Tennessee)

- Created in 1954 by split from **901**.
- Split to create **423** (1995) and **931** (1997).

616: Michigan (Grand Rapids, Greenville, Holland, Ionia, Zeeland, and most of southwestern Michigan)

- Split to create **906** in 1961; **231** in 1999, and **269** in 2002.

617: Massachusetts (Boston, Cambridge, Quincy, Newton, Everett, and the nearby suburbs)

- Split to create **508** (1988) and **781** (1997).
- Overlain by **857** in 2001.

618: Illinois (Carbondale, Alton, Belleville, Cahokia, Centralia, Edwardsville, Marion, Metropolis, Vandalia, and most of southern Illinois)

619: California (southern half of San Diego, California city, and southern and eastern suburbs including Chula Vista, National City, El Cajon, La Mesa, Lemon Grove.

- Created in 1982 by split from **714**.
- Split to create **760** (1997) and **858** (1999).

620: Kansas (southern Kansas not including the Wichita Metropolitan Area and Hutchison.)

- Created in 2001 by split from **316**.

621–622: not used

623: Arizona (part of Maricopa County, including Glendale, Buckeye, Goodyear, Peoria, Sun City, and the

western half of Phoenix)

- Created in 1999 by three-way split from **602** (along with **480**).

624–625: not used

626: California (Pasadena, Arcadia, Clermont, Covina, El Monte, La Cañada-Flintridge, La Puente, Monrovia, the San Gabriel Valley, Upland, and West Covina)

- Created in 1997 by split from **818**.

627 is assigned for numbering relief to Area code 707 (the northern California coast), but no date has been scheduled for this to go into effect.

628 is assigned for numbering relief to Area code 415 (San Francisco, but no date has been scheduled for this to be implemented.

629: not used

630: Illinois (Aurora, Naperville, Oswego, and other suburbs west of Chicago.

- Created in 1996 by split from **708**.
- Overlain by **331** in 2007.

631: New York State (Suffolk County on Long Island)

- Created in 1999 by split from **516**.

632–635: not used

636: Missouri (St. Charles, Chesterfield, Union, Troy, and parts of east-central Missouri)

- Created in 1999 by split from **314**.

637–638: not used

639: Overlay code assigned for numbering relief to **306** in Saskatchewan, starting May, 2013.[4]

640: not used

641: Iowa (Mason City, Oskaloosa, Creston, Pella, Ottumwa, Britt, Clear Lake, Fairfield, and parts of central Iowa)

- Created in 2000 by split from **515**.

642–645: not used

646: New York State (Borough of Manhattan, except for Marble Hill on the mainland of New York)

- Overlain on **212** and **917** in 1999.
- Mnemonic: *MHN*

647: Province of Ontario

- Overlain on **416** in 2001.

648: not used

649: the Turks and Caicos Islands (all)

- Created in 1997 by split from **809**.

650: California (Palo Alto, Redwood City, Menlo Park, Mountain View, San Mateo, Santa Clara, Stanford, South San Francisco)

- Created in 1997 by split from **415**.

651: Minnesota (St. Paul, Eagan, Lindstrom, Red Wing, Hastings, Stillwater and part of east-central Minnesota)

- Created in 1998 by split from **612**.

652–656: not used

657: California

- Overlain on **714** in 2008.

658: not in current use. Listed for Jamaica, but no date is scheduled for this to go into effect.

659 is assigned for numbering relief to **205** (Alabama) but no date has been scheduled for this to go into effect.

660: Missouri (Sedalia, Kirksville, Maryville, Mexico, Whiteman Air Force Base, and part of north-central Missouri)

- Created in 1997 by split from **816**.

661: California (Bakersfield, Lancaster, Palmdale, Edwards Air Force Base, and parts of Kern County, and Los Angeles County.

- Created in 1999 by split from **805**.

662: Mississippi (Tupelo, Columbus, Corinth, Greenville, Greenwood, Starkville, and most of northern Mississippi)

- Created in 1999 by split from **601**.

663: not used

664: Montserrat (all)

- Created in 1996 by split from **809**.
- Mnemonic: *MOI*

665–666: not used

667 is an overlay code for **410 and 443** (Maryland), going into effect in March 2012.

668: not used

669 is assigned for numbering relief to **408** (Calif.) but no date has been scheduled for this to be implemented.

670: Northern Mariana Islands: the former country code for this present Commonwealth of the United States that includes Saipan, Tinian, and Rota.)

671: Guam: the former country code for this possession and unorganized territory of the United States that includes Andersen Air Force Base.

672: British Columbia is assigned for numbering relief to **236** (British Columbia)[8] but no date has been scheduled for this to go into effect.

673–677: not used

678: Georgia

- Overlain on **404** and **770** in 1998.
- Overlain by 470 in 2010.

679 is assigned for numbering relief to **313** (Michigan) but no date has been scheduled for this to go into effect.

680: not used

681: West Virginia

- Overlain on **304** in 2009.

682: Texas

- Overlain on **817** in 2000.

683: not used

684: American Samoa (the former country code for this possession and unorganized territory of the United States)

685–688: not used

689 is assigned for numbering relief to Area code 407 (Florida) but no date has been scheduled for this to be implemented.

690–699: Reserved for potential North American Numbering Plan expansion

700-799

700: Long Distance carrier use for pre-subscribing phone numbers, 1-700-555-4141 most often gives a recorded message indicating the default carrier on a line.

701: North Dakota (all)

702: Nevada (almost all of Clark County, including all of the Las Vegas Valley, including Henderson and Boulder City)

- Originally covered all of Nevada; split in 1998 to create **775**.

703: Virginia (Northern Virginia: mostly the suburbs of Washington, D.C., including Alexandria, Arlington County, Fairfax County, Prince William County, and eastern Loudoun County.

- Originally covered all of Virginia; split to create **804** in 1973 and **540** in 1995.
- Overlain by **571** in 2000.

704: North Carolina (Charlotte, Concord, Gastonia, Salisbury, and much of south-central North Carolina)

- Originally covered all of North Carolina; split to create **919** in 1954 and **828** in 1998.
- Overlain by **980** in 2001.

705: Ontario: (Northeastern Ontario and Central Ontario: Greater Sudbury, Sault Ste. Marie, North Bay, etc.)

- Created in 1957 from parts of **519** and **613**.
- Split in 1962 to create **807**.
- Overlain by **249** in 2011.

706: Georgia (Augusta, Columbus, Carrollton, Carroll County, Cartersville, Dalton, Rome, and much of northern and west-central Georgia)

- Created in 1992 by split from **404**.
- Overlain by **762**.

707: California (Vallejo, Crescent City, Eureka, Redwoods National Park, Santa Rosa, Ukiah, and most of northwestern California)

- Created in 1959 by split from **415**.

708: Illinois (Berwyn, Chicago Heights, Cicero, Elmwood Park, Evergreen Park, Oak Forest, Oak Park, Skokie, Tinley Park, West Chicago, Wheaton, and other southern and inner-western suburbs of Chicago.

- Created in 1989 by split from **312**.
- Split twice in 1996 to create **847** and **630**.

709: Newfoundland and Labrador

- Created in 1962 by split from **506**.

710: U.S. Government Special Services

711: Telecommunications device for the deaf/Relay Service for TTY to Voice and Voice to TTY

712: Iowa (Sioux City, Council Bluffs, Denison, and most of western Iowa). This is one of the original area codes of the U.S. that has gone unchanged and undivided. This is because for some reason, Iowa was given three area codes to begin with back in 1948, even though it did not need that many.

713: Texas (Houston area)—overlays with **281** and **832**

- Split to create **409** (1983) and **281** (1996); **713** and **281** were later un-split to become an overlay, and further overlain by **832** in 1999.

714: California: Orange County: Anaheim, Fullerton, Garden Grove, Huntington Beach, Irvine, Newport Beach, Orange, Santa Ana, etc.)

- Created in 1951 by splitting off from **213**, when it did include San Diego County and several other counties.
- Repeatedly split to create **619** in 1982; **909** in 1992; and **949** in 1998.
- Overlain by **657** in 2008.

715: Wisconsin (Wausau, Eau Claire, Rhinelander, and most of northern Wisconsin)

- Split in 1955 to create part of **608**.
- Overlain by **534** in 2010.

716: New York State (Buffalo, Niagara Falls, Olean, and parts of western New York)

- Split in 1954 to create part of **607**.
- Split in 2002 to create **585**.

717: Pennsylvania (Harrisburg, Gettysburg, Lancaster, York and most of south-central Pennsylvania)

- Split in 1998 to create **570**.

718: New York State (New York City except for Manhattan, but this area code includes Marble Hill, a neighborhood of the Borough of Manhattan located on the mainland.

- Created in 1984 by splitting it off from **212**.
- Overlain by **347**, **917** and **929**.

719: Colorado (Colorado Springs, Pueblo, Florence, Leadville, Limon, Trinidad, La Junta, and most of southeastern Colorado)

- Created in 1988 by being split off from **303**.

720: Colorado

- Overlain on **303** in 1998.

721: Sint Maarten

- Joined the NANP on September 30, 2011; previously +599.[9]

722–723: not used

724: Pennsylvania (Washington, Pa., Indiana, Pa., New Castle, Uniontown, and most of southwestern Pennsylvania)

- Created in 1998 by split from **412**.
- Overlain by **878** in 2001.

725–726: not used

727: Florida (all of Pinellas County including St. Petersburg, Clearwater, Dunedin, Palm Harbor, Tarpon Springs, and the coastal parts of Pasco County)

- Created in 1998 by split from **813**.

728–729: not used

730: Illinois—proposed overlay of **618**

731: Tennessee (most of West Tennessee—*excluding Metropolitan Memphis* and Shelby County (area code 901)—but including Dyersburg, Jackson, Martin, Paris, Union City, and Crockett County)

- Created in 2001 by split from **901**.

732: New Jersey (New Brunswick, Lakewood, Neptune, Fort Dix, and most of east-central New Jersey—but not Atlantic City)

- Created in 1997 by split from **908**.
- Overlain by **848** in 2001.
- Mnemonic: *SEA*

733: not used

734: Michigan (Ann Arbor, Hell, Monroe, Plymouth, Romulus, Wayne, Ypsilanti, and the southwestern suburbs of Detroit)

- Created in 1997 by split from **313**.

735–736: not used

737 is assigned for numbering relief to **512** (Texas) but no date has been scheduled for this to go into effect.

738–739: not used

740: Ohio (Suburban Columbus, central Ohio outside of Franklin County, excluding the Marysville area, together with southeastern Ohio: Athens, Lancaster, Cambridge, Delaware, Marietta, Mt. Vernon, Newark, Portsmouth, Steubenville, Washington Court House, and Zanesville)

- Created in 1997 by split from **614**.

741: not used

742: Reserved as a future area code in the 289/905 region of Ontario, Canada (proposed to be overlain by 365 in 2013).[10]

743–746: not used

747: California (Los Angeles County, San Fernando Valley): an overlay onto **818**, effective on May 18, 2009.

748–753: not used

754: Florida an overlay onto **954**

755–756: not used

757: Virginia (Part of Hampton Roads and the Eastern Shore of Virginia)

- Created in 1996 by split from **804**.

758: Saint Lucia (all)

- Created in 1996 by split from **809**.
- Mnemonic: *SLU*

759: not used

760: California (Bishop, Ridgecrest, Barstow—eastern portions of San Bernardino County and Riverside County, including Palm Springs, Palm Desert, and Indio—Imperial County, and parts of northern and eastern San Diego County, including Oceanside and other northern suburbs of San Diego, California.) In summary: most of the desert and mountain portions of southeastern California.

- Created in 1997 by split from **619**. This was the first part of North America to have its code changed three times: from 213 to 714 in 1951, to 619 in 1982, and to 760 in 1997.
- Overlain by **442** in 2009.

761: Currently reserved for Palm Beach County, FL

762: Georgia: an overlay of **706**

763: Minnesota (Brooklyn Park, Plymouth, Maple Grove, Monticello, Elk River, Fridley, Blaine, and the northwest suburban area of Minneapolis)

- Created in 2000 (along with **952**) by three-way split from **612**.

764 is assigned for numbering relief to **650** (western San Francisco Bay, including Contra Costa County) but no date has been scheduled for this to go into effect.

765: Indiana (Lafayette, Marion, Muncie, Richmond, West Lafayette, and most of central Indiana excluding Indianapolis and immediate suburbs.)

- Created in 1997 by split from **317**.

766: not used

767: Commonwealth of Dominica (all)—split from **809**

- Created in 1997 by split from **809**.
- Mnemonic: *ROS*

768: not used

769: Mississippi

- Overlain onto **601** in 2005.

770: Georgia (Marietta, Carrollton, Gainesville, Jonesboro, Lawrenceville, Roswell, Stone Mountain, and much of north-central Georgia outside of Atlanta's Interstate 285 Perimeter Highway). Area code 770 is shaped like a donut, with Atlanta, Chamblee, Decatur, and Tucker 404 in its hole.

- Created in 1995 by split from **404**.

- Overlain by **678** in 1998.

771: not used

772: Florida (Fort Pierce, Port Saint Lucie, Sebastian, Stuart, and Vero Beach, and all of Indian River County and Martin County)

- Created by splitting it off from **561**.

773: Illinois (City of Chicago, excluding downtown)

- Created in 1996 by split from **312**.
- Overlain in 2009 by **872**.

774: Massachusetts

- Overlain on **508** in 2001.

775: Nevada (Carson City, Reno, Elko, Ely, Sparks, Winnemucca, Great Basin National Park, Naval Air Station Fallon, and all of Nevada except for most of Clark County in southernmost Nevada)

- Created in 1998 by split from **702**.

776–777: not used

778: British Columbia (all)

- Created in 2001 as concentrated overlay of **604**; extended in 2008 to cover all of **604** and **250**

779: Illinois

- Overlain on **815** in 2007.

780: Alberta (Edmonton, Jasper, Grande Prairie, Peace River, and all of northern Alberta). This area code overlays onto **587** and also **403** in southern Alberta.

- Created in 1999 by split from **403**.

781: Massachusetts (Saugus, Norwood, Waltham Woburn, and other suburbs of Boston along Route 128)

- Created in 1997 by split from **617**.
- Overlain by **339** in 2001.

782–783: not used

784: Saint Vincent and the Grenadines (all). Was split from **809**

- Created in 1998 by split from **809**.
- Mnemonic: *SVG* or *SVI*

785: Kansas (Topeka, Salina, Colby, Lawrence, Manhattan, and all of northern and central Kansas not including the Kansas City Metropolitan Area.

- Created in 1997 by spliting it from **913**.

786: Florida (Miami-Dade County and the Florida Keys)

- Overlain on **305** only in Miami-Dade County in 1998. Overlay extended to the Keys in 2008.
- Mnemonic: *SUN*

787: Puerto Rico

- Created in 1996 by split from **809**.
- Overlain by **939** in 2001.
- Mnemonic: *PUR* or *PTR*

788–789: not used

790–799: Reserved for potential North American Numbering Plan expansion

800-899

800: Toll-free telephone service (see also **822**, **833**, **844**, **855**, **866**, **877**, **880** through **882**, **888**)

801: Utah (in the Wasatch Front)

- Originally covered all of Utah; split in 1997 to create **435**.
- Overlain by **385** in 2009.

802: Vermont (all)

803: South Carolina (Columbia, Rock Hill, Sumter, Aiken, and most of central South Carolina)

- Originally covered all of South Carolina; split twice to create **864** (1995) and **843** (1998).

804: Virginia (the Richmond Metropolitan Area, including Petersburg; also the Northern Neck and Middle Peninsula)

- Created in 1973 by split from **703**.
- Split to create **757** (1996) and **434** (2001).

805: California (Ventura County, San Luis Obispo County, Santa Barbara County, Atascadero, Lompoc, Ojai**, Oxnard, Pismo Beach, San Buenaventura, Santa Maria, the Simi Valley, and Vandenberg Air Force Base.

- Created in 1957 by split from **213**.
- Split in 1999 to create **661**.

806: Texas (Lubbock, Amarillo, Canadian, Canyon, Dalhart, and the entire Texas Panhandle)

- Created in 1957 from parts of **817** and **915**.

807: Province of Ontario (Northwestern Ontario: Thunder Bay, Kenora, Dryden, Greenstone)

- Created in 1962 by split from **705**.

808: Hawaii (covers all of the Hawaiian Islands to Midway Atoll, and major Hawaiian cities including Honolulu, Kailua, Mililani, Haleiwa, Hilo, Kahului, Lihue, etc. Area code also includes Wake Island)

809: Dominican Republic.

- Originally covered most of the Caribbean; split to create **441** (1995), **787**, **268**, **246**, **664**, **758**, **345**, **242**, **869** (1996), **264**, **876**, **340**, **649**, **868**, **284**, **767**, **473** (1997), **784** (1998).
- Overlain by **829** (2005) and **849** (2009).

810: Michigan (Port Huron, Flint, Lapeer, and Michigan "Thumb")

- Created in 1993 by split from **313**.
- Split to create **248** (1997) and **586**.

811: Mobile customer service on some North American cellular carriers, also used as a 9-1-1 like emergency number to handle distress calls in much of Jamaica. This code now has separate mandated usages in the U.S. and Canada. In the U.S., the FCC adopted it as the local underground utility assistance service in 2007. In Canada, the CRTC reserved it for non-urgent telehealth services in 2005, though not all provinces and territories have yet adopted it.

812: Indiana (Southern Indiana, including Bloomington, Evansville, Terre Haute, Columbus, Jeffersonville, Lawrenceburg, Madison, and New Albany.

813: Florida (all of Hillsborough County, including Tampa and its suburbs, MacDill Air Force Base, and Plant City; and also the inland areas of Pasco County)

- Created in 1953 by splitting it off from **305** to give Florida more than one area code.
- Split and resplit to create **941** in 1995) and **727** in 1998.

814: Pennsylvania (Erie, State College, Altoona, Clearfield, Emporium, Johnsonburg, Johnstown, Meyersdale, Ridgway, Somerset, and most of northwestern and parts of central Pennsylvania)

815: Illinois (Rockford, La Salle, DeKalb, and much of northern Illinois, including western Chicago suburbs such as Joliet, Plainfield, and Crystal Lake)

- Split in 1957 to create part of **309**.
- Overlain by **779** on March 17, 2007.

816: Missouri (Kansas City, St Joseph, Independence, Harrisonville, and parts of west-central Missouri)

- Split in 1950 to create part of **417**.
- Split again in 1997 to create **660**.

817: Texas (Fort Worth, Arlington, Grand Prairie, Grandview, Weatherford)

- Created in 1953 from parts of **214** and **915**.
- Split in 1957 to create part of **806**.
- Three-way split in 1997 created the new **254** and **940**.
- Overlain by **682** in 2000.

818: California: the San Fernando Valley of Los Angeles County, including Burbank, Canoga Park, Encino, Glendale, North Hollywood, Northridge, Panorama City, Reseda, San Fernando, Sylmar, Tarzana, Van Nuys, and

Woodland Hills.)

- Created in 1984 by split from **213**.
- Split in 1997 to create **626**.
- Overlain by **747** in 2009.

819: Province of Quebec (Abitibi-Témiscamingue, Estrie, Mauricie, Outaouais, Sherbrooke, and Trois-Rivieres.)

- Created in 1957 by splitting it off from **514**.
- Split in 1997 to form part of **867**.

820–821: not used

822: reserved for future toll-free expansion (see also **800, 833, 844, 855, 866, 877, 888**, and **880–882** in this list)

823–824: not used

825: reserved for future usage in Alberta.

826–827: not used

828: North Carolina (Asheville, Franklin, Hickory, Murphy, Waynesville, and parts of western North Carolina)

- Created in 1998 by splitting it off from **704**.

829: the Dominican Republic.

- Overlain on **809** in 2005.

830: Texas (Del Rio, Kerrville, Eagle Pass, Fredericksburg, New Braunfels, and part of the Rio Grande Valley)

- Created in 1997 by splitting it off from **210**.

831: California (Monterey, Salinas, Monterey County, Hollister, and Santa Cruz)

- Created in 1998 by splitting it off from **408**.

832: Texas

- Overlain on **713** and **281** in 1999.

833: reserved for future toll-free expansion (see also **800, 822, 844, 855, 866, 877, 888**, and **880–882** in this list)

834: not used

835 was to be an overlay for **610** and **484** (Pennsylvania), but was canceled in 2008.

836–842: not used

843: South Carolina (Charleston, Florence, Hilton Head Island, Myrtle Beach, Charleston Air Force Base, and most of southeastern South Carolina)

- Created in 1998 by splitting it from **803**.

844: reserved for future toll-free expansion (see also **800, 822, 833, 855, 866, 877, 888**, and **880–882** in this list)

845: New York State (Dutchess, Orange, Putnam, Rockland, Sullivan, and Ulster Counties in southeastern New York State. Includes: Poughkeepsie, Middletown, Kingston, West Point, Newburgh, and the Catskill Mountains west to Margaretville)

- Created in 2000 by splitting it off from **914**.

846: not used

847: Illinois (Arlington Heights, Buffalo Grove, Des Plaines, Skokie, Waukegon, etc., near Chicago

- Created in 1996 by split from **708**.
- Overlain by **224** in 1998.

848: New Jersey

- Overlain on **732** in 2001.

849: the Dominican Republic

- Overlain on **809** and **829** in 2009

850: northwestern Florida (Appalachicola, Pensacola, Tallahassee, Quincy, Panama City, Naval Air Station Pensacola, Eglin Air Force Base, and all of the Florida Panhandle)

- Created in 1997 by split from **904**.

851–854: not used

855: Toll-free telephone service (see also **800, 822, 833, 844, 866, 877, 888**, and **880–882** in this list)

856: New Jersey (Cherry Hill, Camden, Millville, Vineland, and most of southwestern New Jersey)

- Created in 1999 by split from **609**.

857: Massachusetts (Boston, Cambridge, Quincy, Newton, Everett, and the nearby suburbs)

- Overlain on **617** in 2001.

858: Southern California: the northern half of San Diego, California city and some northern suburbs)

- Created in 1999 by splitting it off from **619**.

859: Kentucky (Lexington, Richmond, Danville, Covington, Florence, and northernmost Kentucky)

- Created in 2000 by split from **606**.
- Mnemonic: *UKY*—the University of Kentucky is located in Lexington, the largest city served by this code.

860: Connecticut (Hartford, Bristol, Norwich, and northern and eastern Connecticut)

- Created in 1995 by splitting it off from **203**.

- To be overlain by **959** in or about 2011.

861: not used

862: New Jersey

- Overlain on **973** in 2001.

863: Florida (Lakeland, Bartow, Sebring, Winter Haven in south-central Florida)

- Created in 1999 by split from **941**.

864: South Carolina (Greenville, Spartanburg, Anderson, Clemson, and most of northwestern South Carolina)

- Created in 1995 by splitting it off from **803**.

865: Tennessee (Knoxville, Alcoa, Athens, Clinton, Crossville, Dayton, Gatlinburg, Loudon, Maryville, Newport, Oak Ridge, Oliver Springs, Pigeon Forge, Rockwood, Sweetwater, etc., in East Tennessee)

- Created in 1999 by split from **423**.
- Mnemonic: *VOL*—the University of Tennessee, whose sports teams are the "Volunteers", is in Knoxville, the largest city served by this code.

866: Toll-free telephone service (see also **800, 822, 833, 844, 855, 877, 888**, and **880–882** in this list)

867: Yukon, Northwest Territories, and Nunavut

- Created in 1997 from parts of **403** and **819**.
- Mnemonics: *TOP of the world*. 1867 was the year of Canada's confederation (formation) (long-distance calls to the 867 area code must begin 1-867).

868: Trinidad and Tobago (all)

- Created in 1997 by split from **809**.
- Mnemonic: *TNT*

869: Saint Kitts and Nevis (all)

- Created in 1996 by split from **809**.

870: Arkansas (Texarkana, Jonesboro, Pine Bluff, and southern, eastern, and northeastern Arkansas)

- Created in 1997 by being split off from **501**.
- To be overlain by **327** in 2013.

871: not used

872: Illinois—overlay for **312** and **773**, entered service on November 7, 2009[11]

873: Province of Quebec—overlay of **819**, will enter service June 1, 2013[12]

874–875: not used

876: Jamaica (all)

- Created in 1997 by split from **809**.

877: Toll-free telephone service (see also **800, 822, 833, 844, 855, 866, 888**, and **880–882** in this list)

878: Pennsylvania

- Overlain on **412** and **area code 724** in 2001.

879: not used

880–882: Codes 880 through 882 were previously used to allow international customers to access toll-free numbers they otherwise could not by paying the international portion of the toll. 880 was paired with 800, 881 with 888, and 882 with 877.[13] As of 2008, they were reserved for toll-free area codes, but were not yet in use.

883–887: not used

888: Toll-free telephone service (see also **800, 822, 833, 844, 855, 866, 877**, and **880–882** in this list)

889: not used

890–899: Reserved for potential North American Numbering Plan expansion

900-999

900: premium-rate telephone numbers

901: Tennessee (Memphis, Covington, Gremantown, Somerville, and extreme southwestern Tennessee, mostly in Shelby County).

- Originally covered all of Tennessee; split to create **615** (1954) and **731** (2001).

902: Nova Scotia and Prince Edward Island

- Originally also covered New Brunswick; split in 1955 to create **506**.

903: Texas (Tyler, Sherman, Longview, Marshall, Palestine, Jacksonville, Carthage, and Northeast Texas)

- Created in 1990 by split from **214**.
- Overlain by **430** in 2003.

904:Florida (Jacksonville, St. Augustine, Starke, and most of northeastern Florida)

- Created in 1965 by split from **305**.
- Split to create **352** (1995) and **850** (1997).

905: Province of Ontario (Niagara Falls Region, Hamilton, St. Catharines, suburbs of the Greater Toronto Area, and southeastern Ontario)

- Created in 1993 by splitting it off from **416**.
- Overlain by **289** in 2001.

- Overlain by **365** by 2013.

906: Michigan: Upper Peninsula of Michigan: (Sault Ste. Marie, Escanaba, Houghton, Iron Mountain, Marquette, Menominee, etc.

- Created in 1961 by split from **616**.

907: all of Alaska excluding the lone town of Hyder

908: New Jersey (Alpha, Washington, Elizabeth, Warren, Plainfield, and west-central New Jersey)

- Created in 1991 by splitting it from **201**.
- Split in 1997 to create **732**.

909: California (western portions of San Bernardino County, and parts of Los Angeles County and Riverside County, including Fontana, Pomona, Chino Hills, Claremont, Chino, Ontario, Redlands) Pomona is in Los Angeles County, and it is the home of the county fair.

- Created in 1992 by split from **714**.
- Split in 2004 to create **951**.

910: North Carolina (Fayetteville, Wilmington, Fort Bragg, Seymour Johnson Air Force Base, Jacksonville, Lumberton and much of southeastern North Carolina)

- Created in 1993 by splitting it off from **919**.
- Split in 1997 to create **336**.

911: Emergency Services

912: Georgia (Savannah, Statesboro, Vidalia, Waycross, Brunswick, Douglas, and southeastern Georgia)

- Created in 1954 by split from **404** to give Georgia two area codes.
- Split in 2000 to create **229** and **478**.

913: Kansas (Kansas City, Kansas, Overland Park, Leavenworth, Ft. Leavenworth, parts of eastern Kansas)

- Split in 1997 to create **785**.

914: New York State (Westchester County)

- Split to create **516** in 1951 and **845** in the year 2000.

915: Texas (all of El Paso County and portions of Hudspeth County)

- Split to create part of **817** in 1953 and part of **806** in 1957.
- Split in 2003 to create **325** and **432**.

916: California (the Sacramento Metropolitan Area)

- Originally covered about one-third of California. It was split in 1958 to create **209** and in 1997 to create **530**.

917: New York State (New York City; cellular telephones)

Page 1383
T-Mobile Exhibit 1002
Part 4 of 6

- Overlain on **212** and **718** in 1992.

918: Oklahoma (Tulsa, Bartlesville, McAlester, Muskogee, Henryetta and northeastern Oklahoma)

- Created in 1953 by split from **405**, and to give Oklahoma two area codes, one centered on Oklahoma City and the other centered on Tulsa
- Overlain by **539** in 2011.

919: North Carolina (Raleigh, the state capital city, Durham, Chapel Hill, Goldsboro, and part of north-central North Carolina)

- Created in 1954 by splitting it from **704** to give North Carolina two area codes.
- Split in 1998 to create **252**.

920: Wisconsin (Appleton, Sheboygan, Oshkosh, Green Bay, Manitowoc, Marquette, Fond du Lac, and parts of eastern Wisconsin—but not Milwaukee County)

- Created in 1997 by splitting it off from **414**.
- To be overlain by **274** in 2010.

921–924: not used, yet

925: California (Livermore, Concord, Pleasant Hill, Walnut Creek, Martinez, Pleasanton and Dublin, just east of San Francisco Bay. Much of this area is served by the Bay Area Rapid Transit system, including Concord.)

- Created in 1998 by splitting it off from **510**.

926–927: not used, yet

928: Arizona (Flagstaff, Kingman, Lake Havasu City, Page, Payson, Prescott, Tuba City, Williams, Winslow, Window Rock, Yuma, and most of northern and western Arizona, including the Grand Canyon National Park, the Lower Colorado River Valley, and the Navajo Nation)

- Created in 2001 by split from **520**.

929: New York

- Overlain on **347**, **718** and part of **917** in 2011.

930: not used

931: Tennessee (Middle Tennessee, excluding the Nashville Metropolitan Area, which is in 615: Clarksville, Manchester, Columbia, Cookeville, Sparta)

- Created in 1997 by split from **615**.

932–934: not used

935 is assigned for numbering relief to **619** (San Diego, but no date has been scheduled for this to go into effect.)

936: Texas (Nacogdoches, Lufkin, Conroe, Huntsville, Center, and Southeast Texas)

- Created in 2000 by split from **409**.

937: Ohio (Dayton, Marysville, Springfield, Hillsboro, and southwestern Ohio excluding the Cincinnati Metropolitan Area)

- Created in 1996 by split from **513**.

938: Alabama (Huntsville, Anniston, Cullman, Decatur, Florence, Ft. Payne, Gadsden, Madison, Sheffield, Tuscumbia)

- Overlain on **256** in 2010.

939: Puerto Rico

- Overlain on **787** in 2001.

940: Texas (Denton, Wichita Falls, Decatur, Gainesville, Vernon, etc.) This is Texas immediately north of the Dallas–Fort Worth metroplex

- Created in 1997 by split from **817**.

941: Florida (Gulf Coast immediately south of Tampa Bay: all of Manatee County, Sarasota County, and Charlotte County; includes Bradenton, Port Charlotte, Sarasota, and Punta Gorda.)

- Created in 1995 by split from **813**.
- Split in 1999 to create **863**.

942–946: not used

947: Michigan

- Overlain on **248** in 2002.

948: not used

949: California (Irvine, Lake Forest, California, Newport Beach, Corona del Mar, and part of southern Orange County)

- Created in 1998 by split from **714**.

950: Used internally for carrier access

951: California (western Riverside County, including Riverside, Corona, Lake Elsinore, Mira Loma, Moreno Valley, Sun City, Temecula, Winchester, Lakeview, etc.)

- Created in 2000 by split from **909**.

952: Minnesota (Bloomington, Eden Prairie, Edina, Minnetonka, Chaska, and the southwest suburban Minneapolis area)

- Created in 2000 (along with **763**) by three-way split from **612**.

953: not used

954: Florida (all of Broward County: Fort Lauderdale, Hollywood, Coral Springs, etc.)

- Created in 1995 by splitting it from **305**.
- Overlain by **754** in 2002.

955: not used

956: Texas (Laredo, Harlingen, Brownsville, McAllen, and southmost Texas)

- Created in 1997 by splitting it from **210**.

957–958: not used

959: Connecticut

- Planned to be overlaid on **860** in or after 2011.

960–969: not used (96x block reserved in case consecutive numbers are ever needed)

970: Colorado (Grand Junction, Aspen, Durango, Estes Park, Fort Collins, Frisco, Glenwood Springs, Greeley, Purgatory, Steamboat Springs, Telluride, Vail, Rocky Mountain National Park, Dinosaur National Monument, and most of north-central, south-central, and western Colorado)

- Created in 1995 by splitting it off from **303**.

971: Oregon (Portland, Salem, Hillsboro, Beaverton, and most of northwestern Oregon)

- Partially overlain on **503** in 2000. Clatsop County and Tillamook County, originally excluded from the overlay, were added to it in 2008.

972: Texas

- Created in 1996 by a fission of **214**. In 1999, this fission was reversed to become an overlay, and a second overlay area code of **469** was added.

973: New Jersey (Newark, Paterson, and most of northeastern New Jersey)

- Created in 1997 by split from **201**.
- Overlain by **862**.

974: not used

975 is assigned for numbering relief to **816** (Missouri) but no date has been scheduled for this to go into effect.

976–977: not used

978: Massachusetts (Fitchburg, Peabody, Swampscott, and northeastern Massachusetts)

- Created in 1997 by splitting it off from **508**.
- Overlain by **351** in 2001.

979: Texas (Wharton, Bryan, Bay City, College Station, Lake Jackson, La Grange, and southeastern Texas)

- Created in 2000 by splitting it off from **409**.

980: North Carolina

- Overlain on **704** in 2001.

981–983: not used

984 is assigned for overlay relief to **919** (North Carolina) but no date has been scheduled for this to go into effect. (Twenty thousands blocks have been assigned to a switch in Raleigh, North Carolina.)[3]

985: Louisiana (Houma, Slidell, and southeastern Louisiana excluding New Orleans)

- Created in 2001 by split from **504**.

986–988: not used

989: Michigan (Alpena, Mt. Pleasant, Bay City, Saginaw, Midland, Owosso, and part of central Michigan)

- Created in 2000 by split from **517**.

990–999: Reserved for potential North American Numbering Plan expansion

By state, province or country

Future area codes are written in *italic*.

United States

State	Codes
Alabama	205, 251, 256, 334, *659*, 938
Alaska	907, 250
Arizona	480, 520, 602, 623, 928
Arkansas	*327*, 479, 501, 870
California	209, 213, 310, 323, *341*, *369*, 408, 415, 424, 442, 510, 530, 559, 562, 619, 626, *627*, *628*, 650, 657, 661, 669, 707, 714, 747, 760, *764*, 805, 818, 831, 858, 909, 916, 925, *935*, 949, 951 *(main article)*
Colorado	303, 719, 720, 970 *(main article)*
Connecticut	203, 475, 860, *959*
Delaware	302
District of Columbia	202
Florida	239, 305, 321, 352, 386, 407, 561, *689*, 727, 754, 772, 786, 813, 850, 863, 904, 941, 954 *(main article)*
Georgia	229, 404, 470, 478, 678, 706, 762, 770, 912 *(main article)*
Hawaii	808
Idaho	208

Illinois	217, 224, 309, 312, 331, *447*, *464*, 618, 630, 708, *730*, 773, 779, 815, 847, 872
Indiana	219, 260, 317, 574, 765, 812
Iowa	319, 515, 563, 641, 712 *(main article)*
Kansas	316, 620, 785, 913
Kentucky	270, *364*, 502, 606, 859
Louisiana	225, 318, 337, 504, 985
Maine	207
Maryland	*227*, 240, 301, 410, 443, *667*
Massachusetts	339, 351, 413, 508, 617, 774, 781, 857, 978 *(main article)*
Michigan	231, 248, 269, 313, 517, 586, 616, *679*, 734, 810, 906, 947, 989 *(main article)*
Minnesota	218, 320, 507, 612, 651, 763, 952
Mississippi	228, 601, 662, 769
Missouri	314, 417, *557*, 573, 636, 660, 816, 975
Montana	406
Nebraska	308, 402, 531
Nevada	702, 775
New Hampshire	603
New Jersey	201, 551, 609, 732, 848, 856, 862, 908, 973 *(main article)*
New Mexico	505, 575
New York	212, 315, 347, 516, 518, 585, 607, 631, 646, 716, 718, 845, 914, 917, 929 *(main article)*
North Carolina	252, 336, 704, 828, 910, 919, 980, *984 (main article)*
North Dakota	701
Ohio	216, 234, 283, 330, 380, 419, 440, 513, 567, 614, 740, 937
Oklahoma	405, 539, 580, 918
Oregon	458, 503, 541, 971
Pennsylvania	215, 267, *272*, 412, 445, 484, 570, *582*, 610, 717, 724, 814, *835*, 878 *(main article)*
Rhode Island	401
South Carolina	803, 843, 864
South Dakota	605
Tennessee	423, 615, 731, 865, 901, 931
Texas	210, 214, 254, 281, 325, 361, 409, 430, 432, 469, 512, 682, 713, *737*, 806, 817, 830, 832, 903, 915, 936, 940, 956, 972, 979 *(main article)*
Utah	385, 435, 801
Vermont	802
Virginia	276, 434, 540, 571, 703, 757, 804 *(main article)*
Washington	206, 253, 360, 425, 509, *564*
West Virginia	304, 681

Wisconsin	262, *274*, 414, 534, 608, 715, 920
Wyoming	307

Canada

Province / Territory	Codes (*italics not yet active*)
Alberta	403, 587, 780, 825
British Columbia	*236*, 250, 604, *672*, 778
Manitoba	204, *431*
New Brunswick	506
Newfoundland and Labrador	709
Nova Scotia	902
Ontario	226, 249, 289, 343, *365*, 416, *437*, 519, 613, 647, 705, 807, 905
Prince Edward Island	902
Quebec	418, 438, 450, 514, 579, 581, 819, *873*
Saskatchewan	306, *639*
Yukon, Northwest Territories, and Nunavut	867

Caribbean and Bermuda

Main article: Area codes in the Caribbean

Nation / Territory	Codes
Anguilla	264
Antigua and Barbuda	268
The Bahamas	242
Barbados	246
Bermuda	441
British Virgin Islands	284
Cayman Islands	345
Dominica	767
Dominican Republic	809, 829, 849
Grenada	473
Jamaica	876
Montserrat	664
Puerto Rico	787, 939
Saint Kitts and Nevis	869
Saint Lucia	758

Saint Vincent and the Grenadines	784
Sint Maarten	721
Trinidad and Tobago	868
Turks and Caicos Islands	649
U.S. Virgin Islands	340

U.S. Pacific Territories

Territory	Codes
American Samoa	684
Guam	671
Northern Marianas Islands	670

NANP non-geographic

Use	Codes
Canada special services	600
Inbound international	456
Interexchange carrier services	700
Personal Communications Service	500, *522*, 533, 544, *566*, *577*, *588*
Premium call services	900
Toll-free	800, *822*, *833*, *844*, 855, 866, 877, *880*, *881*, *882*, 888
US government	710

See also

- Bell System
- List of country calling codes
- List of future North American area codes
- List of original area codes

References

1. ^ "Telecom Decision CRTC 2011-451" (http://www.crtc.gc.ca/eng/archive/2011/2011-451.htm) . Canadian Radio-television and Telecommunications Commission. 28 July 2011. http://www.crtc.gc.ca/eng/archive/2011/2011-451.htm. Retrieved 29 July 2011.
2. ^ "?" (http://www.nanpa.com/nas/public/plannedNpasNotInServiceReport.do?method=displayPlannedNpasNotInServiceReport) . http://www.nanpa.com/nas/public/plannedNpasNotInServiceReport.do?method=displayPlannedNpasNotInServiceReport.
3. ^ [a] [b] [c] [d] NECA Tariff 4 filing, May 9, 2012.
4. ^ [a] [b] "639 to be Sask.'s 2nd area code" (http://www.cbc.ca/news/canada/saskatchewan/story/2011/06/14/sk-area-code-changes.html) . *CBC News*. 14 June 2011. http://www.cbc.ca/news/canada/saskatchewan/story/2011/06/14/sk-area-code-changes.html.

5. ^ "Telecom Decision CRTC 2010-213" (http://www.crtc.gc.ca/eng/archive/2010/2010-213.htm) . Canadian Radio-television and Telecommunications Commission. 13 April 2010. http://www.crtc.gc.ca/eng/archive /2010/2010-213.htm. Retrieved 23 August 2010.
6. ^ "?" (http://www.nanpa.com/nas/public/npa_query_step2.do?method=displayNpa& npaReportModel.nasNpaId=470) . http://www.nanpa.com/nas/public/npa_query_step2.do?method=displayNpa& npaReportModel.nasNpaId=470.
7. ^ "Personal Communications Service (NANP): Numbering Resources-5XX" (http://www.nanpa.com /number_resource_info/500_codes.html) . http://www.nanpa.com/number_resource_info/500_codes.html. Retrieved 2012-04-06.
8. ^ [1] (http://www.crtc.gc.ca/eng/archive/2011/2011-451.htm)
9. ^ "PL-423: Introduction of NPA 721 (Sint Maarten)" (http://www.nanpa.com/pdf/PL_423.pdf) . North American Numbering Plan Administration. 2011-07-27. http://www.nanpa.com/pdf/PL_423.pdf. Retrieved 2011-09-01.
10. ^ "?" (http://www.crtc.gc.ca/eng/archive/2010/2010-213.htm) . http://www.crtc.gc.ca/eng/archive /2010/2010-213.htm.
11. ^ Rosenthal, Phil (2009-08-10). "Chicago's newest area code: 872" (http://www.chicagotribune.com/business /chi-biz-area-code-aug10,0,5935115.story) . *Chicago Tribune*. http://www.chicagotribune.com/business/chi-biz-area-code-aug10,0,5935115.story.
12. ^ "Telecom Decision CRTC 2010-94" (http://www.crtc.gc.ca/eng/archive/2010/2010-94.htm) . Canadian Radio-television and Telecommunications Commission. http://www.crtc.gc.ca/eng/archive/2010/2010-94.htm. Retrieved 23 August 2010.
13. ^ "Planning Letter 331: Recovery of NPA Codes 880, 881, 882 (Paid Toll-Free Service) Scheduled for April 1, 2004" (http://www.nanpa.com/pdf/PL_331_v2.pdf) . North American Numbering Plan Administration. 2003-06-02. http://www.nanpa.com/pdf/PL_331_v2.pdf. Retrieved 2008-08-28.

Retrieved from "http://en.wikipedia.org /w/index.php?title=List_of_North_American_Numbering_Plan_area_codes&oldid=496144587"
Categories: North American Numbering Plan | Communication-related lists

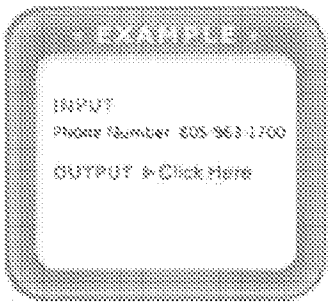
- This page was last modified on 5 June 2012 at 17:45.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. See Terms of use for details.
  Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

DOTS Phone Exchange

Verifies the Telco Exchange of any North American Number

DOTS Phone Exchange instantly recognizes whether a telephone number is valid and acceptable by verifying the first seven digits (NPA/NXX+1) of any telephone number combination in the North American Numbering Plan.

FREE TRIAL 30-DAY USE



Overview | Features & Benefits | Security | Pricing | Developer's Tools

What is it?

DOTS Phone Exchange (PE) is a programmable XML Web Service hosted by Service Objects which allows you to improve the quality of your contact database by eliminating invalid telephone numbers at the point-of-entry. DOTS Phone Exchange℠ allows you to integrate telephone information into your Web site, CRM database, and other business contact dependent processes.

DOTS Phone Exchange can instantly recognize whether a telephone number is valid and acceptable by verifying the first seven digits (NPA/NXX+1) of any telephone number combination in the North American Numbering Plan.

©2001-2012 Service Objects, Inc. | 133 East De La Guerra St., Suite 10 | Santa Barbara, CA 93101 | 1.805.963.1700 | toll free: 1.8

Network Working Group
Request for Comments: 3261
Obsoletes: 2543
Category: Standards Track

J. Rosenberg
dynamicsoft
H. Schulzrinne
Columbia U.
G. Camarillo
Ericsson
A. Johnston
WorldCom
J. Peterson
Neustar
R. Sparks
dynamicsoft
M. Handley
ICIR
E. Schooler
AT&T

June 2002

SIP: Session Initiation Protocol

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2002). All Rights Reserved.

Abstract

This document describes Session Initiation Protocol (SIP), an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences.

SIP invitations used to create sessions carry session descriptions that allow participants to agree on a set of compatible media types. SIP makes use of elements called proxy servers to help route requests to the user's current location, authenticate and authorize users for services, implement provider call-routing policies, and provide features to users. SIP also provides a registration function that allows users to upload their current locations for use by proxy servers. SIP runs on top of several different transport protocols.

Table of Contents

1 Introduction 8
2 Overview of SIP Functionality 9
3 Terminology 10
4 Overview of Operation 10
5 Structure of the Protocol 18
6 Definitions 20
7 SIP Messages 26
7.1 Requests 27
7.2 Responses 28
7.3 Header Fields 29
7.3.1 Header Field Format 30
7.3.2 Header Field Classification 32
7.3.3 Compact Form 32
7.4 Bodies 33
7.4.1 Message Body Type 33
7.4.2 Message Body Length 33
7.5 Framing SIP Messages 34
8 General User Agent Behavior 34
8.1 UAC Behavior 35
8.1.1 Generating the Request 35
8.1.1.1 Request-URI 35
8.1.1.2 To 36
8.1.1.3 From 37
8.1.1.4 Call-ID 37
8.1.1.5 CSeq 38
8.1.1.6 Max-Forwards 38
8.1.1.7 Via 39
8.1.1.8 Contact 40
8.1.1.9 Supported and Require 40
8.1.1.10 Additional Message Components 41
8.1.2 Sending the Request 41
8.1.3 Processing Responses 42
8.1.3.1 Transaction Layer Errors 42
8.1.3.2 Unrecognized Responses 42
8.1.3.3 Vias 43
8.1.3.4 Processing 3xx Responses 43
8.1.3.5 Processing 4xx Responses 45
8.2 UAS Behavior 46
8.2.1 Method Inspection 46
8.2.2 Header Inspection 46
8.2.2.1 To and Request-URI 46
8.2.2.2 Merged Requests 47
8.2.2.3 Require 47
8.2.3 Content Processing 48
8.2.4 Applying Extensions 49
8.2.5 Processing the Request 49

8.2.6 Generating the Response 49
8.2.6.1 Sending a Provisional Response 49
8.2.6.2 Headers and Tags 50
8.2.7 Stateless UAS Behavior 50
8.3 Redirect Servers 51
9 Canceling a Request 53
9.1 Client Behavior 53
9.2 Server Behavior 55
10 Registrations 56
10.1 Overview 56
10.2 Constructing the REGISTER Request 57

10.2.1 Adding Bindings 59
10.2.1.1 Setting the Expiration Interval of Contact Addresses 60
10.2.1.2 Preferences among Contact Addresses 61
10.2.2 Removing Bindings 61
10.2.3 Fetching Bindings 61
10.2.4 Refreshing Bindings 61
10.2.5 Setting the Internal Clock 62
10.2.6 Discovering a Registrar 62
10.2.7 Transmitting a Request 62
10.2.8 Error Responses 63
10.3 Processing REGISTER Requests 63
11 Querying for Capabilities 66
11.1 Construction of OPTIONS Request 67
11.2 Processing of OPTIONS Request 68
12 Dialogs 69
12.1 Creation of a Dialog 70
12.1.1 UAS behavior 70
12.1.2 UAC Behavior 71
12.2 Requests within a Dialog 72
12.2.1 UAC Behavior 73
12.2.1.1 Generating the Request 73
12.2.1.2 Processing the Responses 75
12.2.2 UAS Behavior 76
12.3 Termination of a Dialog 77
13 Initiating a Session 77
13.1 Overview 77
13.2 UAC Processing 78
13.2.1 Creating the Initial INVITE 78
13.2.2 Processing INVITE Responses 81
13.2.2.1 1xx Responses 81
13.2.2.2 3xx Responses 81
13.2.2.3 4xx, 5xx and 6xx Responses 81
13.2.2.4 2xx Responses 82
13.3 UAS Processing 83
13.3.1 Processing of the INVITE 83
13.3.1.1 Progress 84
13.3.1.2 The INVITE is Redirected 84

13.3.1.3 The INVITE is Rejected 85
13.3.1.4 The INVITE is Accepted 85
14 Modifying an Existing Session 86
14.1 UAC Behavior 86
14.2 UAS Behavior 88
15 Terminating a Session 89
15.1 Terminating a Session with a BYE Request 90
15.1.1 UAC Behavior 90
15.1.2 UAS Behavior 91
16 Proxy Behavior 91
16.1 Overview 91
16.2 Stateful Proxy 92
16.3 Request Validation 94
16.4 Route Information Preprocessing 96
16.5 Determining Request Targets 97
16.6 Request Forwarding 99
16.7 Response Processing 107
16.8 Processing Timer C 114
16.9 Handling Transport Errors 115
16.10 CANCEL Processing 115

16.11 Stateless Proxy 116
16.12 Summary of Proxy Route Processing 118
16.12.1 Examples 118
16.12.1.1 Basic SIP Trapezoid 118
16.12.1.2 Traversing a Strict-Routing Proxy 120
16.12.1.3 Rewriting Record-Route Header Field Values 121
17 Transactions 122
17.1 Client Transaction 124
17.1.1 INVITE Client Transaction 125
17.1.1.1 Overview of INVITE Transaction 125
17.1.1.2 Formal Description 125
17.1.1.3 Construction of the ACK Request 129
17.1.2 Non-INVITE Client Transaction 130
17.1.2.1 Overview of the non-INVITE Transaction 130
17.1.2.2 Formal Description 131
17.1.3 Matching Responses to Client Transactions 132
17.1.4 Handling Transport Errors 133
17.2 Server Transaction 134
17.2.1 INVITE Server Transaction 134
17.2.2 Non-INVITE Server Transaction 137
17.2.3 Matching Requests to Server Transactions 138
17.2.4 Handling Transport Errors 141
18 Transport 141
18.1 Clients 142
18.1.1 Sending Requests 142
18.1.2 Receiving Responses 144
18.2 Servers 145
18.2.1 Receiving Requests 145

18.2.2 Sending Responses 146
18.3 Framing 147
18.4 Error Handling 147
19 Common Message Components 147
19.1 SIP and SIPS Uniform Resource Indicators 148
19.1.1 SIP and SIPS URI Components 148
19.1.2 Character Escaping Requirements 152
19.1.3 Example SIP and SIPS URIs 153
19.1.4 URI Comparison 153
19.1.5 Forming Requests from a URI 156
19.1.6 Relating SIP URIs and tel URLs 157
19.2 Option Tags 158
19.3 Tags 159
20 Header Fields 159
20.1 Accept 161
20.2 Accept-Encoding 163
20.3 Accept-Language 164
20.4 Alert-Info 164
20.5 Allow 165
20.6 Authentication-Info 165
20.7 Authorization 165
20.8 Call-ID 166
20.9 Call-Info 166
20.10 Contact 167
20.11 Content-Disposition 168
20.12 Content-Encoding 169
20.13 Content-Language 169
20.14 Content-Length 169
20.15 Content-Type 170

20.16 CSeq 170
20.17 Date 170
20.18 Error-Info 171
20.19 Expires 171
20.20 From 172
20.21 In-Reply-To 172
20.22 Max-Forwards 173
20.23 Min-Expires 173
20.24 MIME-Version 173
20.25 Organization 174
20.26 Priority 174
20.27 Proxy-Authenticate 174
20.28 Proxy-Authorization 175
20.29 Proxy-Require 175
20.30 Record-Route 175
20.31 Reply-To 176
20.32 Require 176
20.33 Retry-After 176
20.34 Route 177

20.35 Server 177
20.36 Subject 177
20.37 Supported 178
20.38 Timestamp 178
20.39 To 178
20.40 Unsupported 179
20.41 User-Agent 179
20.42 Via 179
20.43 Warning 180
20.44 WWW-Authenticate 182
21 Response Codes 182
21.1 Provisional 1xx 182
21.1.1 100 Trying 183
21.1.2 180 Ringing 183
21.1.3 181 Call Is Being Forwarded 183
21.1.4 182 Queued 183
21.1.5 183 Session Progress 183
21.2 Successful 2xx 183
21.2.1 200 OK 183
21.3 Redirection 3xx 184
21.3.1 300 Multiple Choices 184
21.3.2 301 Moved Permanently 184
21.3.3 302 Moved Temporarily 184
21.3.4 305 Use Proxy 185
21.3.5 380 Alternative Service 185
21.4 Request Failure 4xx 185
21.4.1 400 Bad Request 185
21.4.2 401 Unauthorized 185
21.4.3 402 Payment Required 186
21.4.4 403 Forbidden 186
21.4.5 404 Not Found 186
21.4.6 405 Method Not Allowed 186
21.4.7 406 Not Acceptable 186
21.4.8 407 Proxy Authentication Required 186
21.4.9 408 Request Timeout 186
21.4.10 410 Gone 187
21.4.11 413 Request Entity Too Large 187
21.4.12 414 Request-URI Too Long 187

21.4.13 415 Unsupported Media Type 187
21.4.14 416 Unsupported URI Scheme 187
21.4.15 420 Bad Extension 187
21.4.16 421 Extension Required 188
21.4.17 423 Interval Too Brief 188
21.4.18 480 Temporarily Unavailable 188
21.4.19 481 Call/Transaction Does Not Exist 188
21.4.20 482 Loop Detected 188
21.4.21 483 Too Many Hops 189
21.4.22 484 Address Incomplete 189

21.4.23 485 Ambiguous 189
21.4.24 486 Busy Here 189
21.4.25 487 Request Terminated 190
21.4.26 488 Not Acceptable Here 190
21.4.27 491 Request Pending 190
21.4.28 493 Undecipherable 190
21.5 Server Failure 5xx 190
21.5.1 500 Server Internal Error 190
21.5.2 501 Not Implemented 191
21.5.3 502 Bad Gateway 191
21.5.4 503 Service Unavailable 191
21.5.5 504 Server Time-out 191
21.5.6 505 Version Not Supported 192
21.5.7 513 Message Too Large 192
21.6 Global Failures 6xx 192
21.6.1 600 Busy Everywhere 192
21.6.2 603 Decline 192
21.6.3 604 Does Not Exist Anywhere 192
21.6.4 606 Not Acceptable 192
22 Usage of HTTP Authentication 193
22.1 Framework 193
22.2 User-to-User Authentication 195
22.3 Proxy-to-User Authentication 197
22.4 The Digest Authentication Scheme 199
23 S/MIME 201
23.1 S/MIME Certificates 201
23.2 S/MIME Key Exchange 202
23.3 Securing MIME bodies 205
23.4 SIP Header Privacy and Integrity using S/MIME: Tunneling SIP 207
23.4.1 Integrity and Confidentiality Properties of SIP Headers 207
23.4.1.1 Integrity 207
23.4.1.2 Confidentiality 208
23.4.2 Tunneling Integrity and Authentication 209
23.4.3 Tunneling Encryption 211
24 Examples 213
24.1 Registration 213
24.2 Session Setup 214
25 Augmented BNF for the SIP Protocol 219
25.1 Basic Rules 219
26 Security Considerations: Threat Model and Security Usage Recommendations 232
26.1 Attacks and Threat Models 233
26.1.1 Registration Hijacking 233
26.1.2 Impersonating a Server 234
26.1.3 Tampering with Message Bodies 235

26.1.4 Tearing Down Sessions 235

26.1.5 Denial of Service and Amplification 236
26.2 Security Mechanisms 237
26.2.1 Transport and Network Layer Security 238
26.2.2 SIPS URI Scheme 239
26.2.3 HTTP Authentication 240
26.2.4 S/MIME 240
26.3 Implementing Security Mechanisms 241
26.3.1 Requirements for Implementers of SIP 241
26.3.2 Security Solutions 242
26.3.2.1 Registration 242
26.3.2.2 Interdomain Requests 243
26.3.2.3 Peer-to-Peer Requests 245
26.3.2.4 DoS Protection 246
26.4 Limitations 247
26.4.1 HTTP Digest 247
26.4.2 S/MIME 248
26.4.3 TLS 249
26.4.4 SIPS URIs 249
26.5 Privacy 251
27 IANA Considerations 252
27.1 Option Tags 252
27.2 Warn-Codes 252
27.3 Header Field Names 253
27.4 Method and Response Codes 253
27.5 The "message/sip" MIME type. 254
27.6 New Content-Disposition Parameter Registrations 255
28 Changes From RFC 2543 255
28.1 Major Functional Changes 255
28.2 Minor Functional Changes 260
29 Normative References 261
30 Informative References 262
A Table of Timer Values 265
Acknowledgments 266
Authors' Addresses 267
Full Copyright Statement 269

1 Introduction

There are many applications of the Internet that require the creation and management of a session, where a session is considered an exchange of data between an association of participants. The implementation of these applications is complicated by the practices of participants: users may move between endpoints, they may be addressable by multiple names, and they may communicate in several different media - sometimes simultaneously. Numerous protocols have been authored that carry various forms of real-time multimedia session data such as voice, video, or text messages. The Session Initiation Protocol (SIP) works in concert with these protocols by

enabling Internet endpoints (called user agents) to discover one another and to agree on a characterization of a session they would like to share. For locating prospective session participants, and for other functions, SIP enables the creation of an infrastructure of network hosts (called proxy servers) to which user agents can send registrations, invitations to sessions, and other requests. SIP is an agile, general-purpose tool for creating, modifying, and terminating sessions that works independently of underlying transport protocols and without dependency on the type of session that is being established.

2 Overview of SIP Functionality

SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls. SIP can also invite participants to already existing sessions, such as multicast conferences. Media can be added to (and removed from) an existing session. SIP transparently supports name mapping and redirection services, which supports personal mobility [27] - users can maintain a single externally visible identifier regardless of their network location.

SIP supports five facets of establishing and terminating multimedia communications:

- User location: determination of the end system to be used for communication;
- User availability: determination of the willingness of the called party to engage in communications;
- User capabilities: determination of the media and media parameters to be used;
- Session setup: "ringing", establishment of session parameters at both called and calling party;
- Session management: including transfer and termination of sessions, modifying session parameters, and invoking services.

SIP is not a vertically integrated communications system. SIP is rather a component that can be used with other IETF protocols to build a complete multimedia architecture. Typically, these architectures will include protocols such as the Real-time Transport Protocol (RTP) (RFC 1889 [28]) for transporting real-time data and providing QoS feedback, the Real-Time streaming protocol (RTSP) (RFC 2326 [29]) for controlling delivery of streaming media, the Media

Gateway Control Protocol (MEGACO) (RFC 3015 [30]) for controlling gateways to the Public Switched Telephone Network (PSTN), and the Session Description Protocol (SDP) (RFC 2327 [1]) for describing multimedia sessions. Therefore, SIP should be used in conjunction with other protocols in order to provide complete services to the users. However, the basic functionality and operation of SIP does not depend on any of these protocols.

SIP does not provide services. Rather, SIP provides primitives that

can be used to implement different services. For example, SIP can locate a user and deliver an opaque object to his current location. If this primitive is used to deliver a session description written in SDP, for instance, the endpoints can agree on the parameters of a session. If the same primitive is used to deliver a photo of the caller as well as the session description, a "caller ID" service can be easily implemented. As this example shows, a single primitive is typically used to provide several different services.

SIP does not offer conference control services such as floor control or voting and does not prescribe how a conference is to be managed. SIP can be used to initiate a session that uses some other conference control protocol. Since SIP messages and the sessions they establish can pass through entirely different networks, SIP cannot, and does not, provide any kind of network resource reservation capabilities.

The nature of the services provided make security particularly important. To that end, SIP provides a suite of security services, which include denial-of-service prevention, authentication (both user to user and proxy to user), integrity protection, and encryption and privacy services.

SIP works with both IPv4 and IPv6.

3 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [2] and indicate requirement levels for compliant SIP implementations.

4 Overview of Operation

This section introduces the basic operations of SIP using simple examples. This section is tutorial in nature and does not contain any normative statements.

The first example shows the basic functions of SIP: location of an end point, signal of a desire to communicate, negotiation of session parameters to establish the session, and teardown of the session once established.

Figure 1 shows a typical example of a SIP message exchange between two users, Alice and Bob. (Each message is labeled with the letter "F" and a number for reference by the text.) In this example, Alice uses a SIP application on her PC (referred to as a softphone) to call Bob on his SIP phone over the Internet. Also shown are two SIP proxy servers that act on behalf of Alice and Bob to facilitate the session establishment. This typical arrangement is often referred to as the "SIP trapezoid" as shown by the geometric shape of the dotted lines in Figure 1.

Alice "calls" Bob using his SIP identity, a type of Uniform Resource Identifier (URI) called a SIP URI. SIP URIs are defined in Section 19.1. It has a similar form to an email address, typically

containing a username and a host name. In this case, it is sip:bob@biloxi.com, where biloxi.com is the domain of Bob's SIP service provider. Alice has a SIP URI of sip:alice@atlanta.com. Alice might have typed in Bob's URI or perhaps clicked on a hyperlink or an entry in an address book. SIP also provides a secure URI, called a SIPS URI. An example would be sips:bob@biloxi.com. A call made to a SIPS URI guarantees that secure, encrypted transport (namely TLS) is used to carry all SIP messages from the caller to the domain of the callee. From there, the request is sent securely to the callee, but with security mechanisms that depend on the policy of the domain of the callee.

SIP is based on an HTTP-like request/response transaction model. Each transaction consists of a request that invokes a particular method, or function, on the server and at least one response. In this example, the transaction begins with Alice's softphone sending an INVITE request addressed to Bob's SIP URI. INVITE is an example of a SIP method that specifies the action that the requestor (Alice) wants the server (Bob) to take. The INVITE request contains a number of header fields. Header fields are named attributes that provide additional information about a message. The ones present in an INVITE include a unique identifier for the call, the destination address, Alice's address, and information about the type of session that Alice wishes to establish with Bob. The INVITE (message F1 in Figure 1) might look like this:

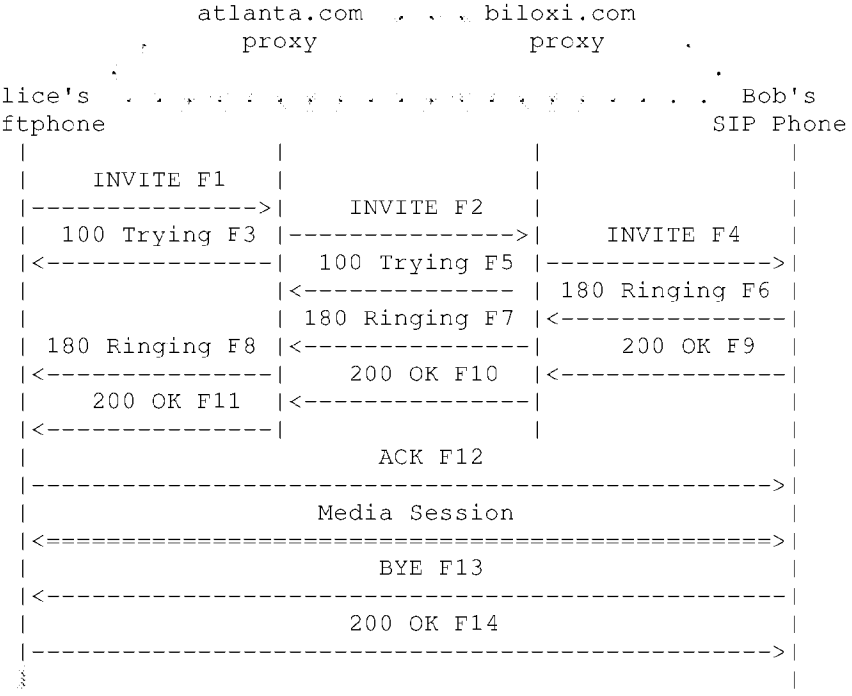


Figure 1: SIP session setup example with SIP trapezoid

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

The first line of the text-encoded message contains the method name (INVITE). The lines that follow are a list of header fields. This example contains a minimum required set. The header fields are briefly described below:

Via contains the address (pc33.atlanta.com) at which Alice is expecting to receive responses to this request. It also contains a branch parameter that identifies this transaction.

To contains a display name (Bob) and a SIP or SIPS URI (sip:bob@biloxi.com) towards which the request was originally directed. Display names are described in RFC 2822 [3].

From also contains a display name (Alice) and a SIP or SIPS URI (sip:alice@atlanta.com) that indicate the originator of the request. This header field also has a tag parameter containing a random string (1928301774) that was added to the URI by the softphone. It is used for identification purposes.

Call-ID contains a globally unique identifier for this call, generated by the combination of a random string and the softphone's host name or IP address. The combination of the To tag, From tag, and Call-ID completely defines a peer-to-peer SIP relationship between Alice and Bob and is referred to as a dialog.

CSeq or Command Sequence contains an integer and a method name. The CSeq number is incremented for each new request within a dialog and is a traditional sequence number.

Contact contains a SIP or SIPS URI that represents a direct route to contact Alice, usually composed of a username at a fully qualified domain name (FQDN). While an FQDN is preferred, many end systems do not have registered domain names, so IP addresses are permitted. While the Via header field tells other elements where to send the response, the Contact header field tells other elements where to send future requests.

Max-Forwards serves to limit the number of hops a request can make on the way to its destination. It consists of an integer that is decremented by one at each hop.

Content-Type contains a description of the message body (not shown).

Content-Length contains an octet (byte) count of the message body.

The complete set of SIP header fields is defined in Section 20.

The details of the session, such as the type of media, codec, or sampling rate, are not described using SIP. Rather, the body of a SIP message contains a description of the session, encoded in some other protocol format. One such format is the Session Description Protocol (SDP) (RFC 2327 [1]). This SDP message (not shown in the

example) is carried by the SIP message in a way that is analogous to a document attachment being carried by an email message, or a web page being carried in an HTTP message.

Since the softphone does not know the location of Bob or the SIP server in the biloxi.com domain, the softphone sends the INVITE to the SIP server that serves Alice's domain, atlanta.com. The address of the atlanta.com SIP server could have been configured in Alice's softphone, or it could have been discovered by DHCP, for example.

The atlanta.com SIP server is a type of SIP server known as a proxy server. A proxy server receives SIP requests and forwards them on behalf of the requestor. In this example, the proxy server receives the INVITE request and sends a 100 (Trying) response back to Alice's softphone. The 100 (Trying) response indicates that the INVITE has been received and that the proxy is working on her behalf to route the INVITE to the destination. Responses in SIP use a three-digit code followed by a descriptive phrase. This response contains the same To, From, Call-ID, CSeq and branch parameter in the Via as the INVITE, which allows Alice's softphone to correlate this response to the sent INVITE. The atlanta.com proxy server locates the proxy server at biloxi.com, possibly by performing a particular type of DNS (Domain Name Service) lookup to find the SIP server that serves the biloxi.com domain. This is described in [4]. As a result, it obtains the IP address of the biloxi.com proxy server and forwards, or proxies, the INVITE request there. Before forwarding the request, the atlanta.com proxy server adds an additional Via header field value that contains its own address (the INVITE already contains Alice's address in the first Via). The biloxi.com proxy server receives the INVITE and responds with a 100 (Trying) response back to the atlanta.com proxy server to indicate that it has received the INVITE and is processing the request. The proxy server consults a database, generically called a location service, that contains the current IP address of Bob. (We shall see in the next section how this database can be populated.) The biloxi.com proxy server adds another Via header field value with its own address to the INVITE and proxies it to Bob's SIP phone.

Bob's SIP phone receives the INVITE and alerts Bob to the incoming call from Alice so that Bob can decide whether to answer the call, that is, Bob's phone rings. Bob's SIP phone indicates this in a 180 (Ringing) response, which is routed back through the two proxies in the reverse direction. Each proxy uses the Via header field to determine where to send the response and removes its own address from the top. As a result, although DNS and location service lookups were

required to route the initial INVITE, the 180 (Ringing) response can be returned to the caller without lookups or without state being

maintained in the proxies. This also has the desirable property that each proxy that sees the INVITE will also see all responses to the INVITE.

When Alice's softphone receives the 180 (Ringing) response, it passes this information to Alice, perhaps using an audio ringback tone or by displaying a message on Alice's screen.

In this example, Bob decides to answer the call. When he picks up the handset, his SIP phone sends a 200 (OK) response to indicate that the call has been answered. The 200 (OK) contains a message body with the SDP media description of the type of session that Bob is willing to establish with Alice. As a result, there is a two-phase exchange of SDP messages: Alice sent one to Bob, and Bob sent one back to Alice. This two-phase exchange provides basic negotiation capabilities and is based on a simple offer/answer model of SDP exchange. If Bob did not wish to answer the call or was busy on another call, an error response would have been sent instead of the 200 (OK), which would have resulted in no media session being established. The complete list of SIP response codes is in Section 21. The 200 (OK) (message F9 in Figure 1) might look like this as Bob sends it out:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com
   ;branch=z9hG4bKnashds8;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
   ;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com
   ;branch=z9hG4bK776asdhds ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(Bob's SDP not shown)
```

The first line of the response contains the response code (200) and the reason phrase (OK). The remaining lines contain header fields. The Via, To, From, Call-ID, and CSeq header fields are copied from the INVITE request. (There are three Via header field values - one added by Alice's SIP phone, one added by the atlanta.com proxy, and one added by the biloxi.com proxy.) Bob's SIP phone has added a tag parameter to the To header field. This tag will be incorporated by both endpoints into the dialog and will be included in all future

requests and responses in this call. The Contact header field contains a URI at which Bob can be directly reached at his SIP phone. The Content-Type and Content-Length refer to the message body (not shown) that contains Bob's SDP media information.

In addition to DNS and location service lookups shown in this example, proxy servers can make flexible "routing decisions" to decide where to send a request. For example, if Bob's SIP phone returned a 486 (Busy Here) response, the biloxi.com proxy server could proxy the INVITE to Bob's voicemail server. A proxy server can also send an INVITE to a number of locations at the same time. This type of parallel search is known as forking.

In this case, the 200 (OK) is routed back through the two proxies and is received by Alice's softphone, which then stops the ringback tone and indicates that the call has been answered. Finally, Alice's softphone sends an acknowledgement message, ACK, to Bob's SIP phone to confirm the reception of the final response (200 (OK)). In this example, the ACK is sent directly from Alice's softphone to Bob's SIP phone, bypassing the two proxies. This occurs because the endpoints have learned each other's address from the Contact header fields through the INVITE/200 (OK) exchange, which was not known when the initial INVITE was sent. The lookups performed by the two proxies are no longer needed, so the proxies drop out of the call flow. This completes the INVITE/200/ACK three-way handshake used to establish SIP sessions. Full details on session setup are in Section 13.

Alice and Bob's media session has now begun, and they send media packets using the format to which they agreed in the exchange of SDP. In general, the end-to-end media packets take a different path from the SIP signaling messages.

During the session, either Alice or Bob may decide to change the characteristics of the media session. This is accomplished by sending a re-INVITE containing a new media description. This re-INVITE references the existing dialog so that the other party knows that it is to modify an existing session instead of establishing a new session. The other party sends a 200 (OK) to accept the change. The requestor responds to the 200 (OK) with an ACK. If the other party does not accept the change, he sends an error response such as 488 (Not Acceptable Here), which also receives an ACK. However, the failure of the re-INVITE does not cause the existing call to fail - the session continues using the previously negotiated characteristics. Full details on session modification are in Section 14.

At the end of the call, Bob disconnects (hangs up) first and generates a BYE message. This BYE is routed directly to Alice's softphone, again bypassing the proxies. Alice confirms receipt of the BYE with a 200 (OK) response, which terminates the session and the BYE transaction. No ACK is sent - an ACK is only sent in response to a response to an INVITE request. The reasons for this special handling for INVITE will be discussed later, but relate to

the reliability mechanisms in SIP, the length of time it can take for a ringing phone to be answered, and forking. For this reason, request handling in SIP is often classified as either INVITE or non-INVITE, referring to all other methods besides INVITE. Full details on session termination are in Section 15.

Section 24.2 describes the messages shown in Figure 1 in full.

In some cases, it may be useful for proxies in the SIP signaling path to see all the messaging between the endpoints for the duration of the session. For example, if the biloxi.com proxy server wished to remain in the SIP messaging path beyond the initial INVITE, it would add to the INVITE a required routing header field known as Record-Route that contained a URI resolving to the hostname or IP address of the proxy. This information would be received by both Bob's SIP phone and (due to the Record-Route header field being passed back in the 200 (OK)) Alice's softphone and stored for the duration of the dialog. The biloxi.com proxy server would then receive and proxy the ACK, BYE, and 200 (OK) to the BYE. Each proxy can independently decide to receive subsequent messages, and those messages will pass through all proxies that elect to receive it. This capability is frequently used for proxies that are providing mid-call features.

Registration is another common operation in SIP. Registration is one way that the biloxi.com server can learn the current location of Bob. Upon initialization, and at periodic intervals, Bob's SIP phone sends REGISTER messages to a server in the biloxi.com domain known as a SIP registrar. The REGISTER messages associate Bob's SIP or SIPS URI (sip:bob@biloxi.com) with the machine into which he is currently logged (conveyed as a SIP or SIPS URI in the Contact header field). The registrar writes this association, also called a binding, to a database, called the location service, where it can be used by the proxy in the biloxi.com domain. Often, a registrar server for a domain is co-located with the proxy for that domain. It is an important concept that the distinction between types of SIP servers is logical, not physical.

Bob is not limited to registering from a single device. For example, both his SIP phone at home and the one in the office could send registrations. This information is stored together in the location

service and allows a proxy to perform various types of searches to locate Bob. Similarly, more than one user can be registered on a single device at the same time.

The location service is just an abstract concept. It generally contains information that allows a proxy to input a URI and receive a set of zero or more URIs that tell the proxy where to send the request. Registrations are one way to create this information, but not the only way. Arbitrary mapping functions can be configured at the discretion of the administrator.

Finally, it is important to note that in SIP, registration is used for routing incoming SIP requests and has no role in authorizing outgoing requests. Authorization and authentication are handled in SIP either on a request-by-request basis with a challenge/response mechanism, or by using a lower layer scheme as discussed in Section

26.

The complete set of SIP message details for this registration example is in Section 24.1.

Additional operations in SIP, such as querying for the capabilities of a SIP server or client using OPTIONS, or canceling a pending request using CANCEL, will be introduced in later sections.

5 Structure of the Protocol

SIP is structured as a layered protocol, which means that its behavior is described in terms of a set of fairly independent processing stages with only a loose coupling between each stage. The protocol behavior is described as layers for the purpose of presentation, allowing the description of functions common across elements in a single section. It does not dictate an implementation in any way. When we say that an element "contains" a layer, we mean it is compliant to the set of rules defined by that layer.

Not every element specified by the protocol contains every layer. Furthermore, the elements specified by SIP are logical elements, not physical ones. A physical realization can choose to act as different logical elements, perhaps even on a transaction-by-transaction basis.

The lowest layer of SIP is its syntax and encoding. Its encoding is specified using an augmented Backus-Naur Form grammar (BNF). The complete BNF is specified in Section 25; an overview of a SIP message's structure can be found in Section 7.

The second layer is the transport layer. It defines how a client sends requests and receives responses and how a server receives requests and sends responses over the network. All SIP elements contain a transport layer. The transport layer is described in Section 18.

The third layer is the transaction layer. Transactions are a fundamental component of SIP. A transaction is a request sent by a client transaction (using the transport layer) to a server transaction, along with all responses to that request sent from the server transaction back to the client. The transaction layer handles application-layer retransmissions, matching of responses to requests, and application-layer timeouts. Any task that a user agent client (UAC) accomplishes takes place using a series of transactions. Discussion of transactions can be found in Section 17. User agents contain a transaction layer, as do stateful proxies. Stateless proxies do not contain a transaction layer. The transaction layer has a client component (referred to as a client transaction) and a server component (referred to as a server transaction), each of which are represented by a finite state machine that is constructed to process a particular request.

The layer above the transaction layer is called the transaction user (TU). Each of the SIP entities, except the stateless proxy, is a transaction user. When a TU wishes to send a request, it creates a

client transaction instance and passes it the request along with the destination IP address, port, and transport to which to send the request. A TU that creates a client transaction can also cancel it. When a client cancels a transaction, it requests that the server stop further processing, revert to the state that existed before the transaction was initiated, and generate a specific error response to that transaction. This is done with a CANCEL request, which constitutes its own transaction, but references the transaction to be cancelled (Section 9).

The SIP elements, that is, user agent clients and servers, stateless and stateful proxies and registrars, contain a core that distinguishes them from each other. Cores, except for the stateless proxy, are transaction users. While the behavior of the UAC and UAS cores depends on the method, there are some common rules for all methods (Section 8). For a UAC, these rules govern the construction of a request; for a UAS, they govern the processing of a request and generating a response. Since registrations play an important role in SIP, a UAS that handles a REGISTER is given the special name registrar. Section 10 describes UAC and UAS core behavior for the REGISTER method. Section 11 describes UAC and UAS core behavior for the OPTIONS method, used for determining the capabilities of a UA.

Certain other requests are sent within a dialog. A dialog is a peer-to-peer SIP relationship between two user agents that persists for some time. The dialog facilitates sequencing of messages and proper routing of requests between the user agents. The INVITE method is the only way defined in this specification to establish a dialog. When a UAC sends a request that is within the context of a dialog, it follows the common UAC rules as discussed in Section 8 but also the rules for mid-dialog requests. Section 12 discusses dialogs and presents the procedures for their construction and maintenance, in addition to construction of requests within a dialog.

The most important method in SIP is the INVITE method, which is used to establish a session between participants. A session is a collection of participants, and streams of media between them, for the purposes of communication. Section 13 discusses how sessions are initiated, resulting in one or more SIP dialogs. Section 14 discusses how characteristics of that session are modified through the use of an INVITE request within a dialog. Finally, section 15 discusses how a session is terminated.

The procedures of Sections 8, 10, 11, 12, 13, 14, and 15 deal entirely with the UA core (Section 9 describes cancellation, which applies to both UA core and proxy core). Section 16 discusses the proxy element, which facilitates routing of messages between user agents.

6 Definitions

The following terms have special significance for SIP.

Address-of-Record: An address-of-record (AOR) is a SIP or SIPS URI that points to a domain with a location service that can map the URI to another URI where the user might be available. Typically, the location service is populated through

registrations. An AOR is frequently thought of as the "public address" of the user.

Back-to-Back User Agent: A back-to-back user agent (B2BUA) is a logical entity that receives a request and processes it as a user agent server (UAS). In order to determine how the request should be answered, it acts as a user agent client (UAC) and generates requests. Unlike a proxy server, it maintains dialog state and must participate in all requests sent on the dialogs it has established. Since it is a concatenation of a UAC and UAS, no explicit definitions are needed for its behavior.

Call: A call is an informal term that refers to some communication between peers, generally set up for the purposes of a multimedia conversation.

Call Leg: Another name for a dialog [31]; no longer used in this specification.

Call Stateful: A proxy is call stateful if it retains state for a dialog from the initiating INVITE to the terminating BYE request. A call stateful proxy is always transaction stateful, but the converse is not necessarily true.

Client: A client is any network element that sends SIP requests and receives SIP responses. Clients may or may not interact directly with a human user. User agent clients and proxies are clients.

Conference: A multimedia session (see below) that contains multiple participants.

Core: Core designates the functions specific to a particular type of SIP entity, i.e., specific to either a stateful or stateless proxy, a user agent or registrar. All cores, except those for the stateless proxy, are transaction users.

Dialog: A dialog is a peer-to-peer SIP relationship between two UAs that persists for some time. A dialog is established by SIP messages, such as a 2xx response to an INVITE request. A dialog is identified by a call identifier, local tag, and a remote tag. A dialog was formerly known as a call leg in RFC 2543.

Downstream: A direction of message forwarding within a transaction that refers to the direction that requests flow from the user agent client to user agent server.

Final Response: A response that terminates a SIP transaction, as opposed to a provisional response that does not. All 2xx, 3xx, 4xx, 5xx and 6xx responses are final.

Header: A header is a component of a SIP message that conveys information about the message. It is structured as a sequence of header fields.

Header Field: A header field is a component of the SIP message header. A header field can appear as one or more header field rows. Header field rows consist of a header field name and zero or more header field values. Multiple header field values on a

given header field row are separated by commas. Some header fields can only have a single header field value, and as a result, always appear as a single header field row.

Header Field Value: A header field value is a single value; a header field consists of zero or more header field values.

Home Domain: The domain providing service to a SIP user. Typically, this is the domain present in the URI in the address-of-record of a registration.

Informational Response: Same as a provisional response.

Initiator, Calling Party, Caller: The party initiating a session (and dialog) with an INVITE request. A caller retains this role from the time it sends the initial INVITE that established a dialog until the termination of that dialog.

Invitation: An INVITE request.

Invitee, Invited User, Called Party, Callee: The party that receives an INVITE request for the purpose of establishing a new session. A callee retains this role from the time it receives the INVITE until the termination of the dialog established by that INVITE.

Location Service: A location service is used by a SIP redirect or proxy server to obtain information about a callee's possible location(s). It contains a list of bindings of address-of-record keys to zero or more contact addresses. The bindings can be created and removed in many ways; this specification defines a REGISTER method that updates the bindings.

Loop: A request that arrives at a proxy, is forwarded, and later arrives back at the same proxy. When it arrives the second time, its Request-URI is identical to the first time, and other header fields that affect proxy operation are unchanged, so that the proxy would make the same processing decision on the request it made the first time. Looped requests are errors, and the procedures for detecting them and handling them are described by the protocol.

Loose Routing: A proxy is said to be loose routing if it follows the procedures defined in this specification for processing of the Route header field. These procedures separate the destination of the request (present in the Request-URI) from

Page 1411
T-Mobile Exhibit 1002
Part 4 of 6

the set of proxies that need to be visited along the way (present in the Route header field). A proxy compliant to these mechanisms is also known as a loose router.

Message: Data sent between SIP elements as part of the protocol. SIP messages are either requests or responses.

Method: The method is the primary function that a request is meant to invoke on a server. The method is carried in the request message itself. Example methods are INVITE and BYE.

Outbound Proxy: A proxy that receives requests from a client, even though it may not be the server resolved by the Request-URI. Typically, a UA is manually configured with an outbound proxy, or can learn about one through auto-configuration protocols.

Parallel Search: In a parallel search, a proxy issues several requests to possible user locations upon receiving an incoming request. Rather than issuing one request and then waiting for the final response before issuing the next request as in a sequential search, a parallel search issues requests without waiting for the result of previous requests.

Provisional Response: A response used by the server to indicate progress, but that does not terminate a SIP transaction. 1xx responses are provisional, other responses are considered final.

Proxy, Proxy Server: An intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server primarily plays the role of routing, which means its job is to ensure that a request is sent to another entity "closer" to the targeted user. Proxies are also useful for enforcing policy (for example, making sure a user is allowed to make a call). A proxy interprets, and, if necessary, rewrites specific parts of a request message before forwarding it.

Recursion: A client recurses on a 3xx response when it generates a new request to one or more of the URIs in the Contact header field in the response.

Redirect Server: A redirect server is a user agent server that generates 3xx responses to requests it receives, directing the client to contact an alternate set of URIs.

Registrar: A registrar is a server that accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles.

Regular Transaction: A regular transaction is any transaction with

a method other than INVITE, ACK, or CANCEL.

Request: A SIP message sent from a client to a server, for the purpose of invoking a particular operation.

Response: A SIP message sent from a server to a client, for indicating the status of a request sent from the client to the server.

Ringback: Ringback is the signaling tone produced by the calling party's application indicating that a called party is being alerted (ringing).

Route Set: A route set is a collection of ordered SIP or SIPS URI which represent a list of proxies that must be traversed when sending a particular request. A route set can be learned, through headers like Record-Route, or it can be configured.

Server: A server is a network element that receives requests in order to service them and sends back responses to those requests. Examples of servers are proxies, user agent servers, redirect servers, and registrars.

Sequential Search: In a sequential search, a proxy server attempts each contact address in sequence, proceeding to the next one only after the previous has generated a final response. A 2xx or 6xx class final response always terminates a sequential search.

Session: From the SDP specification: "A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session." (RFC 2327 [1]) (A session as defined for SDP can comprise one or more RTP sessions.) As defined, a callee can be invited several times, by different calls, to the same session. If SDP is used, a session is defined by the concatenation of the SDP user name, session id, network type, address type, and address elements in the origin field.

SIP Transaction: A SIP transaction occurs between a client and a server and comprises all messages from the first request sent from the client to the server up to a final (non-1xx) response

sent from the server to the client. If the request is INVITE and the final response is a non-2xx, the transaction also includes an ACK to the response. The ACK for a 2xx response to an INVITE request is a separate transaction.

Spiral: A spiral is a SIP request that is routed to a proxy, forwarded onwards, and arrives once again at that proxy, but this time differs in a way that will result in a different processing decision than the original request. Typically, this means that the request's Request-URI differs from its previous arrival. A spiral is not an error condition, unlike a loop. A typical cause for this is call forwarding. A user calls joe@example.com. The example.com proxy forwards it to Joe's PC, which in turn, forwards it to bob@example.com. This

request is proxied back to the example.com proxy. However, this is not a loop. Since the request is targeted at a different user, it is considered a spiral, and is a valid condition.

Stateful Proxy: A logical entity that maintains the client and server transaction state machines defined by this specification during the processing of a request, also known as a transaction stateful proxy. The behavior of a stateful proxy is further defined in Section 16. A (transaction) stateful proxy is not the same as a call stateful proxy.

Stateless Proxy: A logical entity that does not maintain the client or server transaction state machines defined in this specification when it processes requests. A stateless proxy forwards every request it receives downstream and every response it receives upstream.

Strict Routing: A proxy is said to be strict routing if it follows the Route processing rules of RFC 2543 and many prior work in progress versions of this RFC. That rule caused proxies to destroy the contents of the Request-URI when a Route header field was present. Strict routing behavior is not used in this specification, in favor of a loose routing behavior. Proxies that perform strict routing are also known as strict routers.

Target Refresh Request: A target refresh request sent within a dialog is defined as a request that can modify the remote target of the dialog.

Transaction User (TU): The layer of protocol processing that resides above the transaction layer. Transaction users include the UAC core, UAS core, and proxy core.

Upstream: A direction of message forwarding within a transaction that refers to the direction that responses flow from the user agent server back to the user agent client.

URL-encoded: A character string encoded according to RFC 2396, Section 2.4 [5].

User Agent Client (UAC): A user agent client is a logical entity that creates a new request, and then uses the client transaction state machinery to send it. The role of UAC lasts only for the duration of that transaction. In other words, if a piece of software initiates a request, it acts as a UAC for the duration of that transaction. If it receives a request later, it assumes the role of a user agent server for the processing of that transaction.

UAC Core: The set of processing functions required of a UAC that reside above the transaction and transport layers.

User Agent Server (UAS): A user agent server is a logical entity that generates a response to a SIP request. The response accepts, rejects, or redirects the request. This role lasts only for the duration of that transaction. In other words, if

a piece of software responds to a request, it acts as a UAS for the duration of that transaction. If it generates a request later, it assumes the role of a user agent client for the processing of that transaction.

UAS Core: The set of processing functions required at a UAS that resides above the transaction and transport layers.

User Agent (UA): A logical entity that can act as both a user agent client and user agent server.

The role of UAC and UAS, as well as proxy and redirect servers, are defined on a transaction-by-transaction basis. For example, the user agent initiating a call acts as a UAC when sending the initial INVITE request and as a UAS when receiving a BYE request from the callee. Similarly, the same software can act as a proxy server for one request and as a redirect server for the next request.

Proxy, location, and registrar servers defined above are logical entities; implementations MAY combine them into a single application.

7 SIP Messages

SIP is a text-based protocol and uses the UTF-8 charset (RFC 2279 [7]).

A SIP message is either a request from a client to a server, or a response from a server to a client.

Both Request (section 7.1) and Response (section 7.2) messages use the basic format of RFC 2822 [3], even though the syntax differs in character set and syntax specifics. (SIP allows header fields that would not be valid RFC 2822 header fields, for example.) Both types of messages consist of a start-line, one or more header fields, an empty line indicating the end of the header fields, and an optional message-body.

```
generic-message  =  start-line
                    *message-header
                    CRLF
                    [ message-body ]
start-line       =  Request-Line / Status-Line
```

The start-line, each message-header line, and the empty line MUST be terminated by a carriage-return line-feed sequence (CRLF). Note that the empty line MUST be present even if the message-body is not.

Except for the above difference in character sets, much of SIP's message and header field syntax is identical to HTTP/1.1. Rather than repeating the syntax and semantics here, we use [HX.Y] to refer to Section X.Y of the current HTTP/1.1 specification (RFC 2616 [8]).

However, SIP is not an extension of HTTP.

7.1 Requests

SIP requests are distinguished by having a Request-Line for a start-line. A Request-Line contains a method name, a Request-URI, and the

protocol version separated by a single space (SP) character.

The Request-Line ends with CRLF. No CR or LF are allowed except in the end-of-line CRLF sequence. No linear whitespace (LWS) is allowed in any of the elements.

```
Request-Line  =  Method SP Request-URI SP SIP-Version CRLF
```

Method: This specification defines six methods: REGISTER for registering contact information, INVITE, ACK, and CANCEL for setting up sessions, BYE for terminating sessions, and OPTIONS for querying servers about their capabilities. SIP extensions, documented in standards track RFCs, may define additional methods.

Request-URI: The Request-URI is a SIP or SIPS URI as described in Section 19.1 or a general URI (RFC 2396 [5]). It indicates the user or service to which this request is being addressed. The Request-URI MUST NOT contain unescaped spaces or control characters and MUST NOT be enclosed in "<>".

SIP elements MAY support Request-URIs with schemes other than "sip" and "sips", for example the "tel" URI scheme of RFC 2806 [9]. SIP elements MAY translate non-SIP URIs using any mechanism at their disposal, resulting in SIP URI, SIPS URI, or some other scheme.

SIP-Version: Both request and response messages include the version of SIP in use, and follow [H3.1] (with HTTP replaced by SIP, and HTTP/1.1 replaced by SIP/2.0) regarding version ordering, compliance requirements, and upgrading of version numbers. To be compliant with this specification, applications sending SIP messages MUST include a SIP-Version of "SIP/2.0". The SIP-Version string is case-insensitive, but implementations MUST send upper-case.

Unlike HTTP/1.1, SIP treats the version number as a literal string. In practice, this should make no difference.

7.2 Responses

SIP responses are distinguished from requests by having a Status-Line as their start-line. A Status-Line consists of the protocol version followed by a numeric Status-Code and its associated textual phrase, with each element separated by a single SP character.

No CR or LF is allowed except in the final CRLF sequence.

```
Status-Line  =  SIP-Version SP Status-Code SP Reason-Phrase CRLF
```

The Status-Code is a 3-digit integer result code that indicates the outcome of an attempt to understand and satisfy a request. The Reason-Phrase is intended to give a short textual description of the Status-Code. The Status-Code is intended for use by automata, whereas the Reason-Phrase is intended for the human user. A client is not required to examine or display the Reason-Phrase.

While this specification suggests specific wording for the reason phrase, implementations MAY choose other text, for example, in the language indicated in the Accept-Language header field of the request.

The first digit of the Status-Code defines the class of response. The last two digits do not have any categorization role. For this reason, any response with a status code between 100 and 199 is referred to as a "1xx response", any response with a status code between 200 and 299 as a "2xx response", and so on. SIP/2.0 allows six values for the first digit:

1xx: Provisional -- request received, continuing to process the request;

2xx: Success -- the action was successfully received, understood, and accepted;

3xx: Redirection -- further action needs to be taken in order to complete the request;

4xx: Client Error -- the request contains bad syntax or cannot be fulfilled at this server;

5xx: Server Error -- the server failed to fulfill an apparently valid request;

6xx: Global Failure -- the request cannot be fulfilled at any server.

Section 21 defines these classes and describes the individual codes.

7.3 Header Fields

SIP header fields are similar to HTTP header fields in both syntax and semantics. In particular, SIP header fields follow the [H4.2] definitions of syntax for the message-header and the rules for extending header fields over multiple lines. However, the latter is specified in HTTP with implicit whitespace and folding. This specification conforms to RFC 2234 [10] and uses only explicit whitespace and folding as an integral part of the grammar.

[H4.2] also specifies that multiple header fields of the same field name whose value is a comma-separated list can be combined into one header field. That applies to SIP as well, but the specific rule is different because of the different grammars. Specifically, any SIP header whose grammar is of the form

```
header  =  "header-name" HCOLON header-value *(COMMA header-value)
```

allows for combining header fields of the same name into a comma-separated list. The Contact header field allows a comma-separated list unless the header field value is "*".

T-Mobile Exhibit 1002
Part 4 of 6

7.3.1 Header Field Format

Header fields follow the same generic header format as that given in Section 2.2 of RFC 2822 [3]. Each header field consists of a field name followed by a colon (":") and the field value.

```
field-name: field-value
```

The formal grammar for a message-header specified in Section 25 allows for an arbitrary amount of whitespace on either side of the colon; however, implementations should avoid spaces between the field name and the colon and use a single space (SP) between the colon and the field-value.

```
Subject:            lunch
Subject      :      lunch
Subject            :lunch
Subject: lunch
```

Thus, the above are all valid and equivalent, but the last is the preferred form.

Header fields can be extended over multiple lines by preceding each extra line with at least one SP or horizontal tab (HT). The line break and the whitespace at the beginning of the next line are treated as a single SP character. Thus, the following are equivalent:

```
Subject: I know you're there, pick up the phone and talk to me!
Subject: I know you're there,
         pick up the phone
         and talk to me!
```

The relative order of header fields with different field names is not significant. However, it is RECOMMENDED that header fields which are needed for proxy processing (Via, Route, Record-Route, Proxy-Require, Max-Forwards, and Proxy-Authorization, for example) appear towards the top of the message to facilitate rapid parsing. The relative order of header field rows with the same field name is important. Multiple header field rows with the same field-name MAY be present in a message if and only if the entire field-value for that header field is defined as a comma-separated list (that is, if follows the grammar defined in Section 7.3). It MUST be possible to combine the multiple header field rows into one "field-name: field-value" pair, without changing the semantics of the message, by appending each subsequent field-value to the first, each separated by a comma. The exceptions to this rule are the WWW-Authenticate, Authorization, Proxy-Authenticate, and Proxy-Authorization header fields. Multiple header

field rows with these names MAY be present in a message, but since their grammar does not follow the general form listed in Section 7.3, they MUST NOT be combined into a single header field row.

Implementations MUST be able to process multiple header field rows with the same name in any combination of the single-value-per-line or comma-separated value forms.

The following groups of header field rows are valid and equivalent:

```
Route: <sip:alice@atlanta.com>
Subject: Lunch
Route: <sip:bob@biloxi.com>
Route: <sip:carol@chicago.com>
```

```
Route: <sip:alice@atlanta.com>, <sip:bob@biloxi.com>
Route: <sip:carol@chicago.com>
Subject: Lunch
```

```
Subject: Lunch
Route: <sip:alice@atlanta.com>, <sip:bob@biloxi.com>,
       <sip:carol@chicago.com>
```

Each of the following blocks is valid but not equivalent to the others:

```
Route: <sip:alice@atlanta.com>
Route: <sip:bob@biloxi.com>
Route: <sip:carol@chicago.com>
```

```
Route: <sip:bob@biloxi.com>
Route: <sip:alice@atlanta.com>
Route: <sip:carol@chicago.com>
```

```
Route: <sip:alice@atlanta.com>,<sip:carol@chicago.com>,
       <sip:bob@biloxi.com>
```

The format of a header field-value is defined per header-name. It will always be either an opaque sequence of TEXT-UTF8 octets, or a combination of whitespace, tokens, separators, and quoted strings. Many existing header fields will adhere to the general form of a value followed by a semi-colon separated sequence of parameter-name, parameter-value pairs:

```
field-name: field-value *(;parameter-name=parameter-value)
```


Even though an arbitrary number of parameter pairs may be attached to a header field value, any given parameter-name MUST NOT appear more than once.

When comparing header fields, field names are always case-insensitive. Unless otherwise stated in the definition of a particular header field, field values, parameter names, and parameter values are case-insensitive. Tokens are always case-insensitive. Unless specified otherwise, values expressed as quoted strings are case-sensitive. For example,

```
Contact: <sip:alice@atlanta.com>;expires=3600
```

is equivalent to

```
CONTACT: <sip:alice@atlanta.com>;ExPiReS=3600
```

and

```
Content-Disposition: session;handling=optional
```

is equivalent to

```
content-disposition: Session;HANDLING=OPTIONAL
```

The following two header fields are not equivalent:

```
Warning: 370 devnull "Choose a bigger pipe"
Warning: 370 devnull "CHOOSE A BIGGER PIPE"
```

7.3.2 Header Field Classification

Some header fields only make sense in requests or responses. These are called request header fields and response header fields, respectively. If a header field appears in a message not matching its category (such as a request header field in a response), it MUST be ignored. Section 20 defines the classification of each header field.

7.3.3 Compact Form

SIP provides a mechanism to represent common header field names in an abbreviated form. This may be useful when messages would otherwise become too large to be carried on the transport available to it (exceeding the maximum transmission unit (MTU) when using UDP, for example). These compact forms are defined in Section 20. A compact form MAY be substituted for the longer form of a header field name at any time without changing the semantics of the message. A header

field name MAY appear in both long and short forms within the same message. Implementations MUST accept both the long and short forms of each header name.

7.4 Bodies

Requests, including new requests defined in extensions to this specification, MAY contain message bodies unless otherwise noted. The interpretation of the body depends on the request method.

For response messages, the request method and the response status code determine the type and interpretation of any message body. All responses MAY include a body.

7.4.1 Message Body Type

The Internet media type of the message body MUST be given by the Content-Type header field. If the body has undergone any encoding such as compression, then this MUST be indicated by the Content-Encoding header field; otherwise, Content-Encoding MUST be omitted. If applicable, the character set of the message body is indicated as

part of the Content-Type header-field value.

The "multipart" MIME type defined in RFC 2046 [11] MAY be used within the body of the message. Implementations that send requests containing multipart message bodies MUST send a session description as a non-multipart message body if the remote implementation requests this through an Accept header field that does not contain multipart.

SIP messages MAY contain binary bodies or body parts. When no explicit charset parameter is provided by the sender, media subtypes of the "text" type are defined to have a default charset value of "UTF-8".

7.4.2 Message Body Length

The body length in bytes is provided by the Content-Length header field. Section 20.14 describes the necessary contents of this header field in detail.

The "chunked" transfer encoding of HTTP/1.1 MUST NOT be used for SIP. (Note: The chunked encoding modifies the body of a message in order to transfer it as a series of chunks, each with its own size indicator.)

7.5 Framing SIP Messages

Unlike HTTP, SIP implementations can use UDP or other unreliable datagram protocols. Each such datagram carries one request or response. See Section 18 on constraints on usage of unreliable transports.

Implementations processing SIP messages over stream-oriented transports MUST ignore any CRLF appearing before the start-line [H4.1].

> The Content-Length header field value is used to locate the end of each SIP message in a stream. It will always be present when SIP messages are sent over stream-oriented transports.

8 General User Agent Behavior

A user agent represents an end system. It contains a user agent client (UAC), which generates requests, and a user agent server (UAS), which responds to them. A UAC is capable of generating a request based on some external stimulus (the user clicking a button, or a signal on a PSTN line) and processing a response. A UAS is capable of receiving a request and generating a response based on user input, external stimulus, the result of a program execution, or some other mechanism.

When a UAC sends a request, the request passes through some number of proxy servers, which forward the request towards the UAS. When the UAS generates a response, the response is forwarded towards the UAC.

UAC and UAS procedures depend strongly on two factors. First, based on whether the request or response is inside or outside of a dialog, and second, based on the method of a request. Dialogs are discussed thoroughly in Section 12; they represent a peer-to-peer relationship between user agents and are established by specific SIP methods, such as INVITE.

In this section, we discuss the method-independent rules for UAC and UAS behavior when processing requests that are outside of a dialog. This includes, of course, the requests which themselves establish a dialog.

Security procedures for requests and responses outside of a dialog are described in Section 26. Specifically, mechanisms exist for the UAS and UAC to mutually authenticate. A limited set of privacy features are also supported through encryption of bodies using S/MIME.

8.1 UAC Behavior

This section covers UAC behavior outside of a dialog.

8.1.1 Generating the Request

A valid SIP request formulated by a UAC MUST, at a minimum, contain the following header fields: To, From, CSeq, Call-ID, Max-Forwards, and Via; all of these header fields are mandatory in all SIP requests. These six header fields are the fundamental building blocks of a SIP message, as they jointly provide for most of the critical message routing services including the addressing of messages, the routing of responses, limiting message propagation, ordering of messages, and the unique identification of transactions. These header fields are in addition to the mandatory request line, which contains the method, Request-URI, and SIP version.

Examples of requests sent outside of a dialog include an INVITE to establish a session (Section 13) and an OPTIONS to query for capabilities (Section 11).

8.1.1.1 Request-URI

The initial Request-URI of the message SHOULD be set to the value of the URI in the To field. One notable exception is the REGISTER method; behavior for setting the Request-URI of REGISTER is given in Section 10. It may also be undesirable for privacy reasons or convenience to set these fields to the same value (especially if the originating UA expects that the Request-URI will be changed during transit).

In some special circumstances, the presence of a pre-existing route set can affect the Request-URI of the message. A pre-existing route set is an ordered set of URIs that identify a chain of servers, to which a UAC will send outgoing requests that are outside of a dialog. Commonly, they are configured on the UA by a user or service provider manually, or through some other non-SIP mechanism. When a provider wishes to configure a UA with an outbound proxy, it is RECOMMENDED that this be done by providing it with a pre-existing route set with

a single URI, that of the outbound proxy.

When a pre-existing route set is present, the procedures for populating the Request-URI and Route header field detailed in Section 12.2.1.1 MUST be followed (even though there is no dialog), using the desired Request-URI as the remote target URI.

8.1.1.2 To

The To header field first and foremost specifies the desired "logical" recipient of the request, or the address-of-record of the user or resource that is the target of this request. This may or may not be the ultimate recipient of the request. The To header field MAY contain a SIP or SIPS URI, but it may also make use of other URI schemes (the tel URL (RFC 2806 [9]), for example) when appropriate. All SIP implementations MUST support the SIP URI scheme. Any implementation that supports TLS MUST support the SIPS URI scheme. The To header field allows for a display name.

A UAC may learn how to populate the To header field for a particular request in a number of ways. Usually the user will suggest the To header field through a human interface, perhaps inputting the URI manually or selecting it from some sort of address book. Frequently, the user will not enter a complete URI, but rather a string of digits or letters (for example, "bob"). It is at the discretion of the UA to choose how to interpret this input. Using the string to form the user part of a SIP URI implies that the UA wishes the name to be resolved in the domain to the right-hand side (RHS) of the at-sign in the SIP URI (for instance, sip:bob@example.com). Using the string to form the user part of a SIPS URI implies that the UA wishes to communicate securely, and that the name is to be resolved in the domain to the RHS of the at-sign. The RHS will frequently be the home domain of the requestor, which allows for the home domain to process the outgoing request. This is useful for features like "speed dial" that require interpretation of the user part in the home domain. The tel URL may be used when the UA does not wish to specify the domain that should interpret a telephone number that has been input by the user. Rather, each domain through which the request passes would be given that opportunity. As an example, a user in an airport might log in and send requests through an outbound proxy in the airport. If they enter "411" (this is the phone number for local directory assistance in the United States), that needs to be interpreted and processed by the outbound proxy in the airport, not the user's home domain. In this case, tel:411 would be the right choice.

A request outside of a dialog MUST NOT contain a To tag; the tag in the To field of a request identifies the peer of the dialog. Since no dialog is established, no tag is present.

For further information on the To header field, see Section 20.39. The following is an example of a valid To header field:

```
To: Carol <sip:carol@chicago.com>
```

8.1.1.3 From

The From header field indicates the logical identity of the initiator of the request, possibly the user's address-of-record. Like the To header field, it contains a URI and optionally a display name. It is used by SIP elements to determine which processing rules to apply to a request (for example, automatic call rejection). As such, it is very important that the From URI not contain IP addresses or the FQDN of the host on which the UA is running, since these are not logical names.

The From header field allows for a display name. A UAC SHOULD use the display name "Anonymous", along with a syntactically correct, but otherwise meaningless URI (like sip:thisis@anonymous.invalid), if the identity of the client is to remain hidden.

Usually, the value that populates the From header field in requests generated by a particular UA is pre-provisioned by the user or by the administrators of the user's local domain. If a particular UA is used by multiple users, it might have switchable profiles that include a URI corresponding to the identity of the profiled user. Recipients of requests can authenticate the originator of a request in order to ascertain that they are who their From header field claims they are (see Section 22 for more on authentication).

The From field MUST contain a new "tag" parameter, chosen by the UAC. See Section 19.3 for details on choosing a tag.

For further information on the From header field, see Section 20.20. Examples:

```
From: "Bob" <sips:bob@biloxi.com> ;tag=a48s
From: sip:+12125551212@phone2net.com;tag=887s
From: Anonymous <sip:c8oqz84zk7z@privacy.org>;tag=hyh8
```

8.1.1.4 Call-ID

The Call-ID header field acts as a unique identifier to group together a series of messages. It MUST be the same for all requests and responses sent by either UA in a dialog. It SHOULD be the same in each registration from a UA.

In a new request created by a UAC outside of any dialog, the Call-ID header field MUST be selected by the UAC as a globally unique identifier over space and time unless overridden by method-specific behavior. All SIP UAs must have a means to guarantee that the Call-ID header fields they produce will not be inadvertently generated by any other UA. Note that when requests are retried after certain

failure responses that solicit an amendment to a request (for

example, a challenge for authentication), these retried requests are not considered new requests, and therefore do not need new Call-ID header fields; see Section 8.1.3.5.

Use of cryptographically random identifiers (RFC 1750 [12]) in the generation of Call-IDs is RECOMMENDED. Implementations MAY use the form "localid@host". Call-IDs are case-sensitive and are simply compared byte-by-byte.

> Using cryptographically random identifiers provides some protection against session hijacking and reduces the likelihood of unintentional Call-ID collisions.

No provisioning or human interface is required for the selection of the Call-ID header field value for a request.

For further information on the Call-ID header field, see Section 20.8.

Example:

> Call-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6@foo.bar.com

8.1.1.5 CSeq

The CSeq header field serves as a way to identify and order transactions. It consists of a sequence number and a method. The method MUST match that of the request. For non-REGISTER requests outside of a dialog, the sequence number value is arbitrary. The sequence number value MUST be expressible as a 32-bit unsigned integer and MUST be less than 2**31. As long as it follows the above guidelines, a client may use any mechanism it would like to select CSeq header field values.

Section 12.2.1.1 discusses construction of the CSeq for requests within a dialog.

Example:

> CSeq: 4711 INVITE

8.1.1.6 Max-Forwards

The Max-Forwards header field serves to limit the number of hops a request can transit on the way to its destination. It consists of an integer that is decremented by one at each hop. If the Max-Forwards value reaches 0 before the request reaches its destination, it will be rejected with a 483(Too Many Hops) error response.

A UAC MUST insert a Max-Forwards header field into each request it originates with a value that SHOULD be 70. This number was chosen to

be sufficiently large to guarantee that a request would not be dropped in any SIP network when there were no loops, but not so large as to consume proxy resources when a loop does occur. Lower values should be used with caution and only in networks where topologies are known by the UA.

8.1.1.7 Via

The Via header field indicates the transport used for the transaction and identifies the location where the response is to be sent. A Via header field value is added only after the transport that will be used to reach the next hop has been selected (which may involve the usage of the procedures in [4]).

When the UAC creates a request, it MUST insert a Via into that request. The protocol name and protocol version in the header field MUST be SIP and 2.0, respectively. The Via header field value MUST contain a branch parameter. This parameter is used to identify the transaction created by that request. This parameter is used by both the client and the server.

The branch parameter value MUST be unique across space and time for all requests sent by the UA. The exceptions to this rule are CANCEL and ACK for non-2xx responses. As discussed below, a CANCEL request will have the same value of the branch parameter as the request it cancels. As discussed in Section 17.1.1.3, an ACK for a non-2xx response will also have the same branch ID as the INVITE whose response it acknowledges.

> The uniqueness property of the branch ID parameter, to facilitate its use as a transaction ID, was not part of RFC 2543.

The branch ID inserted by an element compliant with this specification MUST always begin with the characters "z9hG4bK". These 7 characters are used as a magic cookie (7 is deemed sufficient to ensure that an older RFC 2543 implementation would not pick such a value), so that servers receiving the request can determine that the branch ID was constructed in the fashion described by this

specification (that is, globally unique). Beyond this requirement, the precise format of the branch token is implementation-defined.

The Via header maddr, ttl, and sent-by components will be set when the request is processed by the transport layer (Section 18).

Via processing for proxies is described in Section 16.6 Item 8 and Section 16.7 Item 3.

8.1.1.8 Contact

The Contact header field provides a SIP or SIPS URI that can be used to contact that specific instance of the UA for subsequent requests. The Contact header field MUST be present and contain exactly one SIP or SIPS URI in any request that can result in the establishment of a dialog. For the methods defined in this specification, that includes only the INVITE request. For these requests, the scope of the Contact is global. That is, the Contact header field value contains the URI at which the UA would like to receive requests, and this URI

MUST be valid even if used in subsequent requests outside of any dialogs.

If the Request-URI or top Route header field value contains a SIPS URI, the Contact header field MUST contain a SIPS URI as well.

For further information on the Contact header field, see Section 20.10.

8.1.1.9 Supported and Require

If the UAC supports extensions to SIP that can be applied by the server to the response, the UAC SHOULD include a Supported header field in the request listing the option tags (Section 19.2) for those extensions.

The option tags listed MUST only refer to extensions defined in standards-track RFCs. This is to prevent servers from insisting that clients implement non-standard, vendor-defined features in order to receive service. Extensions defined by experimental and informational RFCs are explicitly excluded from usage with the Supported header field in a request, since they too are often used to document vendor-defined extensions.

If the UAC wishes to insist that a UAS understand an extension that the UAC will apply to the request in order to process the request, it MUST insert a Require header field into the request listing the option tag for that extension. If the UAC wishes to apply an extension to the request and insist that any proxies that are

traversed understand that extension, it MUST insert a Proxy-Require header field into the request listing the option tag for that extension.

As with the Supported header field, the option tags in the Require and Proxy-Require header fields MUST only refer to extensions defined in standards-track RFCs.

8.1.1.10 Additional Message Components

After a new request has been created, and the header fields described above have been properly constructed, any additional optional header fields are added, as are any header fields specific to the method.

SIP requests MAY contain a MIME-encoded message-body. Regardless of the type of body that a request contains, certain header fields must be formulated to characterize the contents of the body. For further information on these header fields, see Sections 20.11 through 20.15.

8.1.2 Sending the Request

The destination for the request is then computed. Unless there is local policy specifying otherwise, the destination MUST be determined by applying the DNS procedures described in [4] as follows. If the first element in the route set indicated a strict router (resulting in forming the request as described in Section 12.2.1.1), the procedures MUST be applied to the Request-URI of the request. Otherwise, the procedures are applied to the first Route header field

value in the request (if one exists), or to the request's Request-URI if there is no Route header field present. These procedures yield an ordered set of address, port, and transports to attempt. Independent of which URI is used as input to the procedures of [4], if the Request-URI specifies a SIPS resource, the UAC MUST follow the procedures of [4] as if the input URI were a SIPS URI.

Local policy MAY specify an alternate set of destinations to attempt. If the Request-URI contains a SIPS URI, any alternate destinations MUST be contacted with TLS. Beyond that, there are no restrictions on the alternate destinations if the request contains no Route header field. This provides a simple alternative to a pre-existing route set as a way to specify an outbound proxy. However, that approach for configuring an outbound proxy is NOT RECOMMENDED; a pre-existing route set with a single URI SHOULD be used instead. If the request contains a Route header field, the request SHOULD be sent to the locations derived from its topmost value, but MAY be sent to any server that the UA is certain will honor the Route and Request-URI policies specified in this document (as opposed to those in RFC 2543). In particular, a UAC configured with an outbound proxy SHOULD

attempt to send the request to the location indicated in the first Route header field value instead of adopting the policy of sending all messages to the outbound proxy.

> This ensures that outbound proxies that do not add Record-Route header field values will drop out of the path of subsequent requests. It allows endpoints that cannot resolve the first Route URI to delegate that task to an outbound proxy.

The UAC SHOULD follow the procedures defined in [4] for stateful elements, trying each address until a server is contacted. Each try constitutes a new transaction, and therefore each carries a different topmost Via header field value with a new branch parameter. Furthermore, the transport value in the Via header field is set to whatever transport was determined for the target server.

8.1.3 Processing Responses

Responses are first processed by the transport layer and then passed up to the transaction layer. The transaction layer performs its processing and then passes the response up to the TU. The majority of response processing in the TU is method specific. However, there are some general behaviors independent of the method.

8.1.3.1 Transaction Layer Errors

In some cases, the response returned by the transaction layer will not be a SIP message, but rather a transaction layer error. When a timeout error is received from the transaction layer, it MUST be treated as if a 408 (Request Timeout) status code has been received. If a fatal transport error is reported by the transport layer (generally, due to fatal ICMP errors in UDP or connection failures in TCP), the condition MUST be treated as a 503 (Service Unavailable) status code.

8.1.3.2 Unrecognized Responses

A UAC MUST treat any final response it does not recognize as being equivalent to the x00 response code of that class, and MUST be able to process the x00 response code for all classes. For example, if a UAC receives an unrecognized response code of 431, it can safely assume that there was something wrong with its request and treat the response as if it had received a 400 (Bad Request) response code. A UAC MUST treat any provisional response different than 100 that it does not recognize as 183 (Session Progress). A UAC MUST be able to process 100 and 183 responses.

8.1.3.3 Vias

If more than one Via header field value is present in a response, the UAC SHOULD discard the message.

> The presence of additional Via header field values that precede the originator of the request suggests that the message was misrouted or possibly corrupted.

8.1.3.4 Processing 3xx Responses

Upon receipt of a redirection response (for example, a 301 response status code), clients SHOULD use the URI(s) in the Contact header field to formulate one or more new requests based on the redirected request. This process is similar to that of a proxy recursing on a 3xx class response as detailed in Sections 16.5 and 16.6. A client starts with an initial target set containing exactly one URI, the Request-URI of the original request. If a client wishes to formulate new requests based on a 3xx class response to that request, it places the URIs to try into the target set. Subject to the restrictions in this specification, a client can choose which Contact URIs it places into the target set. As with proxy recursion, a client processing 3xx class responses MUST NOT add any given URI to the target set more than once. If the original request had a SIPS URI in the Request-URI, the client MAY choose to recurse to a non-SIPS URI, but SHOULD inform the user of the redirection to an insecure URI.

> Any new request may receive 3xx responses themselves containing the original URI as a contact. Two locations can be configured to redirect to each other. Placing any given URI in the target set only once prevents infinite redirection loops.

As the target set grows, the client MAY generate new requests to the URIs in any order. A common mechanism is to order the set by the "q" parameter value from the Contact header field value. Requests to the URIs MAY be generated serially or in parallel. One approach is to process groups of decreasing q-values serially and process the URIs in each q-value group in parallel. Another is to perform only serial processing in decreasing q-value order, arbitrarily choosing between contacts of equal q-value.

If contacting an address in the list results in a failure, as defined in the next paragraph, the element moves to the next address in the list, until the list is exhausted. If the list is exhausted, then the request has failed.

Failures SHOULD be detected through failure response codes (codes greater than 399); for network errors the client transaction will report any transport layer failures to the transaction user. Note that some response codes (detailed in 8.1.3.5) indicate that the request can be retried; requests that are reattempted should not be considered failures.

When a failure for a particular contact address is received, the client SHOULD try the next contact address. This will involve creating a new client transaction to deliver a new request.

In order to create a request based on a contact address in a 3xx response, a UAC MUST copy the entire URI from the target set into the Request-URI, except for the "method-param" and "header" URI parameters (see Section 19.1.1 for a definition of these parameters). It uses the "header" parameters to create header field values for the new request, overwriting header field values associated with the redirected request in accordance with the guidelines in Section 19.1.5.

Note that in some instances, header fields that have been communicated in the contact address may instead append to existing request header fields in the original redirected request. As a general rule, if the header field can accept a comma-separated list of values, then the new header field value MAY be appended to any existing values in the original redirected request. If the header field does not accept multiple values, the value in the original redirected request MAY be overwritten by the header field value communicated in the contact address. For example, if a contact address is returned with the following value:

sip:user@host?Subject=foo&Call-Info=<http://www.foo.com>

Then any Subject header field in the original redirected request is overwritten, but the HTTP URL is merely appended to any existing Call-Info header field values.

It is RECOMMENDED that the UAC reuse the same To, From, and Call-ID used in the original redirected request, but the UAC MAY also choose to update the Call-ID header field value for new requests, for example.

Finally, once the new request has been constructed, it is sent using a new client transaction, and therefore MUST have a new branch ID in the top Via field as discussed in Section 8.1.1.7.

In all other respects, requests sent upon receipt of a redirect response SHOULD re-use the header fields and bodies of the original request.

In some instances, Contact header field values may be cached at UAC temporarily or permanently depending on the status code received and the presence of an expiration interval; see Sections 21.3.2 and 21.3.3.

8.1.3.5 Processing 4xx Responses

Certain 4xx response codes require specific UA processing, independent of the method.

If a 401 (Unauthorized) or 407 (Proxy Authentication Required) response is received, the UAC SHOULD follow the authorization procedures of Section 22.2 and Section 22.3 to retry the request with credentials.

If a 413 (Request Entity Too Large) response is received (Section 21.4.11), the request contained a body that was longer than the UAS was willing to accept. If possible, the UAC SHOULD retry the request, either omitting the body or using one of a smaller length.

If a 415 (Unsupported Media Type) response is received (Section 21.4.13), the request contained media types not supported by the UAS. The UAC SHOULD retry sending the request, this time only using content with types listed in the Accept header field in the response, with encodings listed in the Accept-Encoding header field in the response, and with languages listed in the Accept-Language in the response.

If a 416 (Unsupported URI Scheme) response is received (Section 21.4.14), the Request-URI used a URI scheme not supported by the server. The client SHOULD retry the request, this time, using a SIP URI.

If a 420 (Bad Extension) response is received (Section 21.4.15), the request contained a Require or Proxy-Require header field listing an option-tag for a feature not supported by a proxy or UAS. The UAC SHOULD retry the request, this time omitting any extensions listed in the Unsupported header field in the response.

In all of the above cases, the request is retried by creating a new request with the appropriate modifications. This new request constitutes a new transaction and SHOULD have the same value of the Call-ID, To, and From of the previous request, but the CSeq should contain a new sequence number that is one higher than the previous.

With other 4xx responses, including those yet to be defined, a retry may or may not be possible depending on the method and the use case.

8.2 UAS Behavior

When a request outside of a dialog is processed by a UAS, there is a set of processing rules that are followed, independent of the method. Section 12 gives guidance on how a UAS can tell whether a request is

inside or outside of a dialog.

Note that request processing is atomic. If a request is accepted, all state changes associated with it MUST be performed. If it is rejected, all state changes MUST NOT be performed.

UASs SHOULD process the requests in the order of the steps that follow in this section (that is, starting with authentication, then inspecting the method, the header fields, and so on throughout the remainder of this section).

8.2.1 Method Inspection

Once a request is authenticated (or authentication is skipped), the UAS MUST inspect the method of the request. If the UAS recognizes but does not support the method of a request, it MUST generate a 405 (Method Not Allowed) response. Procedures for generating responses are described in Section 8.2.6. The UAS MUST also add an Allow header field to the 405 (Method Not Allowed) response. The Allow header field MUST list the set of methods supported by the UAS generating the message. The Allow header field is presented in Section 20.5.

If the method is one supported by the server, processing continues.

8.2.2 Header Inspection

If a UAS does not understand a header field in a request (that is, the header field is not defined in this specification or in any supported extension), the server MUST ignore that header field and continue processing the message. A UAS SHOULD ignore any malformed header fields that are not necessary for processing requests.

8.2.2.1 To and Request-URI

The To header field identifies the original recipient of the request designated by the user identified in the From field. The original recipient may or may not be the UAS processing the request, due to call forwarding or other proxy operations. A UAS MAY apply any policy it wishes to determine whether to accept requests when the To

header field is not the identity of the UAS. However, it is RECOMMENDED that a UAS accept requests even if they do not recognize the URI scheme (for example, a tel: URI) in the To header field, or if the To header field does not address a known or current user of this UAS. If, on the other hand, the UAS decides to reject the request, it SHOULD generate a response with a 403 (Forbidden) status code and pass it to the server transaction for transmission.

However, the Request-URI identifies the UAS that is to process the request. If the Request-URI uses a scheme not supported by the UAS, it SHOULD reject the request with a 416 (Unsupported URI Scheme) response. If the Request-URI does not identify an address that the UAS is willing to accept requests for, it SHOULD reject the request with a 404 (Not Found) response. Typically, a UA that uses the REGISTER method to bind its address-of-record to a specific contact address will see requests whose Request-URI equals that contact address. Other potential sources of received Request-URIs include

the Contact header fields of requests and responses sent by the UA that establish or refresh dialogs.

8.2.2.2 Merged Requests

If the request has no tag in the To header field, the UAS core MUST check the request against ongoing transactions. If the From tag, Call-ID, and CSeq exactly match those associated with an ongoing transaction, but the request does not match that transaction (based on the matching rules in Section 17.2.3), the UAS core SHOULD generate a 482 (Loop Detected) response and pass it to the server transaction.

The same request has arrived at the UAS more than once, following different paths, most likely due to forking. The UAS processes the first such request received and responds with a 482 (Loop Detected) to the rest of them.

8.2.2.3 Require

Assuming the UAS decides that it is the proper element to process the request, it examines the Require header field, if present.

The Require header field is used by a UAC to tell a UAS about SIP extensions that the UAC expects the UAS to support in order to process the request properly. Its format is described in Section 20.32. If a UAS does not understand an option-tag listed in a Require header field, it MUST respond by generating a response with status code 420 (Bad Extension). The UAS MUST add an Unsupported header field, and list in it those options it does not understand amongst those in the Require header field of the request.

Note that Require and Proxy-Require MUST NOT be used in a SIP CANCEL request, or in an ACK request sent for a non-2xx response. These header fields MUST be ignored if they are present in these requests.

An ACK request for a 2xx response MUST contain only those Require and Proxy-Require values that were present in the initial request.

Example:

```
UAC->UAS:   INVITE sip:watson@bell-telephone.com SIP/2.0
            Require: 100rel

UAS->UAC:   SIP/2.0 420 Bad Extension
            Unsupported: 100rel
```

This behavior ensures that the client-server interaction will proceed without delay when all options are understood by both sides, and only slow down if options are not understood (as in the example above). For a well-matched client-server pair, the interaction proceeds quickly, saving a round-trip often required by negotiation mechanisms. In addition, it also removes ambiguity when the client requires features that the server does not understand. Some features, such as call handling fields, are only of interest to end systems.

8.2.3 Content Processing

Assuming the UAS understands any extensions required by the client, the UAS examines the body of the message, and the header fields that describe it. If there are any bodies whose type (indicated by the Content-Type), language (indicated by the Content-Language) or encoding (indicated by the Content-Encoding) are not understood, and that body part is not optional (as indicated by the Content-Disposition header field), the UAS MUST reject the request with a 415 (Unsupported Media Type) response. The response MUST contain an Accept header field listing the types of all bodies it understands, in the event the request contained bodies of types not supported by the UAS. If the request contained content encodings not understood by the UAS, the response MUST contain an Accept-Encoding header field listing the encodings understood by the UAS. If the request contained content with languages not understood by the UAS, the response MUST contain an Accept-Language header field indicating the languages understood by the UAS. Beyond these checks, body handling depends on the method and type. For further information on the processing of content-specific header fields, see Section 7.4 as well as Section 20.11 through 20.15.

8.2.4 Applying Extensions

A UAS that wishes to apply some extension when generating the response MUST NOT do so unless support for that extension is indicated in the Supported header field in the request. If the desired extension is not supported, the server SHOULD rely only on baseline SIP and any other extensions supported by the client. In rare circumstances, where the server cannot process the request without the extension, the server MAY send a 421 (Extension Required) response. This response indicates that the proper response cannot be generated without support of a specific extension. The needed extension(s) MUST be included in a Require header field in the response. This behavior is NOT RECOMMENDED, as it will generally break interoperability.

Any extensions applied to a non-421 response MUST be listed in a Require header field included in the response. Of course, the server MUST NOT apply extensions not listed in the Supported header field in the request. As a result of this, the Require header field in a response will only ever contain option tags defined in standards-track RFCs.

8.2.5 Processing the Request

Assuming all of the checks in the previous subsections are passed, the UAS processing becomes method-specific. Section 10 covers the REGISTER request, Section 11 covers the OPTIONS request, Section 13 covers the INVITE request, and Section 15 covers the BYE request.

8.2.6 Generating the Response

When a UAS wishes to construct a response to a request, it follows the general procedures detailed in the following subsections. Additional behaviors specific to the response code in question, which are not detailed in this section, may also be required.

Once all procedures associated with the creation of a response have been completed, the UAS hands the response back to the server transaction from which it received the request.

8.2.6.1 Sending a Provisional Response

One largely non-method-specific guideline for the generation of responses is that UASs SHOULD NOT issue a provisional response for a non-INVITE request. Rather, UASs SHOULD generate a final response to a non-INVITE request as soon as possible.

When a 100 (Trying) response is generated, any Timestamp header field present in the request MUST be copied into this 100 (Trying) response. If there is a delay in generating the response, the UAS SHOULD add a delay value into the Timestamp value in the response. This value MUST contain the difference between the time of sending of the response and receipt of the request, measured in seconds.

8.2.6.2 Headers and Tags

The From field of the response MUST equal the From header field of the request. The Call-ID header field of the response MUST equal the Call-ID header field of the request. The CSeq header field of the response MUST equal the CSeq field of the request. The Via header field values in the response MUST equal the Via header field values in the request and MUST maintain the same ordering.

If a request contained a To tag in the request, the To header field in the response MUST equal that of the request. However, if the To header field in the request did not contain a tag, the URI in the To header field in the response MUST equal the URI in the To header field; additionally, the UAS MUST add a tag to the To header field in the response (with the exception of the 100 (Trying) response, in which a tag MAY be present). This serves to identify the UAS that is responding, possibly resulting in a component of a dialog ID. The same tag MUST be used for all responses to that request, both final and provisional (again excepting the 100 (Trying)). Procedures for the generation of tags are defined in Section 19.3.

8.2.7 Stateless UAS Behavior

A stateless UAS is a UAS that does not maintain transaction state. It replies to requests normally, but discards any state that would ordinarily be retained by a UAS after a response has been sent. If a stateless UAS receives a retransmission of a request, it regenerates the response and resends it, just as if it were replying to the first instance of the request. A UAS cannot be stateless unless the request processing for that method would always result in the same response if the requests are identical. This rules out stateless registrars, for example. Stateless UASs do not use a transaction layer; they receive requests directly from the transport layer and send responses directly to the transport layer.

The stateless UAS role is needed primarily to handle unauthenticated requests for which a challenge response is issued. If

unauthenticated requests were handled statefully, then malicious floods of unauthenticated requests could create massive amounts of

transaction state that might slow or completely halt call processing in a UAS, effectively creating a denial of service condition; for more information see Section 26.1.5.

The most important behaviors of a stateless UAS are the following:

- A stateless UAS MUST NOT send provisional (1xx) responses.
- A stateless UAS MUST NOT retransmit responses.
- A stateless UAS MUST ignore ACK requests.
- A stateless UAS MUST ignore CANCEL requests.
- To header tags MUST be generated for responses in a stateless manner - in a manner that will generate the same tag for the same request consistently. For information on tag construction see Section 19.3.

In all other respects, a stateless UAS behaves in the same manner as a stateful UAS. A UAS can operate in either a stateful or stateless mode for each new request.

8.3 Redirect Servers

In some architectures it may be desirable to reduce the processing load on proxy servers that are responsible for routing requests, and improve signaling path robustness, by relying on redirection.

Redirection allows servers to push routing information for a request back in a response to the client, thereby taking themselves out of the loop of further messaging for this transaction while still aiding in locating the target of the request. When the originator of the request receives the redirection, it will send a new request based on the URI(s) it has received. By propagating URIs from the core of the network to its edges, redirection allows for considerable network scalability.

A redirect server is logically constituted of a server transaction layer and a transaction user that has access to a location service of some kind (see Section 10 for more on registrars and location services). This location service is effectively a database containing mappings between a single URI and a set of one or more alternative locations at which the target of that URI can be found.

A redirect server does not issue any SIP requests of its own. After receiving a request other than CANCEL, the server either refuses the request or gathers the list of alternative locations from the

location service and returns a final response of class 3xx. For well-formed CANCEL requests, it SHOULD return a 2xx response. This response ends the SIP transaction. The redirect server maintains transaction state for an entire SIP transaction. It is the responsibility of clients to detect forwarding loops between redirect servers.

When a redirect server returns a 3xx response to a request, it populates the list of (one or more) alternative locations into the Contact header field. An "expires" parameter to the Contact header field values may also be supplied to indicate the lifetime of the Contact data.

The Contact header field contains URIs giving the new locations or user names to try, or may simply specify additional transport parameters. A 301 (Moved Permanently) or 302 (Moved Temporarily) response may also give the same location and username that was targeted by the initial request but specify additional transport parameters such as a different server or multicast address to try, or a change of SIP transport from UDP to TCP or vice versa.

However, redirect servers MUST NOT redirect a request to a URI equal to the one in the Request-URI; instead, provided that the URI does not point to itself, the server MAY proxy the request to the destination URI, or MAY reject it with a 404.

> If a client is using an outbound proxy, and that proxy actually redirects requests, a potential arises for infinite redirection loops.

Note that a Contact header field value MAY also refer to a different resource than the one originally called. For example, a SIP call connected to PSTN gateway may need to deliver a special informational announcement such as "The number you have dialed has been changed."

A Contact response header field can contain any suitable URI indicating where the called party can be reached, not limited to SIP URIs. For example, it could contain URIs for phones, fax, or irc (if they were defined) or a mailto: (RFC 2368 [32]) URL. Section 26.4.4 discusses implications and limitations of redirecting a SIPS URI to a non-SIPS URI.

The "expires" parameter of a Contact header field value indicates how long the URI is valid. The value of the parameter is a number indicating seconds. If this parameter is not provided, the value of the Expires header field determines how long the URI is valid. Malformed values SHOULD be treated as equivalent to 3600.

> This provides a modest level of backwards compatibility with RFC 2543, which allowed absolute times in this header field. If an absolute time is received, it will be treated as malformed, and then default to 3600.

Redirect servers MUST ignore features that are not understood

(including unrecognized header fields, any unknown option tags in Require, or even method names) and proceed with the redirection of the request in question.

9 Canceling a Request

The previous section has discussed general UA behavior for generating requests and processing responses for requests of all methods. In this section, we discuss a general purpose method, called CANCEL.

The CANCEL request, as the name implies, is used to cancel a previous request sent by a client. Specifically, it asks the UAS to cease processing the request and to generate an error response to that request. CANCEL has no effect on a request to which a UAS has already given a final response. Because of this, it is most useful to CANCEL requests to which it can take a server long time to respond. For this reason, CANCEL is best for INVITE requests, which can take a long time to generate a response. In that usage, a UAS that receives a CANCEL request for an INVITE, but has not yet sent a final response, would "stop ringing", and then respond to the INVITE with a specific error response (a 487).

CANCEL requests can be constructed and sent by both proxies and user agent clients. Section 15 discusses under what conditions a UAC would CANCEL an INVITE request, and Section 16.10 discusses proxy usage of CANCEL.

A stateful proxy responds to a CANCEL, rather than simply forwarding a response it would receive from a downstream element. For that reason, CANCEL is referred to as a "hop-by-hop" request, since it is responded to at each stateful proxy hop.

9.1 Client Behavior

A CANCEL request SHOULD NOT be sent to cancel a request other than INVITE.

> Since requests other than INVITE are responded to immediately, sending a CANCEL for a non-INVITE request would always create a race condition.

The following procedures are used to construct a CANCEL request. The Request-URI, Call-ID, To, the numeric part of CSeq, and From header fields in the CANCEL request MUST be identical to those in the request being cancelled, including tags. A CANCEL constructed by a client MUST have only a single Via header field value matching the top Via value in the request being cancelled. Using the same values for these header fields allows the CANCEL to be matched with the request it cancels (Section 9.2 indicates how such matching occurs). However, the method part of the CSeq header field MUST have a value of CANCEL. This allows it to be identified and processed as a transaction in its own right (See Section 17).

If the request being cancelled contains a Route header field, the CANCEL request MUST include that Route header field's values.

This is needed so that stateless proxies are able to route CANCEL requests properly.

The CANCEL request MUST NOT contain any Require or Proxy-Require header fields.

Once the CANCEL is constructed, the client SHOULD check whether it has received any response (provisional or final) for the request being cancelled (herein referred to as the "original request").

If no provisional response has been received, the CANCEL request MUST NOT be sent; rather, the client MUST wait for the arrival of a provisional response before sending the request. If the original request has generated a final response, the CANCEL SHOULD NOT be sent, as it is an effective no-op, since CANCEL has no effect on requests that have already generated a final response. When the client decides to send the CANCEL, it creates a client transaction for the CANCEL and passes it the CANCEL request along with the destination address, port, and transport. The destination address, port, and transport for the CANCEL MUST be identical to those used to send the original request.

If it was allowed to send the CANCEL before receiving a response for the previous request, the server could receive the CANCEL before the original request.

Note that both the transaction corresponding to the original request and the CANCEL transaction will complete independently. However, a UAC canceling a request cannot rely on receiving a 487 (Request Terminated) response for the original request, as an RFC 2543-compliant UAS will not generate such a response. If there is no final response for the original request in 64*T1 seconds (T1 is

defined in Section 17.1.1.1), the client SHOULD then consider the original transaction cancelled and SHOULD destroy the client transaction handling the original request.

9.2 Server Behavior

The CANCEL method requests that the TU at the server side cancel a pending transaction. The TU determines the transaction to be cancelled by taking the CANCEL request, and then assuming that the request method is anything but CANCEL or ACK and applying the transaction matching procedures of Section 17.2.3. The matching transaction is the one to be cancelled.

The processing of a CANCEL request at a server depends on the type of server. A stateless proxy will forward it, a stateful proxy might respond to it and generate some CANCEL requests of its own, and a UAS will respond to it. See Section 16.10 for proxy treatment of CANCEL.

A UAS first processes the CANCEL request according to the general UAS processing described in Section 8.2. However, since CANCEL requests are hop-by-hop and cannot be resubmitted, they cannot be challenged by the server in order to get proper credentials in an Authorization header field. Note also that CANCEL requests do not contain a Require header field.

If the UAS did not find a matching transaction for the CANCEL according to the procedure above, it SHOULD respond to the CANCEL with a 481 (Call Leg/Transaction Does Not Exist). If the transaction for the original request still exists, the behavior of the UAS on receiving a CANCEL request depends on whether it has already sent a final response for the original request. If it has, the CANCEL request has no effect on the processing of the original request, no effect on any session state, and no effect on the responses generated for the original request. If the UAS has not issued a final response for the original request, its behavior depends on the method of the original request. If the original request was an INVITE, the UAS SHOULD immediately respond to the INVITE with a 487 (Request Terminated). A CANCEL request has no impact on the processing of transactions with any other method defined in this specification.

Regardless of the method of the original request, as long as the CANCEL matched an existing transaction, the UAS answers the CANCEL request itself with a 200 (OK) response. This response is constructed following the procedures described in Section 8.2.6 noting that the To tag of the response to the CANCEL and the To tag in the response to the original request SHOULD be the same. The response to CANCEL is passed to the server transaction for transmission.

10 Registrations

10.1 Overview

SIP offers a discovery capability. If a user wants to initiate a session with another user, SIP must discover the current host(s) at which the destination user is reachable. This discovery process is frequently accomplished by SIP network elements such as proxy servers and redirect servers which are responsible for receiving a request, determining where to send it based on knowledge of the location of the user, and then sending it there. To do this, SIP network elements consult an abstract service known as a location service, which provides address bindings for a particular domain. These address bindings map an incoming SIP or SIPS URI, sip:bob@biloxi.com, for example, to one or more URIs that are somehow "closer" to the desired user, sip:bob@engineering.biloxi.com, for example. Ultimately, a proxy will consult a location service that maps a received URI to the user agent(s) at which the desired recipient is currently residing.

Registration creates bindings in a location service for a particular domain that associates an address-of-record URI with one or more contact addresses. Thus, when a proxy for that domain receives a request whose Request-URI matches the address-of-record, the proxy will forward the request to the contact addresses registered to that address-of-record. Generally, it only makes sense to register an address-of-record at a domain's location service when requests for that address-of-record would be routed to that domain. In most cases, this means that the domain of the registration will need to match the domain in the URI of the address-of-record.

There are many ways by which the contents of the location service can be established. One way is administratively. In the above example,

Bob is known to be a member of the engineering department through access to a corporate database. However, SIP provides a mechanism for a UA to create a binding explicitly. This mechanism is known as registration.

Registration entails sending a REGISTER request to a special type of UAS known as a registrar. A registrar acts as the front end to the location service for a domain, reading and writing mappings based on the contents of REGISTER requests. This location service is then typically consulted by a proxy server that is responsible for routing requests for that domain.

An illustration of the overall registration process is given in Figure 2. Note that the registrar and proxy server are logical roles that can be played by a single device in a network; for purposes of

clarity the two are separated in this illustration. Also note that UAs may send requests through a proxy server in order to reach a registrar if the two are separate elements.

SIP does not mandate a particular mechanism for implementing the location service. The only requirement is that a registrar for some domain MUST be able to read and write data to the location service, and a proxy or a redirect server for that domain MUST be capable of reading that same data. A registrar MAY be co-located with a particular SIP proxy server for the same domain.

10.2 Constructing the REGISTER Request

REGISTER requests add, remove, and query bindings. A REGISTER request can add a new binding between an address-of-record and one or more contact addresses. Registration on behalf of a particular address-of-record can be performed by a suitably authorized third party. A client can also remove previous bindings or query to determine which bindings are currently in place for an address-of-record.

Except as noted, the construction of the REGISTER request and the behavior of clients sending a REGISTER request is identical to the general UAC behavior described in Section 8.1 and Section 17.1.

A REGISTER request does not establish a dialog. A UAC MAY include a Route header field in a REGISTER request based on a pre-existing route set as described in Section 8.1. The Record-Route header field has no meaning in REGISTER requests or responses, and MUST be ignored if present. In particular, the UAC MUST NOT create a new route set based on the presence or absence of a Record-Route header field in any response to a REGISTER request.

The following header fields, except Contact, MUST be included in a REGISTER request. A Contact header field MAY be included:

Request-URI: The Request-URI names the domain of the location service for which the registration is meant (for example, "sip:chicago.com"). The "userinfo" and "@" components of the SIP URI MUST NOT be present.

To: The To header field contains the address of record whose

registration is to be created, queried, or modified. The To header field and the Request-URI field typically differ, as the former contains a user name. This address-of-record MUST be a SIP URI or SIPS URI.

From: The From header field contains the address-of-record of the person responsible for the registration. The value is the same as the To header field unless the request is a third-party registration.

Call-ID: All registrations from a UAC SHOULD use the same Call-ID header field value for registrations sent to a particular registrar.

If the same client were to use different Call-ID values, a registrar could not detect whether a delayed REGISTER request might have arrived out of order.

CSeq: The CSeq value guarantees proper ordering of REGISTER requests. A UA MUST increment the CSeq value by one for each REGISTER request with the same Call-ID.

Contact: REGISTER requests MAY contain a Contact header field with zero or more values containing address bindings.

UAs MUST NOT send a new registration (that is, containing new Contact header field values, as opposed to a retransmission) until they have received a final response from the registrar for the previous one or the previous REGISTER request has timed out.

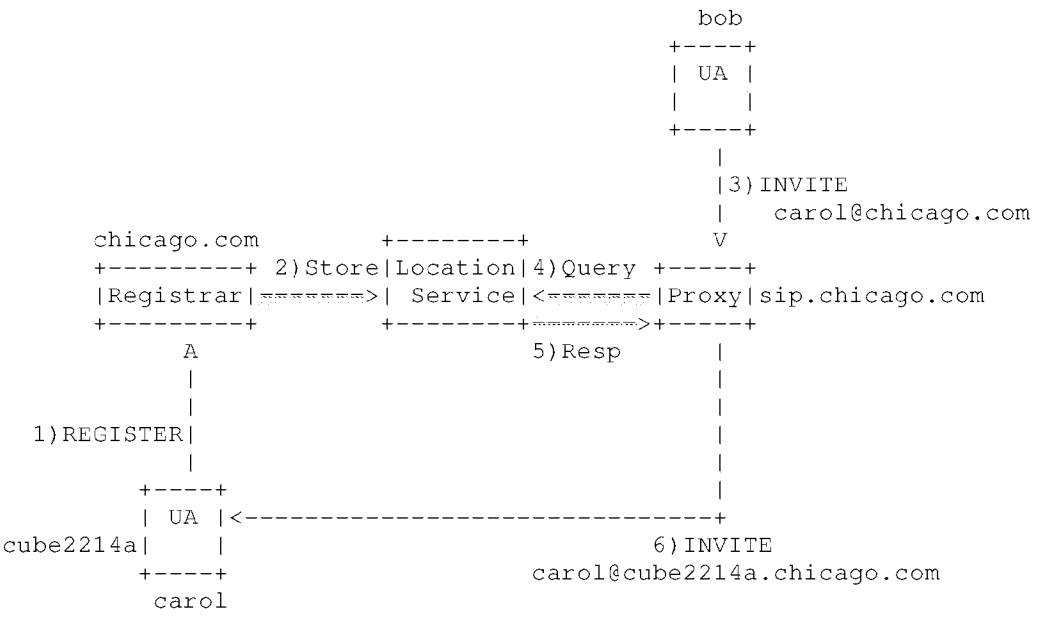


Figure 2: REGISTER example

The following Contact header parameters have a special meaning in REGISTER requests:

action: The "action" parameter from RFC 2543 has been deprecated. UACs SHOULD NOT use the "action" parameter.

expires: The "expires" parameter indicates how long the UA would like the binding to be valid. The value is a number indicating seconds. If this parameter is not provided, the value of the Expires header field is used instead. Implementations MAY treat values larger than 2**32-1 (4294967295 seconds or 136 years) as equivalent to 2**32-1. Malformed values SHOULD be treated as equivalent to 3600.

10.2.1 Adding Bindings

The REGISTER request sent to a registrar includes the contact address(es) to which SIP requests for the address-of-record should be forwarded. The address-of-record is included in the To header field of the REGISTER request.

The Contact header field values of the request typically consist of SIP or SIPS URIs that identify particular SIP endpoints (for example, "sip:carol@cube2214a.chicago.com"), but they MAY use any URI scheme. A SIP UA can choose to register telephone numbers (with the tel URL,

RFC 2806 [9]) or email addresses (with a mailto URL, RFC 2368 [32]) as Contacts for an address-of-record, for example.

For example, Carol, with address-of-record "sip:carol@chicago.com", would register with the SIP registrar of the domain chicago.com. Her registrations would then be used by a proxy server in the chicago.com domain to route requests for Carol's address-of-record to her SIP endpoint.

Once a client has established bindings at a registrar, it MAY send subsequent registrations containing new bindings or modifications to existing bindings as necessary. The 2xx response to the REGISTER request will contain, in a Contact header field, a complete list of bindings that have been registered for this address-of-record at this registrar.

If the address-of-record in the To header field of a REGISTER request is a SIPS URI, then any Contact header field values in the request SHOULD also be SIPS URIs. Clients should only register non-SIPS URIs under a SIPS address-of-record when the security of the resource represented by the contact address is guaranteed by other means. This may be applicable to URIs that invoke protocols other than SIP, or SIP devices secured by protocols other than TLS.

Registrations do not need to update all bindings. Typically, a UA only updates its own contact addresses.

10.2.1.1 Setting the Expiration Interval of Contact Addresses

When a client sends a REGISTER request, it MAY suggest an expiration interval that indicates how long the client would like the registration to be valid. (As described in Section 10.3, the registrar selects the actual time interval based on its local policy.)

There are two ways in which a client can suggest an expiration interval for a binding: through an Expires header field or an "expires" Contact header parameter. The latter allows expiration intervals to be suggested on a per-binding basis when more than one binding is given in a single REGISTER request, whereas the former suggests an expiration interval for all Contact header field values that do not contain the "expires" parameter.

If neither mechanism for expressing a suggested expiration time is present in a REGISTER, the client is indicating its desire for the server to choose.

10.2.1.2 Preferences among Contact Addresses

If more than one Contact is sent in a REGISTER request, the registering UA intends to associate all of the URIs in these Contact header field values with the address-of-record present in the To field. This list can be prioritized with the "q" parameter in the Contact header field. The "q" parameter indicates a relative preference for the particular Contact header field value compared to other bindings for this address-of-record. Section 16.6 describes

how a proxy server uses this preference indication.

10.2.2 Removing Bindings

Registrations are soft state and expire unless refreshed, but can also be explicitly removed. A client can attempt to influence the expiration interval selected by the registrar as described in Section 10.2.1. A UA requests the immediate removal of a binding by specifying an expiration interval of "0" for that contact address in a REGISTER request. UAs SHOULD support this mechanism so that bindings can be removed before their expiration interval has passed.

The REGISTER-specific Contact header field value of "*" applies to all registrations, but it MUST NOT be used unless the Expires header field is present with a value of "0".

> Use of the "*" Contact header field value allows a registering UA to remove all bindings associated with an address-of-record without knowing their precise values.

10.2.3 Fetching Bindings

A success response to any REGISTER request contains the complete list of existing bindings, regardless of whether the request contained a Contact header field. If no Contact header field is present in a REGISTER request, the list of bindings is left unchanged.

10.2.4 Refreshing Bindings

Each UA is responsible for refreshing the bindings that it has previously established. A UA SHOULD NOT refresh bindings set up by other UAs.

The 200 (OK) response from the registrar contains a list of Contact fields enumerating all current bindings. The UA compares each contact address to see if it created the contact address, using comparison rules in Section 19.1.4. If so, it updates the expiration time interval according to the expires parameter or, if absent, the Expires field value. The UA then issues a REGISTER request for each of its bindings before the expiration interval has elapsed. It MAY combine several updates into one REGISTER request.

A UA SHOULD use the same Call-ID for all registrations during a single boot cycle. Registration refreshes SHOULD be sent to the same network address as the original registration, unless redirected.

10.2.5 Setting the Internal Clock

If the response for a REGISTER request contains a Date header field, the client MAY use this header field to learn the current time in order to set any internal clocks.

10.2.6 Discovering a Registrar

UAs can use three ways to determine the address to which to send

registrations: by configuration, using the address-of-record, and multicast. A UA can be configured, in ways beyond the scope of this specification, with a registrar address. If there is no configured registrar address, the UA SHOULD use the host part of the address-of-record as the Request-URI and address the request there, using the normal SIP server location mechanisms [4]. For example, the UA for the user "sip:carol@chicago.com" addresses the REGISTER request to "sip:chicago.com".

Finally, a UA can be configured to use multicast. Multicast registrations are addressed to the well-known "all SIP servers" multicast address "sip.mcast.net" (224.0.1.75 for IPv4). No well-known IPv6 multicast address has been allocated; such an allocation will be documented separately when needed. SIP UAs MAY listen to that address and use it to become aware of the location of other local users (see [33]); however, they do not respond to the request.

Multicast registration may be inappropriate in some environments, for example, if multiple businesses share the same local area network.

10.2.7 Transmitting a Request

Once the REGISTER method has been constructed, and the destination of the message identified, UACs follow the procedures described in Section 8.1.2 to hand off the REGISTER to the transaction layer.

If the transaction layer returns a timeout error because the REGISTER yielded no response, the UAC SHOULD NOT immediately re-attempt a registration to the same registrar.

An immediate re-attempt is likely to also timeout. Waiting some reasonable time interval for the conditions causing the timeout to be corrected reduces unnecessary load on the network. No specific interval is mandated.

10.2.8 Error Responses

If a UA receives a 423 (Interval Too Brief) response, it MAY retry the registration after making the expiration interval of all contact addresses in the REGISTER request equal to or greater than the expiration interval within the Min-Expires header field of the 423 (Interval Too Brief) response.

10.3 Processing REGISTER Requests

A registrar is a UAS that responds to REGISTER requests and maintains a list of bindings that are accessible to proxy servers and redirect servers within its administrative domain. A registrar handles requests according to Section 8.2 and Section 17.2, but it accepts only REGISTER requests. A registrar MUST not generate 6xx responses.

A registrar MAY redirect REGISTER requests as appropriate. One common usage would be for a registrar listening on a multicast interface to redirect multicast REGISTER requests to its own unicast interface with a 302 (Moved Temporarily) response.

Registrars MUST ignore the Record-Route header field if it is

included in a REGISTER request. Registrars MUST NOT include a Record-Route header field in any response to a REGISTER request.

A registrar might receive a request that traversed a proxy which treats REGISTER as an unknown request and which added a Record-Route header field value.

A registrar has to know (for example, through configuration) the set of domain(s) for which it maintains bindings. REGISTER requests MUST be processed by a registrar in the order that they are received. REGISTER requests MUST also be processed atomically, meaning that a particular REGISTER request is either processed completely or not at all. Each REGISTER message MUST be processed independently of any other registration or binding changes.

When receiving a REGISTER request, a registrar follows these steps:

1. The registrar inspects the Request-URI to determine whether it has access to bindings for the domain identified in the Request-URI. If not, and if the server also acts as a proxy server, the server SHOULD forward the request to the addressed domain, following the general behavior for proxying messages described in Section 16.

2. To guarantee that the registrar supports any necessary extensions, the registrar MUST process the Require header field values as described for UASs in Section 8.2.2.

3. A registrar SHOULD authenticate the UAC. Mechanisms for the authentication of SIP user agents are described in Section 22. Registration behavior in no way overrides the generic authentication framework for SIP. If no authentication mechanism is available, the registrar MAY take the From address as the asserted identity of the originator of the request.

4. The registrar SHOULD determine if the authenticated user is authorized to modify registrations for this address-of-record. For example, a registrar might consult an authorization database that maps user names to a list of addresses-of-record for which that user has authorization to modify bindings. If the authenticated user is not authorized to modify bindings, the registrar MUST return a 403 (Forbidden) and skip the remaining steps.

   In architectures that support third-party registration, one entity may be responsible for updating the registrations associated with multiple addresses-of-record.

5. The registrar extracts the address-of-record from the To header field of the request. If the address-of-record is not valid for the domain in the Request-URI, the registrar MUST send a 404 (Not Found) response and skip the remaining steps. The URI MUST then be converted to a canonical form. To do that, all URI parameters MUST be removed (including the user-param), and any escaped characters MUST be converted to their unescaped

form. The result serves as an index into the list of bindings.

6. The registrar checks whether the request contains the Contact header field. If not, it skips to the last step. If the Contact header field is present, the registrar checks if there is one Contact field value that contains the special value "*" and an Expires field. If the request has additional Contact fields or an expiration time other than zero, the request is invalid, and the server MUST return a 400 (Invalid Request) and skip the remaining steps. If not, the registrar checks whether the Call-ID agrees with the value stored for each binding. If not, it MUST remove the binding. If it does agree, it MUST remove the binding only if the CSeq in the request is higher than the value stored for that binding. Otherwise, the update MUST be aborted and the request fails.

7. The registrar now processes each contact address in the Contact header field in turn. For each address, it determines the expiration interval as follows:

   - If the field value has an "expires" parameter, that value MUST be taken as the requested expiration.

   - If there is no such parameter, but the request has an Expires header field, that value MUST be taken as the requested expiration.

   - If there is neither, a locally-configured default value MUST be taken as the requested expiration.

   The registrar MAY choose an expiration less than the requested expiration interval. If and only if the requested expiration interval is greater than zero AND smaller than one hour AND less than a registrar-configured minimum, the registrar MAY reject the registration with a response of 423 (Interval Too Brief). This response MUST contain a Min-Expires header field that states the minimum expiration interval the registrar is willing to honor. It then skips the remaining steps.

   Allowing the registrar to set the registration interval protects it against excessively frequent registration refreshes while limiting the state that it needs to maintain and decreasing the likelihood of registrations going stale. The expiration interval of a registration is frequently used in the creation of services. An example is a follow-me service, where the user may only be available at a terminal for a brief period. Therefore, registrars should accept brief registrations; a request should only be rejected if the interval is so short that the refreshes would degrade registrar performance.

http://www.ietf.org/rfc/rfc3261.txt

For each address, the registrar then searches the list of current bindings using the URI comparison rules. If the binding does not exist, it is tentatively added. If the binding does exist, the registrar checks the Call-ID value. If the Call-ID value in the existing binding differs from the Call-ID value in the request, the binding MUST be removed if the expiration time is zero and updated otherwise. If they are the same, the registrar compares the CSeq value. If the value is higher than that of the existing binding, it MUST update or remove the binding as above. If not, the update MUST be aborted and the request fails.

This algorithm ensures that out-of-order requests from the same UA are ignored.

Each binding record records the Call-ID and CSeq values from the request.

The binding updates MUST be committed (that is, made visible to the proxy or redirect server) if and only if all binding updates and additions succeed. If any one of them fails (for example, because the back-end database commit failed), the request MUST fail with a 500 (Server Error) response and all tentative binding updates MUST be removed.

8. The registrar returns a 200 (OK) response. The response MUST contain Contact header field values enumerating all current bindings. Each Contact value MUST feature an "expires" parameter indicating its expiration interval chosen by the registrar. The response SHOULD include a Date header field.

11 Querying for Capabilities

The SIP method OPTIONS allows a UA to query another UA or a proxy server as to its capabilities. This allows a client to discover information about the supported methods, content types, extensions, codecs, etc. without "ringing" the other party. For example, before a client inserts a Require header field into an INVITE listing an option that it is not certain the destination UAS supports, the client can query the destination UAS with an OPTIONS to see if this option is returned in a Supported header field. All UAs MUST support the OPTIONS method.

The target of the OPTIONS request is identified by the Request-URI, which could identify another UA or a SIP server. If the OPTIONS is addressed to a proxy server, the Request-URI is set without a user part, similar to the way a Request-URI is set for a REGISTER request.

Alternatively, a server receiving an OPTIONS request with a Max-Forwards header field value of 0 MAY respond to the request

regardless of the Request-URI.

This behavior is common with HTTP/1.1. This behavior can be used as a "traceroute" functionality to check the capabilities of individual hop servers by sending a series of OPTIONS requests with incremented Max-Forwards values.

As is the case for general UA behavior, the transaction layer can return a timeout error if the OPTIONS yields no response. This may indicate that the target is unreachable and hence unavailable.

An OPTIONS request MAY be sent as part of an established dialog to query the peer on capabilities that may be utilized later in the dialog.

11.1 Construction of OPTIONS Request

An OPTIONS request is constructed using the standard rules for a SIP request as discussed in Section 8.1.1.

A Contact header field MAY be present in an OPTIONS.

An Accept header field SHOULD be included to indicate the type of message body the UAC wishes to receive in the response. Typically, this is set to a format that is used to describe the media capabilities of a UA, such as SDP (application/sdp).

The response to an OPTIONS request is assumed to be scoped to the Request-URI in the original request. However, only when an OPTIONS is sent as part of an established dialog is it guaranteed that future requests will be received by the server that generated the OPTIONS response.

Example OPTIONS request:

```
OPTIONS sip:carol@chicago.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKhjhs8ass877
Max-Forwards: 70
To: <sip:carol@chicago.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 63104 OPTIONS
Contact: <sip:alice@pc33.atlanta.com>
Accept: application/sdp
Content-Length: 0
```


11.2 Processing of OPTIONS Request

The response to an OPTIONS is constructed using the standard rules for a SIP response as discussed in Section 8.2.6. The response code chosen MUST be the same that would have been chosen had the request been an INVITE. That is, a 200 (OK) would be returned if the UAS is ready to accept a call, a 486 (Busy Here) would be returned if the UAS is busy, etc. This allows an OPTIONS request to be used to determine the basic state of a UAS, which can be an indication of whether the UAS will accept an INVITE request.

An OPTIONS request received within a dialog generates a 200 (OK) response that is identical to one constructed outside a dialog and does not have any impact on the dialog.

This use of OPTIONS has limitations due to the differences in proxy handling of OPTIONS and INVITE requests. While a forked INVITE can result in multiple 200 (OK) responses being returned, a forked OPTIONS will only result in a single 200 (OK) response, since it is treated by proxies using the non-INVITE handling. See Section 16.7 for the normative details.

If the response to an OPTIONS is generated by a proxy server, the proxy returns a 200 (OK), listing the capabilities of the server. The response does not contain a message body.

Allow, Accept, Accept-Encoding, Accept-Language, and Supported header fields SHOULD be present in a 200 (OK) response to an OPTIONS request. If the response is generated by a proxy, the Allow header field SHOULD be omitted as it is ambiguous since a proxy is method agnostic. Contact header fields MAY be present in a 200 (OK) response and have the same semantics as in a 3xx response. That is, they may list a set of alternative names and methods of reaching the user. A Warning header field MAY be present.

A message body MAY be sent, the type of which is determined by the Accept header field in the OPTIONS request (application/sdp is the default if the Accept header field is not present). If the types include one that can describe media capabilities, the UAS SHOULD include a body in the response for that purpose. Details on the construction of such a body in the case of application/sdp are described in [13].

Example OPTIONS response generated by a UAS (corresponding to the request in Section 11.1):

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKhjhs8ass877
 ;received=192.0.2.4
To: <sip:carol@chicago.com>;tag=93810874
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 63104 OPTIONS
Contact: <sip:carol@chicago.com>
Contact: <mailto:carol@chicago.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
Accept: application/sdp
Accept-Encoding: gzip
Accept-Language: en
Supported: foo
Content-Type: application/sdp
Content-Length: 274
```

(SDP not shown)

12 Dialogs

A key concept for a user agent is that of a dialog. A dialog represents a peer-to-peer SIP relationship between two user agents that persists for some time. The dialog facilitates sequencing of messages between the user agents and proper routing of requests between both of them. The dialog represents a context in which to interpret SIP messages. Section 8 discussed method independent UA processing for requests and responses outside of a dialog. This section discusses how those requests and responses are used to construct a dialog, and then how subsequent requests and responses are sent within a dialog.

A dialog is identified at each UA with a dialog ID, which consists of a Call-ID value, a local tag and a remote tag. The dialog ID at each UA involved in the dialog is not the same. Specifically, the local tag at one UA is identical to the remote tag at the peer UA. The tags are opaque tokens that facilitate the generation of unique dialog IDs.

A dialog ID is also associated with all responses and with any request that contains a tag in the To field. The rules for computing the dialog ID of a message depend on whether the SIP element is a UAC or UAS. For a UAC, the Call-ID value of the dialog ID is set to the Call-ID of the message, the remote tag is set to the tag in the To field of the message, and the local tag is set to the tag in the From

field of the message (these rules apply to both requests and responses). As one would expect for a UAS, the Call-ID value of the dialog ID is set to the Call-ID of the message, the remote tag is set to the tag in the From field of the message, and the local tag is set to the tag in the To field of the message.

A dialog contains certain pieces of state needed for further message transmissions within the dialog. This state consists of the dialog ID, a local sequence number (used to order requests from the UA to its peer), a remote sequence number (used to order requests from its peer to the UA), a local URI, a remote URI, remote target, a boolean flag called "secure", and a route set, which is an ordered list of URIs. The route set is the list of servers that need to be traversed to send a request to the peer. A dialog can also be in the "early" state, which occurs when it is created with a provisional response, and then transition to the "confirmed" state when a 2xx final response arrives. For other responses, or if no response arrives at all on that dialog, the early dialog terminates.

12.1 Creation of a Dialog

Dialogs are created through the generation of non-failure responses to requests with specific methods. Within this specification, only 2xx and 101-199 responses with a To tag, where the request was INVITE, will establish a dialog. A dialog established by a non-final response to a request is in the "early" state and it is called an early dialog. Extensions MAY define other means for creating dialogs. Section 13 gives more details that are specific to the INVITE method. Here, we describe the process for creation of dialog

state that is not dependent on the method.

UAs MUST assign values to the dialog ID components as described below.

12.1.1 UAS behavior

When a UAS responds to a request with a response that establishes a dialog (such as a 2xx to INVITE), the UAS MUST copy all Record-Route header field values from the request into the response (including the URIs, URI parameters, and any Record-Route header field parameters, whether they are known or unknown to the UAS) and MUST maintain the order of those values. The UAS MUST add a Contact header field to the response. The Contact header field contains an address where the UAS would like to be contacted for subsequent requests in the dialog (which includes the ACK for a 2xx response in the case of an INVITE). Generally, the host portion of this URI is the IP address or FQDN of the host. The URI provided in the Contact header field MUST be a SIP or SIPS URI. If the request that initiated the dialog contained a

SIPS URI in the Request-URI or in the top Record-Route header field value, if there was any, or the Contact header field if there was no Record-Route header field, the Contact header field in the response MUST be a SIPS URI. The URI SHOULD have global scope (that is, the same URI can be used in messages outside this dialog). The same way, the scope of the URI in the Contact header field of the INVITE is not limited to this dialog either. It can therefore be used in messages to the UAC even outside this dialog.

The UAS then constructs the state of the dialog. This state MUST be maintained for the duration of the dialog.

If the request arrived over TLS, and the Request-URI contained a SIPS URI, the "secure" flag is set to TRUE.

The route set MUST be set to the list of URIs in the Record-Route header field from the request, taken in order and preserving all URI parameters. If no Record-Route header field is present in the request, the route set MUST be set to the empty set. This route set, even if empty, overrides any pre-existing route set for future requests in this dialog. The remote target MUST be set to the URI from the Contact header field of the request.

The remote sequence number MUST be set to the value of the sequence number in the CSeq header field of the request. The local sequence number MUST be empty. The call identifier component of the dialog ID MUST be set to the value of the Call-ID in the request. The local tag component of the dialog ID MUST be set to the tag in the To field in the response to the request (which always includes a tag), and the remote tag component of the dialog ID MUST be set to the tag from the From field in the request. A UAS MUST be prepared to receive a request without a tag in the From field, in which case the tag is considered to have a value of null.

> This is to maintain backwards compatibility with RFC 2543, which did not mandate From tags.

The remote URI MUST be set to the URI in the From field, and the

local URI MUST be set to the URI in the To field.

12.1.2 UAC Behavior

When a UAC sends a request that can establish a dialog (such as an INVITE) it MUST provide a SIP or SIPS URI with global scope (i.e., the same SIP URI can be used in messages outside this dialog) in the Contact header field of the request. If the request has a Request-URI or a topmost Route header field value with a SIPS URI, the Contact header field MUST contain a SIPS URI.

When a UAC receives a response that establishes a dialog, it constructs the state of the dialog. This state MUST be maintained for the duration of the dialog.

If the request was sent over TLS, and the Request-URI contained a SIPS URI, the "secure" flag is set to TRUE.

The route set MUST be set to the list of URIs in the Record-Route header field from the response, taken in reverse order and preserving all URI parameters. If no Record-Route header field is present in the response, the route set MUST be set to the empty set. This route set, even if empty, overrides any pre-existing route set for future requests in this dialog. The remote target MUST be set to the URI from the Contact header field of the response.

The local sequence number MUST be set to the value of the sequence number in the CSeq header field of the request. The remote sequence number MUST be empty (it is established when the remote UA sends a request within the dialog). The call identifier component of the dialog ID MUST be set to the value of the Call-ID in the request. The local tag component of the dialog ID MUST be set to the tag in the From field in the request, and the remote tag component of the dialog ID MUST be set to the tag in the To field of the response. A UAC MUST be prepared to receive a response without a tag in the To field, in which case the tag is considered to have a value of null.

> This is to maintain backwards compatibility with RFC 2543, which did not mandate To tags.

The remote URI MUST be set to the URI in the To field, and the local URI MUST be set to the URI in the From field.

12.2 Requests within a Dialog

Once a dialog has been established between two UAs, either of them MAY initiate new transactions as needed within the dialog. The UA sending the request will take the UAC role for the transaction. The UA receiving the request will take the UAS role. Note that these may be different roles than the UAs held during the transaction that established the dialog.

Requests within a dialog MAY contain Record-Route and Contact header fields. However, these requests do not cause the dialog's route set to be modified, although they may modify the remote target URI. Specifically, requests that are not target refresh requests do not modify the dialog's remote target URI, and requests that are target refresh requests do. For dialogs that have been established with an

INVITE, the only target refresh request defined is re-INVITE (see Section 14). Other extensions may define different target refresh requests for dialogs established in other ways.

Note that an ACK is NOT a target refresh request.

Target refresh requests only update the dialog's remote target URI, and not the route set formed from the Record-Route. Updating the latter would introduce severe backwards compatibility problems with RFC 2543-compliant systems.

12.2.1 UAC Behavior

12.2.1.1 Generating the Request

A request within a dialog is constructed by using many of the components of the state stored as part of the dialog.

The URI in the To field of the request MUST be set to the remote URI from the dialog state. The tag in the To header field of the request MUST be set to the remote tag of the dialog ID. The From URI of the request MUST be set to the local URI from the dialog state. The tag in the From header field of the request MUST be set to the local tag of the dialog ID. If the value of the remote or local tags is null, the tag parameter MUST be omitted from the To or From header fields, respectively.

Usage of the URI from the To and From fields in the original request within subsequent requests is done for backwards compatibility with RFC 2543, which used the URI for dialog identification. In this specification, only the tags are used for dialog identification. It is expected that mandatory reflection of the original To and From URI in mid-dialog requests will be deprecated in a subsequent revision of this specification.

The Call-ID of the request MUST be set to the Call-ID of the dialog. Requests within a dialog MUST contain strictly monotonically increasing and contiguous CSeq sequence numbers (increasing-by-one) in each direction (excepting ACK and CANCEL of course, whose numbers equal the requests being acknowledged or cancelled). Therefore, if the local sequence number is not empty, the value of the local sequence number MUST be incremented by one, and this value MUST be placed into the CSeq header field. If the local sequence number is empty, an initial value MUST be chosen using the guidelines of Section 8.1.1.5. The method field in the CSeq header field value MUST match the method of the request.

With a length of 32 bits, a client could generate, within a single call, one request a second for about 136 years before needing to wrap around. The initial value of the sequence number is chosen so that subsequent requests within the same call will not wrap around. A non-zero initial value allows clients to use a time-based initial sequence number. A client could, for example, choose the 31 most significant bits of a 32-bit second clock as an initial sequence number.

The UAC uses the remote target and route set to build the Request-URI and Route header field of the request.

If the route set is empty, the UAC MUST place the remote target URI into the Request-URI. The UAC MUST NOT add a Route header field to the request.

If the route set is not empty, and the first URI in the route set contains the lr parameter (see Section 19.1.1), the UAC MUST place the remote target URI into the Request-URI and MUST include a Route header field containing the route set values in order, including all parameters.

If the route set is not empty, and its first URI does not contain the lr parameter, the UAC MUST place the first URI from the route set into the Request-URI, stripping any parameters that are not allowed in a Request-URI. The UAC MUST add a Route header field containing the remainder of the route set values in order, including all parameters. The UAC MUST then place the remote target URI into the Route header field as the last value.

For example, if the remote target is sip:user@remoteua and the route set contains:

<sip:proxy1>,<sip:proxy2>,<sip:proxy3;lr>,<sip:proxy4>

The request will be formed with the following Request-URI and Route header field:

```
METHOD sip:proxy1
Route: <sip:proxy2>,<sip:proxy3;lr>,<sip:proxy4>,<sip:user@remoteua>
```

If the first URI of the route set does not contain the lr parameter, the proxy indicated does not understand the routing mechanisms described in this document and will act as specified in RFC 2543, replacing the Request-URI with the first Route header field value it receives while forwarding the message. Placing the Request-URI at the end of the Route header field preserves the

information in that Request-URI across the strict router (it will be returned to the Request-URI when the request reaches a loose-router).

A UAC SHOULD include a Contact header field in any target refresh requests within a dialog, and unless there is a need to change it, the URI SHOULD be the same as used in previous requests within the dialog. If the "secure" flag is true, that URI MUST be a SIPS URI. As discussed in Section 12.2.2, a Contact header field in a target

refresh request updates the remote target URI. This allows a UA to provide a new contact address, should its address change during the duration of the dialog.

However, requests that are not target refresh requests do not affect the remote target URI for the dialog.

The rest of the request is formed as described in Section 8.1.1.

Once the request has been constructed, the address of the server is computed and the request is sent, using the same procedures for requests outside of a dialog (Section 8.1.2).

> The procedures in Section 8.1.2 will normally result in the request being sent to the address indicated by the topmost Route header field value or the Request-URI if no Route header field is present. Subject to certain restrictions, they allow the request to be sent to an alternate address (such as a default outbound proxy not represented in the route set).

12.2.1.2 Processing the Responses

The UAC will receive responses to the request from the transaction layer. If the client transaction returns a timeout, this is treated as a 408 (Request Timeout) response.

The behavior of a UAC that receives a 3xx response for a request sent within a dialog is the same as if the request had been sent outside a dialog. This behavior is described in Section 8.1.3.4.

> Note, however, that when the UAC tries alternative locations, it still uses the route set for the dialog to build the Route header of the request.

When a UAC receives a 2xx response to a target refresh request, it MUST replace the dialog's remote target URI with the URI from the Contact header field in that response, if present.

If the response for a request within a dialog is a 481 (Call/Transaction Does Not Exist) or a 408 (Request Timeout), the UAC SHOULD terminate the dialog. A UAC SHOULD also terminate a dialog if no response at all is received for the request (the client transaction would inform the TU about the timeout.)

> For INVITE initiated dialogs, terminating the dialog consists of sending a BYE.

12.2.2 UAS Behavior

Requests sent within a dialog, as any other requests, are atomic. If a particular request is accepted by the UAS, all the state changes associated with it are performed. If the request is rejected, none of the state changes are performed.

> Note that some requests, such as INVITEs, affect several pieces of state.

The UAS will receive the request from the transaction layer. If the request has a tag in the To header field, the UAS core computes the dialog identifier corresponding to the request and compares it with existing dialogs. If there is a match, this is a mid-dialog request. In that case, the UAS first applies the same processing rules for requests outside of a dialog, discussed in Section 8.2.

If the request has a tag in the To header field, but the dialog identifier does not match any existing dialogs, the UAS may have crashed and restarted, or it may have received a request for a different (possibly failed) UAS (the UASs can construct the To tags so that a UAS can identify that the tag was for a UAS for which it is providing recovery). Another possibility is that the incoming request has been simply misrouted. Based on the To tag, the UAS MAY either accept or reject the request. Accepting the request for acceptable To tags provides robustness, so that dialogs can persist even through crashes. UAs wishing to support this capability must take into consideration some issues such as choosing monotonically increasing CSeq sequence numbers even across reboots, reconstructing the route set, and accepting out-of-range RTP timestamps and sequence numbers.

If the UAS wishes to reject the request because it does not wish to recreate the dialog, it MUST respond to the request with a 481 (Call/Transaction Does Not Exist) status code and pass that to the server transaction.

Requests that do not change in any way the state of a dialog may be received within a dialog (for example, an OPTIONS request). They are processed as if they had been received outside the dialog.

If the remote sequence number is empty, it MUST be set to the value of the sequence number in the CSeq header field value in the request. If the remote sequence number was not empty, but the sequence number of the request is lower than the remote sequence number, the request is out of order and MUST be rejected with a 500 (Server Internal Error) response. If the remote sequence number was not empty, and the sequence number of the request is greater than the remote sequence number, the request is in order. It is possible for the CSeq sequence number to be higher than the remote sequence number by more than one. This is not an error condition, and a UAS SHOULD be prepared to receive and process requests with CSeq values more than one higher than the previous received request. The UAS MUST then set the remote sequence number to the value of the sequence number in the CSeq header field value in the request.

> If a proxy challenges a request generated by the UAC, the UAC has to resubmit the request with credentials. The resubmitted request will have a new CSeq number. The UAS will never see the first request, and thus, it will notice a gap in the CSeq number space. Such a gap does not represent any error condition.

When a UAS receives a target refresh request, it MUST replace the dialog's remote target URI with the URI from the Contact header field

in that request, if present.

12.3 Termination of a Dialog

Independent of the method, if a request outside of a dialog generates a non-2xx final response, any early dialogs created through provisional responses to that request are terminated. The mechanism for terminating confirmed dialogs is method specific. In this specification, the BYE method terminates a session and the dialog associated with it. See Section 15 for details.

13 Initiating a Session

13.1 Overview

When a user agent client desires to initiate a session (for example, audio, video, or a game), it formulates an INVITE request. The INVITE request asks a server to establish a session. This request may be forwarded by proxies, eventually arriving at one or more UAS that can potentially accept the invitation. These UASs will frequently need to query the user about whether to accept the

invitation. After some time, those UASs can accept the invitation (meaning the session is to be established) by sending a 2xx response. If the invitation is not accepted, a 3xx, 4xx, 5xx or 6xx response is sent, depending on the reason for the rejection. Before sending a final response, the UAS can also send provisional responses (1xx) to advise the UAC of progress in contacting the called user.

After possibly receiving one or more provisional responses, the UAC will get one or more 2xx responses or one non-2xx final response. Because of the protracted amount of time it can take to receive final responses to INVITE, the reliability mechanisms for INVITE transactions differ from those of other requests (like OPTIONS). Once it receives a final response, the UAC needs to send an ACK for every final response it receives. The procedure for sending this ACK depends on the type of response. For final responses between 300 and 699, the ACK processing is done in the transaction layer and follows one set of rules (See Section 17). For 2xx responses, the ACK is generated by the UAC core.

A 2xx response to an INVITE establishes a session, and it also creates a dialog between the UA that issued the INVITE and the UA that generated the 2xx response. Therefore, when multiple 2xx responses are received from different remote UAs (because the INVITE forked), each 2xx establishes a different dialog. All these dialogs are part of the same call.

This section provides details on the establishment of a session using INVITE. A UA that supports INVITE MUST also support ACK, CANCEL and BYE.

13.2 UAC Processing

13.2.1 Creating the Initial INVITE

Since the initial INVITE represents a request outside of a dialog, its construction follows the procedures of Section 8.1.1. Additional

processing is required for the specific case of INVITE.

An Allow header field (Section 20.5) SHOULD be present in the INVITE. It indicates what methods can be invoked within a dialog, on the UA sending the INVITE, for the duration of the dialog. For example, a UA capable of receiving INFO requests within a dialog [34] SHOULD include an Allow header field listing the INFO method.

A Supported header field (Section 20.37) SHOULD be present in the INVITE. It enumerates all the extensions understood by the UAC.

An Accept (Section 20.1) header field MAY be present in the INVITE. It indicates which Content-Types are acceptable to the UA, in both the response received by it, and in any subsequent requests sent to it within dialogs established by the INVITE. The Accept header field is especially useful for indicating support of various session description formats.

The UAC MAY add an Expires header field (Section 20.19) to limit the validity of the invitation. If the time indicated in the Expires header field is reached and no final answer for the INVITE has been received, the UAC core SHOULD generate a CANCEL request for the INVITE, as per Section 9.

A UAC MAY also find it useful to add, among others, Subject (Section 20.36), Organization (Section 20.25) and User-Agent (Section 20.41) header fields. They all contain information related to the INVITE.

The UAC MAY choose to add a message body to the INVITE. Section 8.1.1.10 deals with how to construct the header fields -- Content-Type among others -- needed to describe the message body.

There are special rules for message bodies that contain a session description - their corresponding Content-Disposition is "session". SIP uses an offer/answer model where one UA sends a session description, called the offer, which contains a proposed description of the session. The offer indicates the desired communications means (audio, video, games), parameters of those means (such as codec types) and addresses for receiving media from the answerer. The other UA responds with another session description, called the answer, which indicates which communications means are accepted, the parameters that apply to those means, and addresses for receiving media from the offerer. An offer/answer exchange is within the context of a dialog, so that if a SIP INVITE results in multiple dialogs, each is a separate offer/answer exchange. The offer/answer model defines restrictions on when offers and answers can be made (for example, you cannot make a new offer while one is in progress). This results in restrictions on where the offers and answers can appear in SIP messages. In this specification, offers and answers can only appear in INVITE requests and responses, and ACK. The usage of offers and answers is further restricted. For the initial INVITE transaction, the rules are:

- The initial offer MUST be in either an INVITE or, if not there, in the first reliable non-failure message from the UAS back to the UAC. In this specification, that is the final 2xx

response.

- If the initial offer is in an INVITE, the answer MUST be in a reliable non-failure message from UAS back to UAC which is correlated to that INVITE. For this specification, that is only the final 2xx response to that INVITE. That same exact answer MAY also be placed in any provisional responses sent prior to the answer. The UAC MUST treat the first session description it receives as the answer, and MUST ignore any session descriptions in subsequent responses to the initial INVITE.

- If the initial offer is in the first reliable non-failure message from the UAS back to UAC, the answer MUST be in the acknowledgement for that message (in this specification, ACK for a 2xx response).

- After having sent or received an answer to the first offer, the UAC MAY generate subsequent offers in requests based on rules specified for that method, but only if it has received answers to any previous offers, and has not sent any offers to which it hasn't gotten an answer.

- Once the UAS has sent or received an answer to the initial offer, it MUST NOT generate subsequent offers in any responses to the initial INVITE. This means that a UAS based on this specification alone can never generate subsequent offers until completion of the initial transaction.

Concretely, the above rules specify two exchanges for UAs compliant to this specification alone - the offer is in the INVITE, and the answer in the 2xx (and possibly in a 1xx as well, with the same value), or the offer is in the 2xx, and the answer is in the ACK. All user agents that support INVITE MUST support these two exchanges.

The Session Description Protocol (SDP) (RFC 2327 [1]) MUST be supported by all user agents as a means to describe sessions, and its usage for constructing offers and answers MUST follow the procedures defined in [13].

The restrictions of the offer-answer model just described only apply to bodies whose Content-Disposition header field value is "session". Therefore, it is possible that both the INVITE and the ACK contain a body message (for example, the INVITE carries a photo (Content-Disposition: render) and the ACK a session description (Content-Disposition: session)).

If the Content-Disposition header field is missing, bodies of Content-Type application/sdp imply the disposition "session", while other content types imply "render".

Once the INVITE has been created, the UAC follows the procedures defined for sending requests outside of a dialog (Section 8). This results in the construction of a client transaction that will ultimately send the request and deliver responses to the UAC.

13.2.2 Processing INVITE Responses

Once the INVITE has been passed to the INVITE client transaction, the UAC waits for responses for the INVITE. If the INVITE client transaction returns a timeout rather than a response the TU acts as if a 408 (Request Timeout) response had been received, as described in Section 8.1.3.

13.2.2.1 1xx Responses

Zero, one or multiple provisional responses may arrive before one or more final responses are received. Provisional responses for an INVITE request can create "early dialogs". If a provisional response has a tag in the To field, and if the dialog ID of the response does not match an existing dialog, one is constructed using the procedures defined in Section 12.1.2.

The early dialog will only be needed if the UAC needs to send a request to its peer within the dialog before the initial INVITE transaction completes. Header fields present in a provisional response are applicable as long as the dialog is in the early state (for example, an Allow header field in a provisional response contains the methods that can be used in the dialog while this is in the early state).

13.2.2.2 3xx Responses

A 3xx response may contain one or more Contact header field values providing new addresses where the callee might be reachable. Depending on the status code of the 3xx response (see Section 21.3), the UAC MAY choose to try those new addresses.

13.2.2.3 4xx, 5xx and 6xx Responses

A single non-2xx final response may be received for the INVITE. 4xx, 5xx and 6xx responses may contain a Contact header field value indicating the location where additional information about the error can be found. Subsequent final responses (which would only arrive under error conditions) MUST be ignored.

All early dialogs are considered terminated upon reception of the non-2xx final response.

After having received the non-2xx final response the UAC core considers the INVITE transaction completed. The INVITE client transaction handles the generation of ACKs for the response (see Section 17).

13.2.2.4 2xx Responses

Multiple 2xx responses may arrive at the UAC for a single INVITE request due to a forking proxy. Each response is distinguished by the tag parameter in the To header field, and each represents a distinct dialog, with a distinct dialog identifier.

If the dialog identifier in the 2xx response matches the dialog identifier of an existing dialog, the dialog MUST be transitioned to the "confirmed" state, and the route set for the dialog MUST be recomputed based on the 2xx response using the procedures of Section 12.2.1.2. Otherwise, a new dialog in the "confirmed" state MUST be constructed using the procedures of Section 12.1.2.

> Note that the only piece of state that is recomputed is the route set. Other pieces of state such as the highest sequence numbers (remote and local) sent within the dialog are not recomputed. The route set only is recomputed for backwards compatibility. RFC 2543 did not mandate mirroring of the Record-Route header field in a 1xx, only 2xx. However, we cannot update the entire state of the dialog, since mid-dialog requests may have been sent within the early dialog, modifying the sequence numbers, for example.

The UAC core MUST generate an ACK request for each 2xx received from the transaction layer. The header fields of the ACK are constructed in the same way as for any request sent within a dialog (see Section 12) with the exception of the CSeq and the header fields related to authentication. The sequence number of the CSeq header field MUST be the same as the INVITE being acknowledged, but the CSeq method MUST be ACK. The ACK MUST contain the same credentials as the INVITE. If the 2xx contains an offer (based on the rules above), the ACK MUST carry an answer in its body. If the offer in the 2xx response is not acceptable, the UAC core MUST generate a valid answer in the ACK and then send a BYE immediately.

Once the ACK has been constructed, the procedures of [4] are used to determine the destination address, port and transport. However, the request is passed to the transport layer directly for transmission, rather than a client transaction. This is because the UAC core handles retransmissions of the ACK, not the transaction layer. The ACK MUST be passed to the client transport every time a retransmission of the 2xx final response that triggered the ACK arrives.

The UAC core considers the INVITE transaction completed 64*T1 seconds after the reception of the first 2xx response. At this point all the early dialogs that have not transitioned to established dialogs are terminated. Once the INVITE transaction is considered completed by the UAC core, no more new 2xx responses are expected to arrive.

If, after acknowledging any 2xx response to an INVITE, the UAC does not want to continue with that dialog, then the UAC MUST terminate the dialog by sending a BYE request as described in Section 15.

13.3 UAS Processing

13.3.1 Processing of the INVITE

The UAS core will receive INVITE requests from the transaction layer. It first performs the request processing procedures of Section 8.2,

which are applied for both requests inside and outside of a dialog.

Assuming these processing states are completed without generating a response, the UAS core performs the additional processing steps:

1. If the request is an INVITE that contains an Expires header field, the UAS core sets a timer for the number of seconds indicated in the header field value. When the timer fires, the invitation is considered to be expired. If the invitation expires before the UAS has generated a final response, a 487 (Request Terminated) response SHOULD be generated.

2. If the request is a mid-dialog request, the method-independent processing described in Section 12.2.2 is first applied. It might also modify the session; Section 14 provides details.

3. If the request has a tag in the To header field but the dialog identifier does not match any of the existing dialogs, the UAS may have crashed and restarted, or may have received a request for a different (possibly failed) UAS. Section 12.2.2 provides guidelines to achieve a robust behavior under such a situation.

Processing from here forward assumes that the INVITE is outside of a dialog, and is thus for the purposes of establishing a new session.

The INVITE may contain a session description, in which case the UAS is being presented with an offer for that session. It is possible that the user is already a participant in that session, even though the INVITE is outside of a dialog. This can happen when a user is invited to the same multicast conference by multiple other participants. If desired, the UAS MAY use identifiers within the session description to detect this duplication. For example, SDP

contains a session id and version number in the origin (o) field. If the user is already a member of the session, and the session parameters contained in the session description have not changed, the UAS MAY silently accept the INVITE (that is, send a 2xx response without prompting the user).

If the INVITE does not contain a session description, the UAS is being asked to participate in a session, and the UAC has asked that the UAS provide the offer of the session. It MUST provide the offer in its first non-failure reliable message back to the UAC. In this specification, that is a 2xx response to the INVITE.

The UAS can indicate progress, accept, redirect, or reject the invitation. In all of these cases, it formulates a response using the procedures described in Section 8.2.6.

13.3.1.1 Progress

If the UAS is not able to answer the invitation immediately, it can choose to indicate some kind of progress to the UAC (for example, an indication that a phone is ringing). This is accomplished with a provisional response between 101 and 199. These provisional responses establish early dialogs and therefore follow the procedures of Section 12.1.1 in addition to those of Section 8.2.6. A UAS MAY send as many provisional responses as it likes. Each of these MUST

indicate the same dialog ID. However, these will not be delivered reliably.

If the UAS desires an extended period of time to answer the INVITE, it will need to ask for an "extension" in order to prevent proxies from canceling the transaction. A proxy has the option of canceling a transaction when there is a gap of 3 minutes between responses in a transaction. To prevent cancellation, the UAS MUST send a non-100 provisional response at every minute, to handle the possibility of lost provisional responses.

An INVITE transaction can go on for extended durations when the user is placed on hold, or when interworking with PSTN systems which allow communications to take place without answering the call. The latter is common in Interactive Voice Response (IVR) systems.

13.3.1.2 The INVITE is Redirected

If the UAS decides to redirect the call, a 3xx response is sent. A 300 (Multiple Choices), 301 (Moved Permanently) or 302 (Moved Temporarily) response SHOULD contain a Contact header field

containing one or more URIs of new addresses to be tried. The response is passed to the INVITE server transaction, which will deal with its retransmissions.

13.3.1.3 The INVITE is Rejected

A common scenario occurs when the callee is currently not willing or able to take additional calls at this end system. A 486 (Busy Here) SHOULD be returned in such a scenario. If the UAS knows that no other end system will be able to accept this call, a 600 (Busy Everywhere) response SHOULD be sent instead. However, it is unlikely that a UAS will be able to know this in general, and thus this response will not usually be used. The response is passed to the INVITE server transaction, which will deal with its retransmissions.

A UAS rejecting an offer contained in an INVITE SHOULD return a 488 (Not Acceptable Here) response. Such a response SHOULD include a Warning header field value explaining why the offer was rejected.

13.3.1.4 The INVITE is Accepted

The UAS core generates a 2xx response. This response establishes a dialog, and therefore follows the procedures of Section 12.1.1 in addition to those of Section 8.2.6.

A 2xx response to an INVITE SHOULD contain the Allow header field and the Supported header field, and MAY contain the Accept header field. Including these header fields allows the UAC to determine the features and extensions supported by the UAS for the duration of the call, without probing.

If the INVITE request contained an offer, and the UAS had not yet sent an answer, the 2xx MUST contain an answer. If the INVITE did not contain an offer, the 2xx MUST contain an offer if the UAS had

not yet sent an offer.

Once the response has been constructed, it is passed to the INVITE server transaction. Note, however, that the INVITE server transaction will be destroyed as soon as it receives this final response and passes it to the transport. Therefore, it is necessary to periodically pass the response directly to the transport until the ACK arrives. The 2xx response is passed to the transport with an interval that starts at T1 seconds and doubles for each retransmission until it reaches T2 seconds (T1 and T2 are defined in Section 17). Response retransmissions cease when an ACK request for the response is received. This is independent of whatever transport protocols are used to send the response.

> Since 2xx is retransmitted end-to-end, there may be hops between UAS and UAC that are UDP. To ensure reliable delivery across these hops, the response is retransmitted periodically even if the transport at the UAS is reliable.

If the server retransmits the 2xx response for 64*T1 seconds without receiving an ACK, the dialog is confirmed, but the session SHOULD be terminated. This is accomplished with a BYE, as described in Section 15.

14 Modifying an Existing Session

A successful INVITE request (see Section 13) establishes both a dialog between two user agents and a session using the offer-answer model. Section 12 explains how to modify an existing dialog using a target refresh request (for example, changing the remote target URI of the dialog). This section describes how to modify the actual session. This modification can involve changing addresses or ports, adding a media stream, deleting a media stream, and so on. This is accomplished by sending a new INVITE request within the same dialog that established the session. An INVITE request sent within an existing dialog is known as a re-INVITE.

> Note that a single re-INVITE can modify the dialog and the parameters of the session at the same time.

Either the caller or callee can modify an existing session.

The behavior of a UA on detection of media failure is a matter of local policy. However, automated generation of re-INVITE or BYE is NOT RECOMMENDED to avoid flooding the network with traffic when there is congestion. In any case, if these messages are sent automatically, they SHOULD be sent after some randomized interval.

> Note that the paragraph above refers to automatically generated BYEs and re-INVITEs. If the user hangs up upon media failure, the UA would send a BYE request as usual.

14.1 UAC Behavior

The same offer-answer model that applies to session descriptions in INVITEs (Section 13.2.1) applies to re-INVITEs. As a result, a UAC that wants to add a media stream, for example, will create a new

offer that contains this media stream, and send that in an INVITE request to its peer. It is important to note that the full description of the session, not just the change, is sent. This supports stateless session processing in various elements, and supports failover and recovery capabilities. Of course, a UAC MAY

send a re-INVITE with no session description, in which case the first reliable non-failure response to the re-INVITE will contain the offer (in this specification, that is a 2xx response).

If the session description format has the capability for version numbers, the offerer SHOULD indicate that the version of the session description has changed.

The To, From, Call-ID, CSeq, and Request-URI of a re-INVITE are set following the same rules as for regular requests within an existing dialog, described in Section 12.

A UAC MAY choose not to add an Alert-Info header field or a body with Content-Disposition "alert" to re-INVITEs because UASs do not typically alert the user upon reception of a re-INVITE.

Unlike an INVITE, which can fork, a re-INVITE will never fork, and therefore, only ever generate a single final response. The reason a re-INVITE will never fork is that the Request-URI identifies the target as the UA instance it established the dialog with, rather than identifying an address-of-record for the user.

Note that a UAC MUST NOT initiate a new INVITE transaction within a dialog while another INVITE transaction is in progress in either direction.

1. If there is an ongoing INVITE client transaction, the TU MUST wait until the transaction reaches the completed or terminated state before initiating the new INVITE.

2. If there is an ongoing INVITE server transaction, the TU MUST wait until the transaction reaches the confirmed or terminated state before initiating the new INVITE.

However, a UA MAY initiate a regular transaction while an INVITE transaction is in progress. A UA MAY also initiate an INVITE transaction while a regular transaction is in progress.

If a UA receives a non-2xx final response to a re-INVITE, the session parameters MUST remain unchanged, as if no re-INVITE had been issued. Note that, as stated in Section 12.2.1.2, if the non-2xx final response is a 481 (Call/Transaction Does Not Exist), or a 408 (Request Timeout), or no response at all is received for the re-INVITE (that is, a timeout is returned by the INVITE client transaction), the UAC will terminate the dialog.

If a UAC receives a 491 response to a re-INVITE, it SHOULD start a timer with a value T chosen as follows:

1. If the UAC is the owner of the Call-ID of the dialog ID (meaning it generated the value), T has a randomly chosen value between 2.1 and 4 seconds in units of 10 ms.

2. If the UAC is not the owner of the Call-ID of the dialog ID, T has a randomly chosen value of between 0 and 2 seconds in units of 10 ms.

When the timer fires, the UAC SHOULD attempt the re-INVITE once more, if it still desires for that session modification to take place. For example, if the call was already hung up with a BYE, the re-INVITE would not take place.

The rules for transmitting a re-INVITE and for generating an ACK for a 2xx response to re-INVITE are the same as for the initial INVITE (Section 13.2.1).

14.2 UAS Behavior

Section 13.3.1 describes the procedure for distinguishing incoming re-INVITEs from incoming initial INVITEs and handling a re-INVITE for an existing dialog.

A UAS that receives a second INVITE before it sends the final response to a first INVITE with a lower CSeq sequence number on the same dialog MUST return a 500 (Server Internal Error) response to the second INVITE and MUST include a Retry-After header field with a randomly chosen value of between 0 and 10 seconds.

A UAS that receives an INVITE on a dialog while an INVITE it had sent on that dialog is in progress MUST return a 491 (Request Pending) response to the received INVITE.

If a UA receives a re-INVITE for an existing dialog, it MUST check any version identifiers in the session description or, if there are no version identifiers, the content of the session description to see if it has changed. If the session description has changed, the UAS MUST adjust the session parameters accordingly, possibly after asking the user for confirmation.

> Versioning of the session description can be used to accommodate the capabilities of new arrivals to a conference, add or delete media, or change from a unicast to a multicast conference.

If the new session description is not acceptable, the UAS can reject it by returning a 488 (Not Acceptable Here) response for the re-INVITE. This response SHOULD include a Warning header field.

If a UAS generates a 2xx response and never receives an ACK, it

SHOULD generate a BYE to terminate the dialog.

A UAS MAY choose not to generate 180 (Ringing) responses for a re-INVITE because UACs do not typically render this information to the user. For the same reason, UASs MAY choose not to use an Alert-Info header field or a body with Content-Disposition "alert" in responses to a re-INVITE.

A UAS providing an offer in a 2xx (because the INVITE did not contain an offer) SHOULD construct the offer as if the UAS were making a brand new call, subject to the constraints of sending an offer that updates an existing session, as described in [13] in the case of SDP. Specifically, this means that it SHOULD include as many media formats and media types that the UA is willing to support. The UAS MUST ensure that the session description overlaps with its previous session description in media formats, transports, or other parameters that require support from the peer. This is to avoid the need for the peer to reject the session description. If, however, it is unacceptable to the UAC, the UAC SHOULD generate an answer with a valid session description, and then send a BYE to terminate the session.

15 Terminating a Session

This section describes the procedures for terminating a session established by SIP. The state of the session and the state of the dialog are very closely related. When a session is initiated with an INVITE, each 1xx or 2xx response from a distinct UAS creates a dialog, and if that response completes the offer/answer exchange, it also creates a session. As a result, each session is "associated" with a single dialog - the one which resulted in its creation. If an initial INVITE generates a non-2xx final response, that terminates all sessions (if any) and all dialogs (if any) that were created through responses to the request. By virtue of completing the transaction, a non-2xx final response also prevents further sessions from being created as a result of the INVITE. The BYE request is used to terminate a specific session or attempted session. In this case, the specific session is the one with the peer UA on the other side of the dialog. When a BYE is received on a dialog, any session associated with that dialog SHOULD terminate. A UA MUST NOT send a BYE outside of a dialog. The caller's UA MAY send a BYE for either confirmed or early dialogs, and the callee's UA MAY send a BYE on confirmed dialogs, but MUST NOT send a BYE on early dialogs.

However, the callee's UA MUST NOT send a BYE on a confirmed dialog until it has received an ACK for its 2xx response or until the server transaction times out. If no SIP extensions have defined other application layer states associated with the dialog, the BYE also terminates the dialog.

The impact of a non-2xx final response to INVITE on dialogs and sessions makes the use of CANCEL attractive. The CANCEL attempts to force a non-2xx response to the INVITE (in particular, a 487). Therefore, if a UAC wishes to give up on its call attempt entirely, it can send a CANCEL. If the INVITE results in 2xx final response(s) to the INVITE, this means that a UAS accepted the invitation while the CANCEL was in progress. The UAC MAY continue with the sessions established by any 2xx responses, or MAY terminate them with BYE.

The notion of "hanging up" is not well defined within SIP. It is specific to a particular, albeit common, user interface. Typically, when the user hangs up, it indicates a desire to terminate the attempt to establish a session, and to terminate any sessions already created. For the caller's UA, this would imply a CANCEL request if the initial INVITE has not generated a final response, and a BYE to all confirmed dialogs after a final response. For the callee's UA, it would typically imply a BYE; presumably, when the user picked up the phone, a 2xx was generated, and so hanging up would result in a BYE after the ACK is received. This does not mean a user cannot hang up before receipt of the ACK, it just means that the software in his phone needs to maintain state for a short while in order to clean up properly. If the particular UI allows for the user to reject a call before its answered, a 403 (Forbidden) is a good way to express that. As per the rules above, a BYE can't be sent.

15.1 Terminating a Session with a BYE Request

15.1.1 UAC Behavior

A BYE request is constructed as would any other request within a dialog, as described in Section 12.

Once the BYE is constructed, the UAC core creates a new non-INVITE client transaction, and passes it the BYE request. The UAC MUST consider the session terminated (and therefore stop sending or listening for media) as soon as the BYE request is passed to the client transaction. If the response for the BYE is a 481 (Call/Transaction Does Not Exist) or a 408 (Request Timeout) or no

response at all is received for the BYE (that is, a timeout is returned by the client transaction), the UAC MUST consider the session and the dialog terminated.

15.1.2 UAS Behavior

A UAS first processes the BYE request according to the general UAS processing described in Section 8.2. A UAS core receiving a BYE request checks if it matches an existing dialog. If the BYE does not match an existing dialog, the UAS core SHOULD generate a 481 (Call/Transaction Does Not Exist) response and pass that to the server transaction.

This rule means that a BYE sent without tags by a UAC will be rejected. This is a change from RFC 2543, which allowed BYE without tags.

A UAS core receiving a BYE request for an existing dialog MUST follow the procedures of Section 12.2.2 to process the request. Once done, the UAS SHOULD terminate the session (and therefore stop sending and listening for media). The only case where it can elect not to are multicast sessions, where participation is possible even if the other participant in the dialog has terminated its involvement in the

session. Whether or not it ends its participation on the session, the UAS core MUST generate a 2xx response to the BYE, and MUST pass that to the server transaction for transmission.

The UAS MUST still respond to any pending requests received for that dialog. It is RECOMMENDED that a 487 (Request Terminated) response be generated to those pending requests.

16 Proxy Behavior

16.1 Overview

SIP proxies are elements that route SIP requests to user agent servers and SIP responses to user agent clients. A request may traverse several proxies on its way to a UAS. Each will make routing decisions, modifying the request before forwarding it to the next element. Responses will route through the same set of proxies traversed by the request in the reverse order.

Being a proxy is a logical role for a SIP element. When a request arrives, an element that can play the role of a proxy first decides if it needs to respond to the request on its own. For instance, the request may be malformed or the element may need credentials from the client before acting as a proxy. The element MAY respond with any

appropriate error code. When responding directly to a request, the element is playing the role of a UAS and MUST behave as described in Section 8.2.

A proxy can operate in either a stateful or stateless mode for each new request. When stateless, a proxy acts as a simple forwarding element. It forwards each request downstream to a single element determined by making a targeting and routing decision based on the request. It simply forwards every response it receives upstream. A stateless proxy discards information about a message once the message has been forwarded. A stateful proxy remembers information (specifically, transaction state) about each incoming request and any requests it sends as a result of processing the incoming request. It uses this information to affect the processing of future messages associated with that request. A stateful proxy MAY choose to "fork" a request, routing it to multiple destinations. Any request that is forwarded to more than one location MUST be handled statefully.

In some circumstances, a proxy MAY forward requests using stateful transports (such as TCP) without being transaction-stateful. For instance, a proxy MAY forward a request from one TCP connection to another transaction statelessly as long as it places enough information in the message to be able to forward the response down the same connection the request arrived on. Requests forwarded between different types of transports where the proxy's TU must take an active role in ensuring reliable delivery on one of the transports MUST be forwarded transaction statefully.

A stateful proxy MAY transition to stateless operation at any time during the processing of a request, so long as it did not do anything that would otherwise prevent it from being stateless initially (forking, for example, or generation of a 100 response). When

performing such a transition, all state is simply discarded. The proxy SHOULD NOT initiate a CANCEL request.

Much of the processing involved when acting statelessly or statefully for a request is identical. The next several subsections are written from the point of view of a stateful proxy. The last section calls out those places where a stateless proxy behaves differently.

16.2 Stateful Proxy

When stateful, a proxy is purely a SIP transaction processing engine. Its behavior is modeled here in terms of the server and client transactions defined in Section 17. A stateful proxy has a server transaction associated with one or more client transactions by a higher layer proxy processing component (see figure 3), known as a proxy core. An incoming request is processed by a server

transaction. Requests from the server transaction are passed to a proxy core. The proxy core determines where to route the request, choosing one or more next-hop locations. An outgoing request for each next-hop location is processed by its own associated client transaction. The proxy core collects the responses from the client transactions and uses them to send responses to the server transaction.

A stateful proxy creates a new server transaction for each new request received. Any retransmissions of the request will then be handled by that server transaction per Section 17. The proxy core MUST behave as a UAS with respect to sending an immediate provisional on that server transaction (such as 100 Trying) as described in Section 8.2.6. Thus, a stateful proxy SHOULD NOT generate 100 (Trying) responses to non-INVITE requests.

This is a model of proxy behavior, not of software. An implementation is free to take any approach that replicates the external behavior this model defines.

For all new requests, including any with unknown methods, an element intending to proxy the request MUST:

1. Validate the request (Section 16.3)

2. Preprocess routing information (Section 16.4)

3. Determine target(s) for the request (Section 16.5)

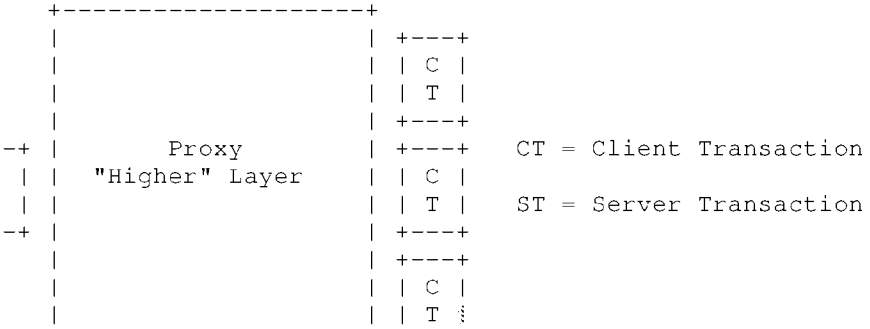


Figure 3: Stateful Proxy Model

4. Forward the request to each target (Section 16.6)

5. Process all responses (Section 16.7)

16.3 Request Validation

Before an element can proxy a request, it MUST verify the message's validity. A valid message must pass the following checks:

1. Reasonable Syntax

2. URI scheme

3. Max-Forwards

4. (Optional) Loop Detection

5. Proxy-Require

6. Proxy-Authorization

If any of these checks fail, the element MUST behave as a user agent server (see Section 8.2) and respond with an error code.

Notice that a proxy is not required to detect merged requests and MUST NOT treat merged requests as an error condition. The endpoints receiving the requests will resolve the merge as described in Section 8.2.2.2.

1. Reasonable syntax check

The request MUST be well-formed enough to be handled with a server transaction. Any components involved in the remainder of these Request Validation steps or the Request Forwarding section MUST be well-formed. Any other components, well-formed or not, SHOULD be ignored and remain unchanged when the message is forwarded. For instance, an element would not reject a request because of a malformed Date header field. Likewise, a proxy would not remove a malformed Date header field before forwarding a request.

This protocol is designed to be extended. Future extensions may define new methods and header fields at any time. An element MUST NOT refuse to proxy a request because it contains a method or header field it does not know about.

2. URI scheme check

   If the Request-URI has a URI whose scheme is not understood by the proxy, the proxy SHOULD reject the request with a 416 (Unsupported URI Scheme) response.

3. Max-Forwards check

   The Max-Forwards header field (Section 20.22) is used to limit the number of elements a SIP request can traverse.

   If the request does not contain a Max-Forwards header field, this check is passed.

   If the request contains a Max-Forwards header field with a field value greater than zero, the check is passed.

   If the request contains a Max-Forwards header field with a field value of zero (0), the element MUST NOT forward the request. If the request was for OPTIONS, the element MAY act as the final recipient and respond per Section 11. Otherwise, the element MUST return a 483 (Too many hops) response.

4. Optional Loop Detection check

   An element MAY check for forwarding loops before forwarding a request. If the request contains a Via header field with a sent-by value that equals a value placed into previous requests by the proxy, the request has been forwarded by this element before. The request has either looped or is legitimately spiraling through the element. To determine if the request has looped, the element MAY perform the branch parameter calculation described in Step 8 of Section 16.6 on this message and compare it to the parameter received in that Via header field. If the parameters match, the request has looped. If they differ, the request is spiraling, and processing continues. If a loop is detected, the element MAY return a 482 (Loop Detected) response.

5. Proxy-Require check

   Future extensions to this protocol may introduce features that require special handling by proxies. Endpoints will include a Proxy-Require header field in requests that use these features, telling the proxy not to process the request unless the feature is understood.

   If the request contains a Proxy-Require header field (Section 20.29) with one or more option-tags this element does not understand, the element MUST return a 420 (Bad Extension)

response. The response MUST include an Unsupported (Section 20.40) header field listing those option-tags the element did not understand.

6. Proxy-Authorization check

If an element requires credentials before forwarding a request, the request MUST be inspected as described in Section 22.3. That section also defines what the element must do if the inspection fails.

16.4 Route Information Preprocessing

The proxy MUST inspect the Request-URI of the request. If the Request-URI of the request contains a value this proxy previously placed into a Record-Route header field (see Section 16.6 item 4), the proxy MUST replace the Request-URI in the request with the last value from the Route header field, and remove that value from the Route header field. The proxy MUST then proceed as if it received this modified request.

This will only happen when the element sending the request to the proxy (which may have been an endpoint) is a strict router. This rewrite on receive is necessary to enable backwards compatibility with those elements. It also allows elements following this specification to preserve the Request-URI through strict-routing proxies (see Section 12.2.1.1).

This requirement does not obligate a proxy to keep state in order to detect URIs it previously placed in Record-Route header fields. Instead, a proxy need only place enough information in those URIs to recognize them as values it provided when they later appear.

If the Request-URI contains a maddr parameter, the proxy MUST check to see if its value is in the set of addresses or domains the proxy is configured to be responsible for. If the Request-URI has a maddr parameter with a value the proxy is responsible for, and the request was received using the port and transport indicated (explicitly or by default) in the Request-URI, the proxy MUST strip the maddr and any non-default port or transport parameter and continue processing as if those values had not been present in the request.

A request may arrive with a maddr matching the proxy, but on a port or transport different from that indicated in the URI. Such a request needs to be forwarded to the proxy using the indicated port and transport.

If the first value in the Route header field indicates this proxy, the proxy MUST remove that value from the request.

16.5 Determining Request Targets

Next, the proxy calculates the target(s) of the request. The set of targets will either be predetermined by the contents of the request

or will be obtained from an abstract location service. Each target in the set is represented as a URI.

If the Request-URI of the request contains an maddr parameter, the Request-URI MUST be placed into the target set as the only target URI, and the proxy MUST proceed to Section 16.6.

If the domain of the Request-URI indicates a domain this element is not responsible for, the Request-URI MUST be placed into the target set as the only target, and the element MUST proceed to the task of Request Forwarding (Section 16.6).

> There are many circumstances in which a proxy might receive a request for a domain it is not responsible for. A firewall proxy handling outgoing calls (the way HTTP proxies handle outgoing requests) is an example of where this is likely to occur.

If the target set for the request has not been predetermined as described above, this implies that the element is responsible for the domain in the Request-URI, and the element MAY use whatever mechanism it desires to determine where to send the request. Any of these mechanisms can be modeled as accessing an abstract Location Service. This may consist of obtaining information from a location service created by a SIP Registrar, reading a database, consulting a presence server, utilizing other protocols, or simply performing an algorithmic substitution on the Request-URI. When accessing the location service constructed by a registrar, the Request-URI MUST first be canonicalized as described in Section 10.3 before being used as an index. The output of these mechanisms is used to construct the target set.

If the Request-URI does not provide sufficient information for the proxy to determine the target set, it SHOULD return a 485 (Ambiguous) response. This response SHOULD contain a Contact header field containing URIs of new addresses to be tried. For example, an INVITE

to sip:John.Smith@company.com may be ambiguous at a proxy whose location service has multiple John Smiths listed. See Section 21.4.23 for details.

Any information in or about the request or the current environment of the element MAY be used in the construction of the target set. For instance, different sets may be constructed depending on contents or the presence of header fields and bodies, the time of day of the request's arrival, the interface on which the request arrived, failure of previous requests, or even the element's current level of utilization.

As potential targets are located through these services, their URIs are added to the target set. Targets can only be placed in the target set once. If a target URI is already present in the set (based on the definition of equality for the URI type), it MUST NOT be added again.

A proxy MUST NOT add additional targets to the target set if the Request-URI of the original request does not indicate a resource this proxy is responsible for.

A proxy can only change the Request-URI of a request during forwarding if it is responsible for that URI. If the proxy is not responsible for that URI, it will not recurse on 3xx or 416 responses as described below.

If the Request-URI of the original request indicates a resource this proxy is responsible for, the proxy MAY continue to add targets to the set after beginning Request Forwarding. It MAY use any information obtained during that processing to determine new targets. For instance, a proxy may choose to incorporate contacts obtained in a redirect response (3xx) into the target set. If a proxy uses a dynamic source of information while building the target set (for instance, if it consults a SIP Registrar), it SHOULD monitor that source for the duration of processing the request. New locations SHOULD be added to the target set as they become available. As above, any given URI MUST NOT be added to the set more than once.

Allowing a URI to be added to the set only once reduces unnecessary network traffic, and in the case of incorporating contacts from redirect requests prevents infinite recursion.

For example, a trivial location service is a "no-op", where the target URI is equal to the incoming request URI. The request is sent to a specific next hop proxy for further processing. During request

forwarding of Section 16.6, Item 6, the identity of that next hop, expressed as a SIP or SIPS URI, is inserted as the top-most Route header field value into the request.

If the Request-URI indicates a resource at this proxy that does not exist, the proxy MUST return a 404 (Not Found) response.

If the target set remains empty after applying all of the above, the proxy MUST return an error response, which SHOULD be the 480 (Temporarily Unavailable) response.

16.6 Request Forwarding

As soon as the target set is non-empty, a proxy MAY begin forwarding the request. A stateful proxy MAY process the set in any order. It MAY process multiple targets serially, allowing each client transaction to complete before starting the next. It MAY start client transactions with every target in parallel. It also MAY arbitrarily divide the set into groups, processing the groups serially and processing the targets in each group in parallel.

A common ordering mechanism is to use the qvalue parameter of targets obtained from Contact header fields (see Section 20.10). Targets are processed from highest qvalue to lowest. Targets with equal qvalues may be processed in parallel.

A stateful proxy must have a mechanism to maintain the target set as responses are received and associate the responses to each forwarded request with the original request. For the purposes of this model, this mechanism is a "response context" created by the proxy layer

before forwarding the first request.

For each target, the proxy forwards the request following these steps:

1. Make a copy of the received request
2. Update the Request-URI
3. Update the Max-Forwards header field
4. Optionally add a Record-route header field value
5. Optionally add additional header fields
6. Postprocess routing information
7. Determine the next-hop address, port, and transport

8. Add a Via header field value
9. Add a Content-Length header field if necessary
10. Forward the new request
11. Set timer C

Each of these steps is detailed below:

1. Copy request

   The proxy starts with a copy of the received request. The copy MUST initially contain all of the header fields from the received request. Fields not detailed in the processing described below MUST NOT be removed. The copy SHOULD maintain the ordering of the header fields as in the received request. The proxy MUST NOT reorder field values with a common field name (See Section 7.3.1). The proxy MUST NOT add to, modify, or remove the message body.

   An actual implementation need not perform a copy; the primary requirement is that the processing for each next hop begin with the same request.

2. Request-URI

   The Request-URI in the copy's start line MUST be replaced with the URI for this target. If the URI contains any parameters not allowed in a Request-URI, they MUST be removed.

   This is the essence of a proxy's role. This is the mechanism through which a proxy routes a request toward its destination.

   In some circumstances, the received Request-URI is placed into the target set without being modified. For that target, the replacement above is effectively a no-op.

3. Max-Forwards

If the copy contains a Max-Forwards header field, the proxy MUST decrement its value by one (1).

If the copy does not contain a Max-Forwards header field, the proxy MUST add one with a field value, which SHOULD be 70.

Some existing UAs will not provide a Max-Forwards header field in a request.

4. Record-Route

If this proxy wishes to remain on the path of future requests in a dialog created by this request (assuming the request creates a dialog), it MUST insert a Record-Route header field value into the copy before any existing Record-Route header field values, even if a Route header field is already present.

Requests establishing a dialog may contain a preloaded Route header field.

If this request is already part of a dialog, the proxy SHOULD insert a Record-Route header field value if it wishes to remain on the path of future requests in the dialog. In normal endpoint operation as described in Section 12, these Record-Route header field values will not have any effect on the route sets used by the endpoints.

The proxy will remain on the path if it chooses to not insert a Record-Route header field value into requests that are already part of a dialog. However, it would be removed from the path when an endpoint that has failed reconstitutes the dialog.

A proxy MAY insert a Record-Route header field value into any request. If the request does not initiate a dialog, the endpoints will ignore the value. See Section 12 for details on how endpoints use the Record-Route header field values to construct Route header fields.

Each proxy in the path of a request chooses whether to add a Record-Route header field value independently - the presence of a Record-Route header field in a request does not obligate this proxy to add a value.

The URI placed in the Record-Route header field value MUST be a SIP or SIPS URI. This URI MUST contain an lr parameter (see Section 19.1.1). This URI MAY be different for each destination the request is forwarded to. The URI SHOULD NOT contain the transport parameter unless the proxy has knowledge (such as in a private network) that the next downstream element that will be in the path of subsequent requests supports that transport.

The URI this proxy provides will be used by some other element to make a routing decision. This proxy, in general, has no way of knowing the capabilities of that element, so it must restrict itself to the mandatory elements of a SIP implementation: SIP URIs and either the TCP or UDP transports.

The URI placed in the Record-Route header field MUST resolve to the element inserting it (or a suitable stand-in) when the server location procedures of [4] are applied to it, so that subsequent requests reach the same SIP element. If the Request-URI contains a SIPS URI, or the topmost Route header field value (after the post processing of bullet 6) contains a SIPS URI, the URI placed into the Record-Route header field MUST be a SIPS URI. Furthermore, if the request was not received over TLS, the proxy MUST insert a Record-Route header field. In a similar fashion, a proxy that receives a request over TLS, but generates a request without a SIPS URI in the Request-URI or topmost Route header field value (after the post processing of bullet 6), MUST insert a Record-Route header field that is not a SIPS URI.

A proxy at a security perimeter must remain on the perimeter throughout the dialog.

If the URI placed in the Record-Route header field needs to be rewritten when it passes back through in a response, the URI MUST be distinct enough to locate at that time. (The request may spiral through this proxy, resulting in more than one Record-Route header field value being added). Item 8 of Section 16.7 recommends a mechanism to make the URI sufficiently distinct.

The proxy MAY include parameters in the Record-Route header field value. These will be echoed in some responses to the request such as the 200 (OK) responses to INVITE. Such parameters may be useful for keeping state in the message rather than the proxy.

If a proxy needs to be in the path of any type of dialog (such as one straddling a firewall), it SHOULD add a Record-Route header field value to every request with a method it does not understand since that method may have dialog semantics.

The URI a proxy places into a Record-Route header field is only valid for the lifetime of any dialog created by the transaction in which it occurs. A dialog-stateful proxy, for example, MAY refuse to accept future requests with that value in the Request-URI after the dialog has terminated. Non-dialog-stateful proxies, of course, have no concept of when the dialog has terminated, but they MAY encode enough information in the value to compare it against the dialog identifier of future requests and MAY reject requests not matching that information. Endpoints MUST NOT use a URI obtained from a Record-Route header field outside the dialog in which it was provided. See

Section 12 for more information on an endpoint's use of

Record-Route header fields.

Record-routing may be required by certain services where the proxy needs to observe all messages in a dialog. However, it slows down processing and impairs scalability and thus proxies should only record-route if required for a particular service.

The Record-Route process is designed to work for any SIP request that initiates a dialog. INVITE is the only such request in this specification, but extensions to the protocol MAY define others.

5. Add Additional Header Fields

The proxy MAY add any other appropriate header fields to the copy at this point.

6. Postprocess routing information

A proxy MAY have a local policy that mandates that a request visit a specific set of proxies before being delivered to the destination. A proxy MUST ensure that all such proxies are loose routers. Generally, this can only be known with certainty if the proxies are within the same administrative domain. This set of proxies is represented by a set of URIs (each of which contains the lr parameter). This set MUST be pushed into the Route header field of the copy ahead of any existing values, if present. If the Route header field is absent, it MUST be added, containing that list of URIs.

If the proxy has a local policy that mandates that the request visit one specific proxy, an alternative to pushing a Route value into the Route header field is to bypass the forwarding logic of item 10 below, and instead just send the request to the address, port, and transport for that specific proxy. If the request has a Route header field, this alternative MUST NOT be used unless it is known that next hop proxy is a loose router. Otherwise, this approach MAY be used, but the Route insertion mechanism above is preferred for its robustness, flexibility, generality and consistency of operation. Furthermore, if the Request-URI contains a SIPS URI, TLS MUST be used to communicate with that proxy.

If the copy contains a Route header field, the proxy MUST inspect the URI in its first value. If that URI does not contain an lr parameter, the proxy MUST modify the copy as follows:

- The proxy MUST place the Request-URI into the Route header field as the last value.

- The proxy MUST then place the first Route header field value into the Request-URI and remove that value from the Route header field.

Appending the Request-URI to the Route header field is part of a mechanism used to pass the information in that Request-URI through strict-routing elements. "Popping" the first Route

header field value into the Request-URI formats the message the way a strict-routing element expects to receive it (with its own URI in the Request-URI and the next location to visit in the first Route header field value).

7. Determine Next-Hop Address, Port, and Transport

   The proxy MAY have a local policy to send the request to a specific IP address, port, and transport, independent of the values of the Route and Request-URI. Such a policy MUST NOT be used if the proxy is not certain that the IP address, port, and transport correspond to a server that is a loose router. However, this mechanism for sending the request through a specific next hop is NOT RECOMMENDED; instead a Route header field should be used for that purpose as described above.

   In the absence of such an overriding mechanism, the proxy applies the procedures listed in [4] as follows to determine where to send the request. If the proxy has reformatted the request to send to a strict-routing element as described in step 6 above, the proxy MUST apply those procedures to the Request-URI of the request. Otherwise, the proxy MUST apply the procedures to the first value in the Route header field, if present, else the Request-URI. The procedures will produce an ordered set of (address, port, transport) tuples. Independently of which URI is being used as input to the procedures of [4], if the Request-URI specifies a SIPS resource, the proxy MUST follow the procedures of [4] as if the input URI were a SIPS URI.

   As described in [4], the proxy MUST attempt to deliver the message to the first tuple in that set, and proceed through the set in order until the delivery attempt succeeds.

   For each tuple attempted, the proxy MUST format the message as appropriate for the tuple and send the request using a new client transaction as detailed in steps 8 through 10.

   Since each attempt uses a new client transaction, it represents a new branch. Thus, the branch parameter provided with the Via header field inserted in step 8 MUST be different for each attempt.

   If the client transaction reports failure to send the request or a timeout from its state machine, the proxy continues to the next address in that ordered set. If the ordered set is exhausted, the request cannot be forwarded to this element in the target set. The proxy does not need to place anything in the response context, but otherwise acts as if this element of the target set returned a 408 (Request Timeout) final response.

8. Add a Via header field value

   The proxy MUST insert a Via header field value into the copy before the existing Via header field values. The construction of this value follows the same guidelines of Section 8.1.1.7. This implies that the proxy will compute its own branch

parameter, which will be globally unique for that branch, and contain the requisite magic cookie. Note that this implies that the branch parameter will be different for different instances of a spiraled or looped request through a proxy.

Proxies choosing to detect loops have an additional constraint in the value they use for construction of the branch parameter. A proxy choosing to detect loops SHOULD create a branch parameter separable into two parts by the implementation. The first part MUST satisfy the constraints of Section 8.1.1.7 as described above. The second is used to perform loop detection and distinguish loops from spirals.

Loop detection is performed by verifying that, when a request returns to a proxy, those fields having an impact on the processing of the request have not changed. The value placed in this part of the branch parameter SHOULD reflect all of those fields (including any Route, Proxy-Require and Proxy-Authorization header fields). This is to ensure that if the request is routed back to the proxy and one of those fields changes, it is treated as a spiral and not a loop (see Section 16.3). A common way to create this value is to compute a cryptographic hash of the To tag, From tag, Call-ID header field, the Request-URI of the request received (before translation), the topmost Via header, and the sequence number from the CSeq header field, in addition to any Proxy-Require and Proxy-Authorization header fields that may be present. The

algorithm used to compute the hash is implementation-dependent, but MD5 (RFC 1321 [35]), expressed in hexadecimal, is a reasonable choice. (Base64 is not permissible for a token.)

If a proxy wishes to detect loops, the "branch" parameter it supplies MUST depend on all information affecting processing of a request, including the incoming Request-URI and any header fields affecting the request's admission or routing. This is necessary to distinguish looped requests from requests whose routing parameters have changed before returning to this server.

The request method MUST NOT be included in the calculation of the branch parameter. In particular, CANCEL and ACK requests (for non-2xx responses) MUST have the same branch value as the corresponding request they cancel or acknowledge. The branch parameter is used in correlating those requests at the server handling them (see Sections 17.2.3 and 9.2).

9. Add a Content-Length header field if necessary

If the request will be sent to the next hop using a stream-based transport and the copy contains no Content-Length header field, the proxy MUST insert one with the correct value for the body of the request (see Section 20.14).

10. Forward Request

A stateful proxy MUST create a new client transaction for this request as described in Section 17.1 and instructs the transaction to send the request using the address, port and transport determined in step 7.

11. Set timer C

In order to handle the case where an INVITE request never generates a final response, the TU uses a timer which is called timer C. Timer C MUST be set for each client transaction when an INVITE request is proxied. The timer MUST be larger than 3 minutes. Section 16.7 bullet 2 discusses how this timer is updated with provisional responses, and Section 16.8 discusses processing when it fires.

16.7 Response Processing

When a response is received by an element, it first tries to locate a client transaction (Section 17.1.3) matching the response. If none is found, the element MUST process the response (even if it is an informational response) as a stateless proxy (described below). If a match is found, the response is handed to the client transaction.

Forwarding responses for which a client transaction (or more generally any knowledge of having sent an associated request) is not found improves robustness. In particular, it ensures that "late" 2xx responses to INVITE requests are forwarded properly.

As client transactions pass responses to the proxy layer, the following processing MUST take place:

1. Find the appropriate response context
2. Update timer C for provisional responses
3. Remove the topmost Via
4. Add the response to the response context
5. Check to see if this response should be forwarded immediately
6. When necessary, choose the best final response from the response context

If no final response has been forwarded after every client transaction associated with the response context has been terminated, the proxy must choose and forward the "best" response from those it has seen so far.

The following processing MUST be performed on each response that is forwarded. It is likely that more than one response to each request will be forwarded: at least each provisional and one final response.

7. Aggregate authorization header field values if necessary

8. Optionally rewrite Record-Route header field values

9. Forward the response

10. Generate any necessary CANCEL requests

Each of the above steps are detailed below:

1. Find Context

   The proxy locates the "response context" it created before forwarding the original request using the key described in Section 16.6. The remaining processing steps take place in this context.

2. Update timer C for provisional responses

   For an INVITE transaction, if the response is a provisional response with status codes 101 to 199 inclusive (i.e., anything but 100), the proxy MUST reset timer C for that client transaction. The timer MAY be reset to a different value, but this value MUST be greater than 3 minutes.

3. Via

   The proxy removes the topmost Via header field value from the response.

   If no Via header field values remain in the response, the response was meant for this element and MUST NOT be forwarded. The remainder of the processing described in this section is not performed on this message, the UAC processing rules described in Section 8.1.3 are followed instead (transport layer processing has already occurred).

   This will happen, for instance, when the element generates CANCEL requests as described in Section 10.

4. Add response to context

   Final responses received are stored in the response context until a final response is generated on the server transaction associated with this context. The response may be a candidate for the best final response to be returned on that server transaction. Information from this response may be needed in forming the best response, even if this response is not chosen.

   If the proxy chooses to recurse on any contacts in a 3xx response by adding them to the target set, it MUST remove them from the response before adding the response to the response context. However, a proxy SHOULD NOT recurse to a non-SIPS URI if the Request-URI of the original request was a SIPS URI. If

the proxy recurses on all of the contacts in a 3xx response, the proxy SHOULD NOT add the resulting contactless response to the response context.

Removing the contact before adding the response to the response context prevents the next element upstream from retrying a location this proxy has already attempted.

3xx responses may contain a mixture of SIP, SIPS, and non-SIP URIs. A proxy may choose to recurse on the SIP and SIPS URIs and place the remainder into the response context to be returned, potentially in the final response.

If a proxy receives a 416 (Unsupported URI Scheme) response to a request whose Request-URI scheme was not SIP, but the scheme in the original received request was SIP or SIPS (that is, the proxy changed the scheme from SIP or SIPS to something else when it proxied a request), the proxy SHOULD add a new URI to the target set. This URI SHOULD be a SIP URI version of the non-SIP URI that was just tried. In the case of the tel URL, this is accomplished by placing the telephone-subscriber part of the tel URL into the user part of the SIP URI, and setting the hostpart to the domain where the prior request was sent. See Section 19.1.6 for more detail on forming SIP URIs from tel URLs.

As with a 3xx response, if a proxy "recurses" on the 416 by trying a SIP or SIPS URI instead, the 416 response SHOULD NOT be added to the response context.

5. Check response for forwarding

Until a final response has been sent on the server transaction, the following responses MUST be forwarded immediately:

- Any provisional response other than 100 (Trying)

- Any 2xx response

If a 6xx response is received, it is not immediately forwarded, but the stateful proxy SHOULD cancel all client pending transactions as described in Section 10, and it MUST NOT create any new branches in this context.

This is a change from RFC 2543, which mandated that the proxy was to forward the 6xx response immediately. For an INVITE transaction, this approach had the problem that a 2xx response could arrive on another branch, in which case the proxy would

have to forward the 2xx. The result was that the UAC could receive a 6xx response followed by a 2xx response, which should never be allowed to happen. Under the new rules, upon receiving a 6xx, a proxy will issue a CANCEL request, which will generally result in 487 responses from all outstanding client transactions, and then at that point the 6xx is forwarded upstream.

After a final response has been sent on the server transaction, the following responses MUST be forwarded immediately:

- Any 2xx response to an INVITE request

A stateful proxy MUST NOT immediately forward any other responses. In particular, a stateful proxy MUST NOT forward any 100 (Trying) response. Those responses that are candidates for forwarding later as the "best" response have been gathered as described in step "Add Response to Context".

Any response chosen for immediate forwarding MUST be processed as described in steps "Aggregate Authorization Header Field Values" through "Record-Route".

This step, combined with the next, ensures that a stateful proxy will forward exactly one final response to a non-INVITE request, and either exactly one non-2xx response or one or more 2xx responses to an INVITE request.

6. Choosing the best response

A stateful proxy MUST send a final response to a response context's server transaction if no final responses have been immediately forwarded by the above rules and all client transactions in this response context have been terminated.

The stateful proxy MUST choose the "best" final response among those received and stored in the response context.

If there are no final responses in the context, the proxy MUST send a 408 (Request Timeout) response to the server transaction.

Otherwise, the proxy MUST forward a response from the responses stored in the response context. It MUST choose from the 6xx class responses if any exist in the context. If no 6xx class responses are present, the proxy SHOULD choose from the lowest response class stored in the response context. The proxy MAY select any response within that chosen class. The proxy SHOULD

give preference to responses that provide information affecting resubmission of this request, such as 401, 407, 415, 420, and 484 if the 4xx class is chosen.

A proxy which receives a 503 (Service Unavailable) response SHOULD NOT forward it upstream unless it can determine that any subsequent requests it might proxy will also generate a 503. In other words, forwarding a 503 means that the proxy knows it

cannot service any requests, not just the one for the Request-URI in the request which generated the 503. If the only response that was received is a 503, the proxy SHOULD generate a 500 response and forward that upstream.

The forwarded response MUST be processed as described in steps "Aggregate Authorization Header Field Values" through "Record-Route".

For example, if a proxy forwarded a request to 4 locations, and received 503, 407, 501, and 404 responses, it may choose to forward the 407 (Proxy Authentication Required) response.

1xx and 2xx responses may be involved in the establishment of dialogs. When a request does not contain a To tag, the To tag in the response is used by the UAC to distinguish multiple responses to a dialog creating request. A proxy MUST NOT insert a tag into the To header field of a 1xx or 2xx response if the request did not contain one. A proxy MUST NOT modify the tag in the To header field of a 1xx or 2xx response.

Since a proxy may not insert a tag into the To header field of a 1xx response to a request that did not contain one, it cannot issue non-100 provisional responses on its own. However, it can branch the request to a UAS sharing the same element as the proxy. This UAS can return its own provisional responses, entering into an early dialog with the initiator of the request. The UAS does not have to be a discreet process from the proxy. It could be a virtual UAS implemented in the same code space as the proxy.

3-6xx responses are delivered hop-by-hop. When issuing a 3-6xx response, the element is effectively acting as a UAS, issuing its own response, usually based on the responses received from downstream elements. An element SHOULD preserve the To tag when simply forwarding a 3-6xx response to a request that did not contain a To tag.

A proxy MUST NOT modify the To tag in any forwarded response to a request that contains a To tag.

While it makes no difference to the upstream elements if the proxy replaced the To tag in a forwarded 3-6xx response, preserving the original tag may assist with debugging.

When the proxy is aggregating information from several responses, choosing a To tag from among them is arbitrary, and generating a new To tag may make debugging easier. This happens, for instance, when combining 401 (Unauthorized) and 407 (Proxy Authentication Required) challenges, or combining Contact values from unencrypted and unauthenticated 3xx responses.

7. Aggregate Authorization Header Field Values

If the selected response is a 401 (Unauthorized) or 407 (Proxy Authentication Required), the proxy MUST collect any WWW-Authenticate and Proxy-Authenticate header field values from

all other 401 (Unauthorized) and 407 (Proxy Authentication Required) responses received so far in this response context and add them to this response without modification before forwarding. The resulting 401 (Unauthorized) or 407 (Proxy Authentication Required) response could have several WWW-Authenticate AND Proxy-Authenticate header field values.

This is necessary because any or all of the destinations the request was forwarded to may have requested credentials. The client needs to receive all of those challenges and supply credentials for each of them when it retries the request. Motivation for this behavior is provided in Section 26.

8. Record-Route

If the selected response contains a Record-Route header field value originally provided by this proxy, the proxy MAY choose to rewrite the value before forwarding the response. This allows the proxy to provide different URIs for itself to the next upstream and downstream elements. A proxy may choose to use this mechanism for any reason. For instance, it is useful for multi-homed hosts.

If the proxy received the request over TLS, and sent it out over a non-TLS connection, the proxy MUST rewrite the URI in the Record-Route header field to be a SIPS URI. If the proxy received the request over a non-TLS connection, and sent it out over TLS, the proxy MUST rewrite the URI in the Record-Route header field to be a SIP URI.

The new URI provided by the proxy MUST satisfy the same constraints on URIs placed in Record-Route header fields in requests (see Step 4 of Section 16.6) with the following modifications:

The URI SHOULD NOT contain the transport parameter unless the proxy has knowledge that the next upstream (as opposed to downstream) element that will be in the path of subsequent requests supports that transport.

When a proxy does decide to modify the Record-Route header field in the response, one of the operations it performs is locating the Record-Route value that it had inserted. If the request spiraled, and the proxy inserted a Record-Route value in each iteration of the spiral, locating the correct value in the response (which must be the proper iteration in the reverse direction) is tricky. The rules above recommend that a proxy wishing to rewrite Record-Route header field values insert sufficiently distinct URIs into the Record-Route header field so that the right one may be selected for rewriting. A RECOMMENDED mechanism to achieve this is for the proxy to append a unique identifier for the proxy instance to the user portion of the URI.

When the response arrives, the proxy modifies the first Record-Route whose identifier matches the proxy instance. The

modification results in a URI without this piece of data appended to the user portion of the URI. Upon the next iteration, the same algorithm (find the topmost Record-Route header field value with the parameter) will correctly extract the next Record-Route header field value inserted by that proxy.

Not every response to a request to which a proxy adds a Record-Route header field value will contain a Record-Route header field. If the response does contain a Record-Route header field, it will contain the value the proxy added.

9. Forward response

After performing the processing described in steps "Aggregate Authorization Header Field Values" through "Record-Route", the proxy MAY perform any feature specific manipulations on the selected response. The proxy MUST NOT add to, modify, or remove the message body. Unless otherwise specified, the proxy MUST NOT remove any header field values other than the Via header field value discussed in Section 16.7 Item 3. In particular, the proxy MUST NOT remove any "received" parameter

it may have added to the next Via header field value while processing the request associated with this response. The proxy MUST pass the response to the server transaction associated with the response context. This will result in the response being sent to the location now indicated in the topmost Via header field value. If the server transaction is no longer available to handle the transmission, the element MUST forward the response statelessly by sending it to the server transport. The server transaction might indicate failure to send the response or signal a timeout in its state machine. These errors would be logged for diagnostic purposes as appropriate, but the protocol requires no remedial action from the proxy.

The proxy MUST maintain the response context until all of its associated transactions have been terminated, even after forwarding a final response.

10. Generate CANCELs

If the forwarded response was a final response, the proxy MUST generate a CANCEL request for all pending client transactions associated with this response context. A proxy SHOULD also generate a CANCEL request for all pending client transactions associated with this response context when it receives a 6xx response. A pending client transaction is one that has received a provisional response, but no final response (it is in the proceeding state) and has not had an associated CANCEL generated for it. Generating CANCEL requests is described in Section 9.1.

The requirement to CANCEL pending client transactions upon forwarding a final response does not guarantee that an endpoint will not receive multiple 200 (OK) responses to an INVITE. 200 (OK) responses on more than one branch may be generated before

the CANCEL requests can be sent and processed. Further, it is reasonable to expect that a future extension may override this requirement to issue CANCEL requests.

16.8 Processing Timer C

If timer C should fire, the proxy MUST either reset the timer with any value it chooses, or terminate the client transaction. If the client transaction has received a provisional response, the proxy MUST generate a CANCEL request matching that transaction. If the client transaction has not received a provisional response, the proxy MUST behave as if the transaction received a 408 (Request Timeout) response.

Allowing the proxy to reset the timer allows the proxy to dynamically extend the transaction's lifetime based on current conditions (such as utilization) when the timer fires.

16.9 Handling Transport Errors

If the transport layer notifies a proxy of an error when it tries to forward a request (see Section 18.4), the proxy MUST behave as if the forwarded request received a 503 (Service Unavailable) response.

If the proxy is notified of an error when forwarding a response, it drops the response. The proxy SHOULD NOT cancel any outstanding client transactions associated with this response context due to this notification.

If a proxy cancels its outstanding client transactions, a single malicious or misbehaving client can cause all transactions to fail through its Via header field.

16.10 CANCEL Processing

A stateful proxy MAY generate a CANCEL to any other request it has generated at any time (subject to receiving a provisional response to that request as described in section 9.1). A proxy MUST cancel any pending client transactions associated with a response context when it receives a matching CANCEL request.

A stateful proxy MAY generate CANCEL requests for pending INVITE client transactions based on the period specified in the INVITE's Expires header field elapsing. However, this is generally unnecessary since the endpoints involved will take care of signaling the end of the transaction.

While a CANCEL request is handled in a stateful proxy by its own server transaction, a new response context is not created for it. Instead, the proxy layer searches its existing response contexts for the server transaction handling the request associated with this CANCEL. If a matching response context is found, the element MUST immediately return a 200 (OK) response to the CANCEL request. In this case, the element is acting as a user agent server as defined in Section 8.2. Furthermore, the element MUST generate CANCEL requests for all pending client transactions in the context as described in Section 16.7 step 10.

If a response context is not found, the element does not have any knowledge of the request to apply the CANCEL to. It MUST statelessly forward the CANCEL request (it may have statelessly forwarded the associated request previously).

16.11 Stateless Proxy

When acting statelessly, a proxy is a simple message forwarder. Much of the processing performed when acting statelessly is the same as when behaving statefully. The differences are detailed here.

A stateless proxy does not have any notion of a transaction, or of the response context used to describe stateful proxy behavior. Instead, the stateless proxy takes messages, both requests and responses, directly from the transport layer (See section 18). As a result, stateless proxies do not retransmit messages on their own. They do, however, forward all retransmissions they receive (they do not have the ability to distinguish a retransmission from the original message). Furthermore, when handling a request statelessly, an element MUST NOT generate its own 100 (Trying) or any other provisional response.

A stateless proxy MUST validate a request as described in Section 16.3

A stateless proxy MUST follow the request processing steps described in Sections 16.4 through 16.5 with the following exception:

- A stateless proxy MUST choose one and only one target from the target set. This choice MUST only rely on fields in the message and time-invariant properties of the server. In particular, a retransmitted request MUST be forwarded to the same destination each time it is processed. Furthermore, CANCEL and non-Routed ACK requests MUST generate the same choice as their associated INVITE.

A stateless proxy MUST follow the request processing steps described in Section 16.6 with the following exceptions:

- The requirement for unique branch IDs across space and time applies to stateless proxies as well. However, a stateless proxy cannot simply use a random number generator to compute the first component of the branch ID, as described in Section 16.6 bullet 8. This is because retransmissions of a request need to have the same value, and a stateless proxy cannot tell a retransmission from the original request. Therefore, the component of the branch parameter that makes it unique MUST be the same each time a retransmitted request is forwarded. Thus for a stateless proxy, the branch parameter MUST be computed as a combinatoric function of message parameters which are invariant on retransmission.

The stateless proxy MAY use any technique it likes to guarantee uniqueness of its branch IDs across transactions. However, the following procedure is RECOMMENDED. The proxy examines the branch ID in the topmost Via header field of the received request. If it begins with the magic cookie, the first component of the branch ID of the outgoing request is computed as a hash of the received branch ID. Otherwise, the first component of the branch ID is computed as a hash of the topmost Via, the tag in the To header field, the tag in the From header field, the Call-ID header field, the CSeq number (but not method), and the Request-URI from the received request. One of these fields will always vary across two different transactions.

- All other message transformations specified in Section 16.6 MUST result in the same transformation of a retransmitted request. In particular, if the proxy inserts a Record-Route value or pushes URIs into the Route header field, it MUST place the same values in retransmissions of the request. As for the Via branch parameter, this implies that the transformations MUST be based on time-invariant configuration or retransmission-invariant properties of the request.

- A stateless proxy determines where to forward the request as described for stateful proxies in Section 16.6 Item 10. The request is sent directly to the transport layer instead of through a client transaction.

  Since a stateless proxy must forward retransmitted requests to the same destination and add identical branch parameters to each of them, it can only use information from the message itself and time-invariant configuration data for those calculations. If the configuration state is not time-invariant (for example, if a routing table is updated) any requests that could be affected by the change may not be forwarded statelessly during an interval equal to the transaction timeout window before or after the change. The method of processing the affected requests in that interval is an implementation decision. A common solution is to forward them transaction statefully.

Stateless proxies MUST NOT perform special processing for CANCEL requests. They are processed by the above rules as any other requests. In particular, a stateless proxy applies the same Route header field processing to CANCEL requests that it applies to any other request.

Response processing as described in Section 16.7 does not apply to a proxy behaving statelessly. When a response arrives at a stateless proxy, the proxy MUST inspect the sent-by value in the first (topmost) Via header field value. If that address matches the proxy, (it equals a value this proxy has inserted into previous requests) the proxy MUST remove that header field value from the response and

forward the result to the location indicated in the next Via header field value. The proxy MUST NOT add to, modify, or remove the message body. Unless specified otherwise, the proxy MUST NOT remove any other header field values. If the address does not match the proxy, the message MUST be silently discarded.

16.12 Summary of Proxy Route Processing

In the absence of local policy to the contrary, the processing a proxy performs on a request containing a Route header field can be summarized in the following steps.

1. The proxy will inspect the Request-URI. If it indicates a resource owned by this proxy, the proxy will replace it with the results of running a location service. Otherwise, the proxy will not change the Request-URI.

2. The proxy will inspect the URI in the topmost Route header field value. If it indicates this proxy, the proxy removes it from the Route header field (this route node has been reached).

3. The proxy will forward the request to the resource indicated by the URI in the topmost Route header field value or in the Request-URI if no Route header field is present. The proxy determines the address, port and transport to use when forwarding the request by applying the procedures in [4] to that URI.

If no strict-routing elements are encountered on the path of the request, the Request-URI will always indicate the target of the request.

16.12.1 Examples

16.12.1.1 Basic SIP Trapezoid

This scenario is the basic SIP trapezoid, U1 -> P1 -> P2 -> U2, with both proxies record-routing. Here is the flow.

U1 sends:

```
INVITE sip:callee@domain.com SIP/2.0
Contact: sip:caller@u1.example.com
```

to P1. P1 is an outbound proxy. P1 is not responsible for domain.com, so it looks it up in DNS and sends it there. It also adds a Record-Route header field value:

```
INVITE sip:callee@domain.com SIP/2.0
Contact: sip:caller@u1.example.com
Record-Route: <sip:p1.example.com;lr>
```

P2 gets this. It is responsible for domain.com so it runs a location service and rewrites the Request-URI. It also adds a Record-Route

header field value. There is no Route header field, so it resolves the new Request-URI to determine where to send the request:

```
INVITE sip:callee@u2.domain.com SIP/2.0
Contact: sip:caller@u1.example.com
Record-Route: <sip:p2.domain.com;lr>
Record-Route: <sip:p1.example.com;lr>
```

The callee at u2.domain.com gets this and responds with a 200 OK:

```
SIP/2.0 200 OK
Contact: sip:callee@u2.domain.com
Record-Route: <sip:p2.domain.com;lr>
Record-Route: <sip:p1.example.com;lr>
```

The callee at u2 also sets its dialog state's remote target URI to sip:caller@u1.example.com and its route set to:

```
(<sip:p2.domain.com;lr>,<sip:p1.example.com;lr>)
```

This is forwarded by P2 to P1 to U1 as normal. Now, U1 sets its dialog state's remote target URI to sip:callee@u2.domain.com and its route set to:

```
(<sip:p1.example.com;lr>,<sip:p2.domain.com;lr>)
```

Since all the route set elements contain the lr parameter, U1 constructs the following BYE request:

```
BYE sip:callee@u2.domain.com SIP/2.0
Route: <sip:p1.example.com;lr>,<sip:p2.domain.com;lr>
```


As any other element (including proxies) would do, it resolves the URI in the topmost Route header field value using DNS to determine where to send the request. This goes to P1. P1 notices that it is not responsible for the resource indicated in the Request-URI so it doesn't change it. It does see that it is the first value in the Route header field, so it removes that value, and forwards the request to P2:

```
BYE sip:callee@u2.domain.com SIP/2.0
Route: <sip:p2.domain.com;lr>
```

P2 also notices it is not responsible for the resource indicated by the Request-URI (it is responsible for domain.com, not u2.domain.com), so it doesn't change it. It does see itself in the first Route header field value, so it removes it and forwards the following to u2.domain.com based on a DNS lookup against the Request-URI:

```
BYE sip:callee@u2.domain.com SIP/2.0
```

16.12.1.2 Traversing a Strict-Routing Proxy

In this scenario, a dialog is established across four proxies, each of which adds Record-Route header field values. The third proxy

implements the strict-routing procedures specified in RFC 2543 and many works in progress.

```
U1->P1->P2->P3->P4->U2
```

The INVITE arriving at U2 contains:

```
INVITE sip:callee@u2.domain.com SIP/2.0
Contact: sip:caller@u1.example.com
Record-Route: <sip:p4.domain.com;lr>
Record-Route: <sip:p3.middle.com>
Record-Route: <sip:p2.example.com;lr>
Record-Route: <sip:p1.example.com;lr>
```

Which U2 responds to with a 200 OK. Later, U2 sends the following BYE request to P4 based on the first Route header field value.

```
BYE sip:caller@u1.example.com SIP/2.0
Route: <sip:p4.domain.com;lr>
Route: <sip:p3.middle.com>
Route: <sip:p2.example.com;lr>
Route: <sip:p1.example.com;lr>
```


P4 is not responsible for the resource indicated in the Request-URI so it will leave it alone. It notices that it is the element in the first Route header field value so it removes it. It then prepares to send the request based on the now first Route header field value of sip:p3.middle.com, but it notices that this URI does not contain the lr parameter, so before sending, it reformats the request to be:

```
BYE sip:p3.middle.com SIP/2.0
Route: <sip:p2.example.com;lr>
Route: <sip:p1.example.com;lr>
Route: <sip:caller@u1.example.com>
```

P3 is a strict router, so it forwards the following to P2:

```
BYE sip:p2.example.com;lr SIP/2.0
Route: <sip:p1.example.com;lr>
Route: <sip:caller@u1.example.com>
```

P2 sees the request-URI is a value it placed into a Record-Route header field, so before further processing, it rewrites the request to be:

```
BYE sip:caller@u1.example.com SIP/2.0
Route: <sip:p1.example.com;lr>
```

P2 is not responsible for u1.example.com, so it sends the request to P1 based on the resolution of the Route header field value.

P1 notices itself in the topmost Route header field value, so it removes it, resulting in:

```
BYE sip:caller@u1.example.com SIP/2.0
```

Since P1 is not responsible for u1.example.com and there is no Route header field, P1 will forward the request to u1.example.com based on the Request-URI.

16.12.1.3 Rewriting Record-Route Header Field Values

In this scenario, U1 and U2 are in different private namespaces and they enter a dialog through a proxy P1, which acts as a gateway between the namespaces.

U1->P1->U2

U1 sends:

```
INVITE sip:callee@gateway.leftprivatespace.com SIP/2.0
Contact: <sip:caller@u1.leftprivatespace.com>
```

P1 uses its location service and sends the following to U2:

```
INVITE sip:callee@rightprivatespace.com SIP/2.0
Contact: <sip:caller@u1.leftprivatespace.com>
Record-Route: <sip:gateway.rightprivatespace.com;lr>
```

U2 sends this 200 (OK) back to P1:

```
SIP/2.0 200 OK
Contact: <sip:callee@u2.rightprivatespace.com>
Record-Route: <sip:gateway.rightprivatespace.com;lr>
```

P1 rewrites its Record-Route header parameter to provide a value that U1 will find useful, and sends the following to U1:

```
SIP/2.0 200 OK
Contact: <sip:callee@u2.rightprivatespace.com>
Record-Route: <sip:gateway.leftprivatespace.com;lr>
```

Later, U1 sends the following BYE request to P1:

```
BYE sip:callee@u2.rightprivatespace.com SIP/2.0
Route: <sip:gateway.leftprivatespace.com;lr>
```

which P1 forwards to U2 as:

```
BYE sip:callee@u2.rightprivatespace.com SIP/2.0
```

17 Transactions

SIP is a transactional protocol: interactions between components take place in a series of independent message exchanges. Specifically, a SIP transaction consists of a single request and any responses to that request, which include zero or more provisional responses and one or more final responses. In the case of a transaction where the request was an INVITE (known as an INVITE transaction), the transaction also includes the ACK only if the final response was not

a 2xx response. If the response was a 2xx, the ACK is not considered part of the transaction.

The reason for this separation is rooted in the importance of delivering all 200 (OK) responses to an INVITE to the UAC. To deliver them all to the UAC, the UAS alone takes responsibility

for retransmitting them (see Section 13.3.1.4), and the UAC alone takes responsibility for acknowledging them with ACK (see Section 13.2.2.4). Since this ACK is retransmitted only by the UAC, it is effectively considered its own transaction.

Transactions have a client side and a server side. The client side is known as a client transaction and the server side as a server transaction. The client transaction sends the request, and the server transaction sends the response. The client and server transactions are logical functions that are embedded in any number of elements. Specifically, they exist within user agents and stateful proxy servers. Consider the example in Section 4. In this example, the UAC executes the client transaction, and its outbound proxy executes the server transaction. The outbound proxy also executes a client transaction, which sends the request to a server transaction in the inbound proxy. That proxy also executes a client transaction, which in turn sends the request to a server transaction in the UAS. This is shown in Figure 4.

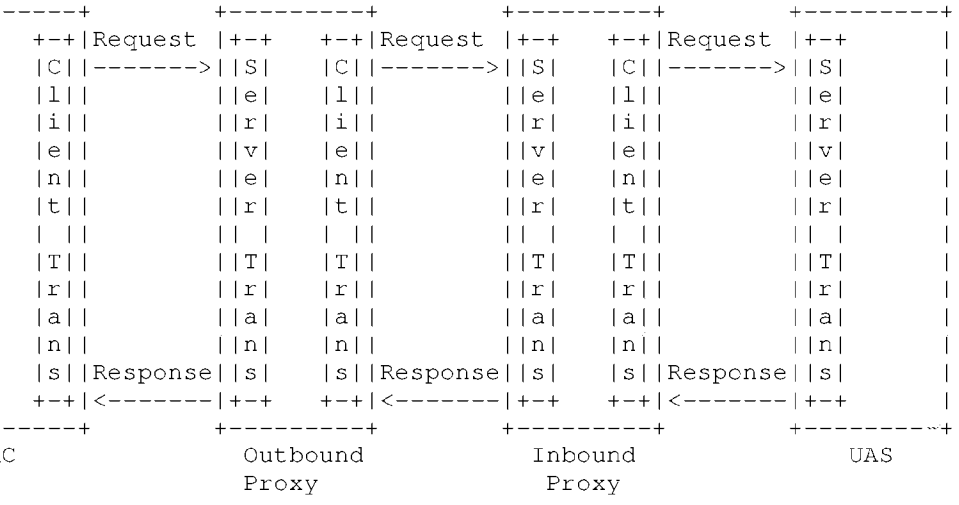


Figure 4: Transaction relationships

A stateless proxy does not contain a client or server transaction. The transaction exists between the UA or stateful proxy on one side, and the UA or stateful proxy on the other side. As far as SIP transactions are concerned, stateless proxies are effectively transparent. The purpose of the client transaction is to receive a request from the element in which the client is embedded (call this element the "Transaction User" or TU; it can be a UA or a stateful proxy), and reliably deliver the request to a server transaction.

The client transaction is also responsible for receiving responses and delivering them to the TU, filtering out any response retransmissions or disallowed responses (such as a response to ACK). Additionally, in the case of an INVITE request, the client transaction is responsible for generating the ACK request for any final response accepting a 2xx response.

Similarly, the purpose of the server transaction is to receive requests from the transport layer and deliver them to the TU. The server transaction filters any request retransmissions from the network. The server transaction accepts responses from the TU and delivers them to the transport layer for transmission over the network. In the case of an INVITE transaction, it absorbs the ACK request for any final response excepting a 2xx response.

The 2xx response and its ACK receive special treatment. This response is retransmitted only by a UAS, and its ACK generated only by the UAC. This end-to-end treatment is needed so that a caller knows the entire set of users that have accepted the call. Because of this special handling, retransmissions of the 2xx response are handled by the UA core, not the transaction layer. Similarly, generation of the ACK for the 2xx is handled by the UA core. Each proxy along the path merely forwards each 2xx response to INVITE and its corresponding ACK.

17.1 Client Transaction

The client transaction provides its functionality through the maintenance of a state machine.

The TU communicates with the client transaction through a simple interface. When the TU wishes to initiate a new transaction, it creates a client transaction and passes it the SIP request to send and an IP address, port, and transport to which to send it. The client transaction begins execution of its state machine. Valid responses are passed up to the TU from the client transaction.

There are two types of client transaction state machines, depending on the method of the request passed by the TU. One handles client transactions for INVITE requests. This type of machine is referred to as an INVITE client transaction. Another type handles client transactions for all requests except INVITE and ACK. This is referred to as a non-INVITE client transaction. There is no client transaction for ACK. If the TU wishes to send an ACK, it passes one directly to the transport layer for transmission.

The INVITE transaction is different from those of other methods because of its extended duration. Normally, human input is required in order to respond to an INVITE. The long delays expected for sending a response argue for a three-way handshake. On the other

hand, requests of other methods are expected to complete rapidly. Because of the non-INVITE transaction's reliance on a two-way handshake, TUs SHOULD respond immediately to non-INVITE requests.

17.1.1 INVITE Client Transaction

17.1.1.1 Overview of INVITE Transaction

The INVITE transaction consists of a three-way handshake. The client transaction sends an INVITE, the server transaction sends responses, and the client transaction sends an ACK. For unreliable transports (such as UDP), the client transaction retransmits requests at an interval that starts at T1 seconds and doubles after every retransmission. T1 is an estimate of the round-trip time (RTT), and it defaults to 500 ms. Nearly all of the transaction timers described here scale with T1, and changing T1 adjusts their values. The request is not retransmitted over reliable transports. After receiving a 1xx response, any retransmissions cease altogether, and the client waits for further responses. The server transaction can send additional 1xx responses, which are not transmitted reliably by the server transaction. Eventually, the server transaction decides to send a final response. For unreliable transports, that response is retransmitted periodically, and for reliable transports, it is sent once. For each final response that is received at the client transaction, the client transaction sends an ACK, the purpose of which is to quench retransmissions of the response.

17.1.1.2 Formal Description

The state machine for the INVITE client transaction is shown in Figure 5. The initial state, "calling", MUST be entered when the TU initiates a new client transaction with an INVITE request. The client transaction MUST pass the request to the transport layer for transmission (see Section 18). If an unreliable transport is being used, the client transaction MUST start timer A with a value of T1. If a reliable transport is being used, the client transaction SHOULD NOT start timer A (Timer A controls request retransmissions). For any transport, the client transaction MUST start timer B with a value of 64*T1 seconds (Timer B controls transaction timeouts).

When timer A fires, the client transaction MUST retransmit the request by passing it to the transport layer, and MUST reset the timer with a value of 2*T1. The formal definition of retransmit

within the context of the transaction layer is to take the message previously sent to the transport layer and pass it to the transport layer once more.

When timer A fires 2*T1 seconds later, the request MUST be retransmitted again (assuming the client transaction is still in this state). This process MUST continue so that the request is retransmitted with intervals that double after each transmission. These retransmissions SHOULD only be done while the client transaction is in the "calling" state.

The default value for T1 is 500 ms. T1 is an estimate of the RTT between the client and server transactions. Elements MAY (though it

is NOT RECOMMENDED) use smaller values of T1 within closed, private networks that do not permit general Internet connection. T1 MAY be chosen larger, and this is RECOMMENDED if it is known in advance (such as on high latency access links) that the RTT is larger. Whatever the value of T1, the exponential backoffs on retransmissions described in this section MUST be used.

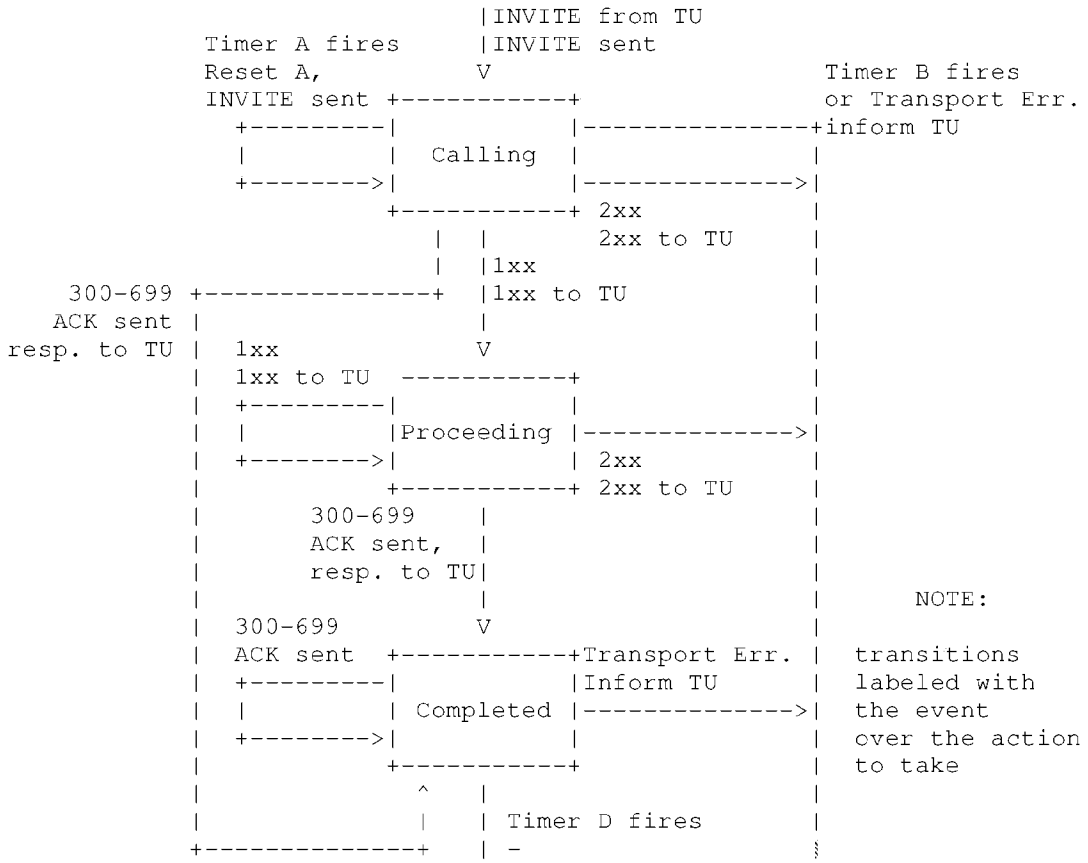
If the client transaction is still in the "Calling" state when timer B fires, the client transaction SHOULD inform the TU that a timeout has occurred. The client transaction MUST NOT generate an ACK. The value of 64*T1 is equal to the amount of time required to send seven requests in the case of an unreliable transport.

If the client transaction receives a provisional response while in the "Calling" state, it transitions to the "Proceeding" state. In the "Proceeding" state, the client transaction SHOULD NOT retransmit the request any longer. Furthermore, the provisional response MUST be passed to the TU. Any further provisional responses MUST be passed up to the TU while in the "Proceeding" state.

When in either the "Calling" or "Proceeding" states, reception of a response with status code from 300-699 MUST cause the client transaction to transition to "Completed". The client transaction MUST pass the received response up to the TU, and the client transaction MUST generate an ACK request, even if the transport is reliable (guidelines for constructing the ACK from the response are given in Section 17.1.1.3) and then pass the ACK to the transport layer for transmission. The ACK MUST be sent to the same address, port, and transport to which the original request was sent. The client transaction SHOULD start timer D when it enters the "Completed" state, with a value of at least 32 seconds for unreliable transports, and a value of zero seconds for reliable transports. Timer D reflects the amount of time that the server transaction can remain in the "Completed" state when unreliable transports are used. This is equal to Timer H in the INVITE server transaction, whose

default is 64*T1. However, the client transaction does not know the value of T1 in use by the server transaction, so an absolute minimum of 32s is used instead of basing Timer D on T1.

Any retransmissions of the final response that are received while in the "Completed" state MUST cause the ACK to be re-passed to the transport layer for retransmission, but the newly received response MUST NOT be passed up to the TU. A retransmission of the response is defined as any response which would match the same client transaction based on the rules of Section 17.1.3.


|INVITE from TU
Timer A fires
Reset A,
INVITE sent
|INVITE sent
V
Timer B fires
or Transport Err.
inform TU
Calling
2xx
2xx to TU
1xx
1xx to TU
300-699
ACK sent
resp. to TU
1xx
1xx to TU
Proceeding
2xx
2xx to TU
300-699
ACK sent,
resp. to TU
300-699
ACK sent
Transport Err.
Inform TU
Completed
Timer D fires
-
NOTE:
transitions
labeled with
the event
over the action
to take

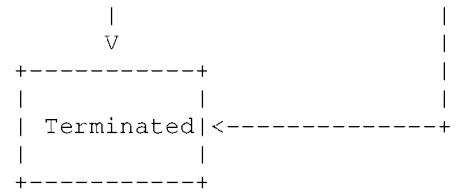


Figure 5: INVITE client transaction

If timer D fires while the client transaction is in the "Completed" state, the client transaction MUST move to the terminated state.

When in either the "Calling" or "Proceeding" states, reception of a 2xx response MUST cause the client transaction to enter the "Terminated" state, and the response MUST be passed up to the TU. The handling of this response depends on whether the TU is a proxy

core or a UAC core. A UAC core will handle generation of the ACK for this response, while a proxy core will always forward the 200 (OK) upstream. The differing treatment of 200 (OK) between proxy and UAC is the reason that handling of it does not take place in the transaction layer.

The client transaction MUST be destroyed the instant it enters the "Terminated" state. This is actually necessary to guarantee correct operation. The reason is that 2xx responses to an INVITE are treated differently; each one is forwarded by proxies, and the ACK handling in a UAC is different. Thus, each 2xx needs to be passed to a proxy core (so that it can be forwarded) and to a UAC core (so it can be acknowledged). No transaction layer processing takes place. Whenever a response is received by the transport, if the transport layer finds no matching client transaction (using the rules of Section 17.1.3), the response is passed directly to the core. Since the matching client transaction is destroyed by the first 2xx, subsequent 2xx will find no match and therefore be passed to the core.

17.1.1.3 Construction of the ACK Request

This section specifies the construction of ACK requests sent within the client transaction. A UAC core that generates an ACK for 2xx MUST instead follow the rules described in Section 13.

The ACK request constructed by the client transaction MUST contain values for the Call-ID, From, and Request-URI that are equal to the values of those header fields in the request passed to the transport by the client transaction (call this the "original request"). The To header field in the ACK MUST equal the To header field in the response being acknowledged, and therefore will usually differ from the To header field in the original request by the addition of the tag parameter. The ACK MUST contain a single Via header field, and this MUST be equal to the top Via header field of the original request. The CSeq header field in the ACK MUST contain the same value for the sequence number as was present in the original request, but the method parameter MUST be equal to "ACK".

If the INVITE request whose response is being acknowledged had Route header fields, those header fields MUST appear in the ACK. This is to ensure that the ACK can be routed properly through any downstream stateless proxies.

Although any request MAY contain a body, a body in an ACK is special since the request cannot be rejected if the body is not understood. Therefore, placement of bodies in ACK for non-2xx is NOT RECOMMENDED, but if done, the body types are restricted to any that appeared in the INVITE, assuming that the response to the INVITE was not 415. If it was, the body in the ACK MAY be any type listed in the Accept header field in the 415.

For example, consider the following request:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKkjshdyff
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=88sja8x
Max-Forwards: 70
Call-ID: 987asjd97y7atg
CSeq: 986759 INVITE
```

The ACK request for a non-2xx final response to this request would look like this:

```
ACK sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKkjshdyff
To: Bob <sip:bob@biloxi.com>;tag=99sa0xk
From: Alice <sip:alice@atlanta.com>;tag=88sja8x
Max-Forwards: 70
Call-ID: 987asjd97y7atg
CSeq: 986759 ACK
```

17.1.2 Non-INVITE Client Transaction

17.1.2.1 Overview of the non-INVITE Transaction

Non-INVITE transactions do not make use of ACK. They are simple request-response interactions. For unreliable transports, requests are retransmitted at an interval which starts at T1 and doubles until it hits T2. If a provisional response is received, retransmissions continue for unreliable transports, but at an interval of T2. The server transaction retransmits the last response it sent, which can be a provisional or final response, only when a retransmission of the request is received. This is why request retransmissions need to continue even after a provisional response; they are to ensure reliable delivery of the final response.

Unlike an INVITE transaction, a non-INVITE transaction has no special handling for the 2xx response. The result is that only a single 2xx response to a non-INVITE is ever delivered to a UAC.

17.1.2.2 Formal Description

The state machine for the non-INVITE client transaction is shown in Figure 6. It is very similar to the state machine for INVITE.

The "Trying" state is entered when the TU initiates a new client transaction with a request. When entering this state, the client transaction SHOULD set timer F to fire in 64*T1 seconds. The request MUST be passed to the transport layer for transmission. If an unreliable transport is in use, the client transaction MUST set timer E to fire in T1 seconds. If timer E fires while still in this state, the timer is reset, but this time with a value of MIN(2*T1, T2). When the timer fires again, it is reset to a MIN(4*T1, T2). This process continues so that retransmissions occur with an exponentially increasing interval that caps at T2. The default value of T2 is 4s, and it represents the amount of time a non-INVITE server transaction will take to respond to a request, if it does not respond immediately. For the default values of T1 and T2, this results in intervals of 500 ms, 1 s, 2 s, 4 s, 4 s, 4 s, etc.

If Timer F fires while the client transaction is still in the "Trying" state, the client transaction SHOULD inform the TU about the timeout, and then it SHOULD enter the "Terminated" state. If a provisional response is received while in the "Trying" state, the response MUST be passed to the TU, and then the client transaction SHOULD move to the "Proceeding" state. If a final response (status codes 200-699) is received while in the "Trying" state, the response MUST be passed to the TU, and the client transaction MUST transition to the "Completed" state.

If Timer E fires while in the "Proceeding" state, the request MUST be passed to the transport layer for retransmission, and Timer E MUST be reset with a value of T2 seconds. If timer F fires while in the "Proceeding" state, the TU MUST be informed of a timeout, and the client transaction MUST transition to the terminated state. If a final response (status codes 200-699) is received while in the "Proceeding" state, the response MUST be passed to the TU, and the client transaction MUST transition to the "Completed" state.

Once the client transaction enters the "Completed" state, it MUST set Timer K to fire in T4 seconds for unreliable transports, and zero seconds for reliable transports. The "Completed" state exists to buffer any additional response retransmissions that may be received (which is why the client transaction remains there only for

unreliable transports). T4 represents the amount of time the network will take to clear messages between client and server transactions.

The default value of T4 is 5s. A response is a retransmission when it matches the same transaction, using the rules specified in Section 17.1.3. If Timer K fires while in this state, the client transaction MUST transition to the "Terminated" state.

Once the transaction is in the terminated state, it MUST be destroyed immediately.

17.1.3 Matching Responses to Client Transactions

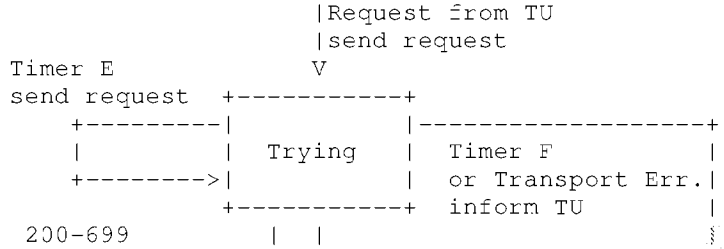
When the transport layer in the client receives a response, it has to determine which client transaction will handle the response, so that the processing of Sections 17.1.1 and 17.1.2 can take place. The branch parameter in the top Via header field is used for this purpose. A response matches a client transaction under two conditions:

1. If the response has the same value of the branch parameter in the top Via header field as the branch parameter in the top Via header field of the request that created the transaction.

2. If the method parameter in the CSeq header field matches the method of the request that created the transaction. The method is needed since a CANCEL request constitutes a different transaction, but shares the same value of the branch parameter.

If a request is sent via multicast, it is possible that it will generate multiple responses from different servers. These responses will all have the same branch parameter in the topmost Via, but vary in the To tag. The first response received, based on the rules above, will be used, and others will be viewed as retransmissions. That is not an error; multicast SIP provides only a rudimentary "single-hop-discovery-like" service that is limited to processing a single response. See Section 18.1.1 for details.

17.1.4 Handling Transport Errors



Page 1506
T-Mobile Exhibit 1002
Part 4 of 6

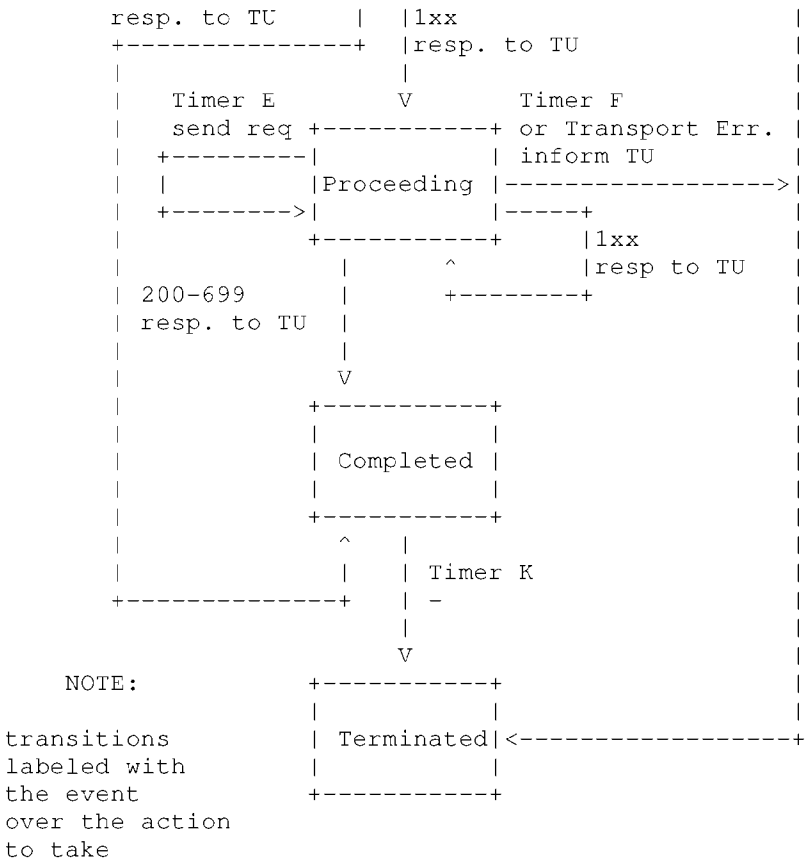


Figure 6: non-INVITE client transaction

When the client transaction sends a request to the transport layer to be sent, the following procedures are followed if the transport layer indicates a failure.

The client transaction SHOULD inform the TU that a transport failure has occurred, and the client transaction SHOULD transition directly to the "Terminated" state. The TU will handle the failover mechanisms described in [4].

17.2 Server Transaction

The server transaction is responsible for the delivery of requests to the TU and the reliable transmission of responses. It accomplishes this through a state machine. Server transactions are created by the core when a request is received, and transaction handling is desired for that request (this is not always the case).

As with the client transactions, the state machine depends on whether the received request is an INVITE request.

17.2.1 INVITE Server Transaction

The state diagram for the INVITE server transaction is shown in Figure 7.

When a server transaction is constructed for a request, it enters the "Proceeding" state. The server transaction MUST generate a 100 (Trying) response unless it knows that the TU will generate a provisional or final response within 200 ms, in which case it MAY generate a 100 (Trying) response. This provisional response is needed to quench request retransmissions rapidly in order to avoid network congestion. The 100 (Trying) response is constructed according to the procedures in Section 8.2.6, except that the insertion of tags in the To header field of the response (when none was present in the request) is downgraded from MAY to SHOULD NOT. The request MUST be passed to the TU.

The TU passes any number of provisional responses to the server transaction. So long as the server transaction is in the "Proceeding" state, each of these MUST be passed to the transport layer for transmission. They are not sent reliably by the transaction layer (they are not retransmitted by it) and do not cause a change in the state of the server transaction. If a request retransmission is received while in the "Proceeding" state, the most recent provisional response that was received from the TU MUST be passed to the transport layer for retransmission. A request is a retransmission if it matches the same server transaction based on the rules of Section 17.2.3.

If, while in the "Proceeding" state, the TU passes a 2xx response to the server transaction, the server transaction MUST pass this response to the transport layer for transmission. It is not

retransmitted by the server transaction; retransmissions of 2xx responses are handled by the TU. The server transaction MUST then transition to the "Terminated" state.

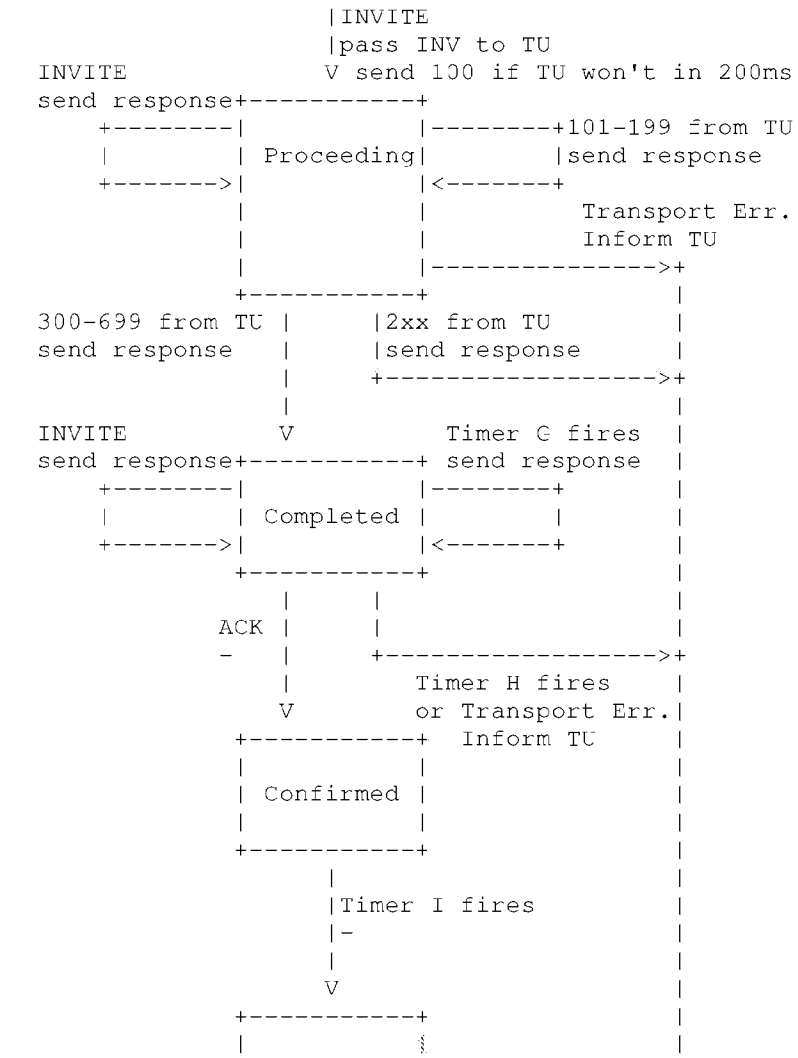
While in the "Proceeding" state, if the TU passes a response with status code from 300 to 699 to the server transaction, the response MUST be passed to the transport layer for transmission, and the state machine MUST enter the "Completed" state. For unreliable transports, timer G is set to fire in T1 seconds, and is not set to fire for reliable transports.

> This is a change from RFC 2543, where responses were always retransmitted, even over reliable transports.

When the "Completed" state is entered, timer H MUST be set to fire in 64*T1 seconds for all transports. Timer H determines when the server transaction abandons retransmitting the response. Its value is chosen to equal Timer B, the amount of time a client transaction will continue to retry sending a request. If timer G fires, the response is passed to the transport layer once more for retransmission, and timer G is set to fire in MIN(2*T1, T2) seconds. From then on, when timer G fires, the response is passed to the transport again for transmission, and timer G is reset with a value that doubles, unless that value exceeds T2, in which case it is reset with the value of T2. This is identical to the retransmit behavior for requests in the "Trying" state of the non-INVITE client transaction. Furthermore, while in the "Completed" state, if a request retransmission is received, the server SHOULD pass the response to the transport for retransmission.

If an ACK is received while the server transaction is in the "Completed" state, the server transaction MUST transition to the "Confirmed" state. As Timer G is ignored in this state, any retransmissions of the response will cease.

If timer H fires while in the "Completed" state, it implies that the ACK was never received. In this case, the server transaction MUST transition to the "Terminated" state, and MUST indicate to the TU that a transaction failure has occurred.



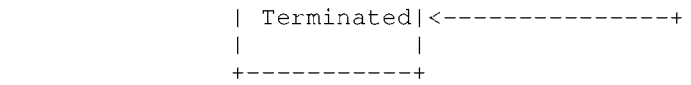


Figure 7: INVITE server transaction

The purpose of the "Confirmed" state is to absorb any additional ACK messages that arrive, triggered from retransmissions of the final response. When this state is entered, timer I is set to fire in T4 seconds for unreliable transports, and zero seconds for reliable transports. Once timer I fires, the server MUST transition to the "Terminated" state.

Once the transaction is in the "Terminated" state, it MUST be destroyed immediately. As with client transactions, this is needed to ensure reliability of the 2xx responses to INVITE.

17.2.2 Non-INVITE Server Transaction

The state machine for the non-INVITE server transaction is shown in Figure 8.

The state machine is initialized in the "Trying" state and is passed a request other than INVITE or ACK when initialized. This request is passed up to the TU. Once in the "Trying" state, any further request retransmissions are discarded. A request is a retransmission if it matches the same server transaction, using the rules specified in Section 17.2.3.

While in the "Trying" state, if the TU passes a provisional response to the server transaction, the server transaction MUST enter the "Proceeding" state. The response MUST be passed to the transport layer for transmission. Any further provisional responses that are received from the TU while in the "Proceeding" state MUST be passed to the transport layer for transmission. If a retransmission of the request is received while in the "Proceeding" state, the most recently sent provisional response MUST be passed to the transport layer for retransmission. If the TU passes a final response (status codes 200-699) to the server while in the "Proceeding" state, the transaction MUST enter the "Completed" state, and the response MUST be passed to the transport layer for transmission.

When the server transaction enters the "Completed" state, it MUST set Timer J to fire in 64*T1 seconds for unreliable transports, and zero seconds for reliable transports. While in the "Completed" state, the server transaction MUST pass the final response to the transport layer for retransmission whenever a retransmission of the request is received. Any other final responses passed by the TU to the server transaction MUST be discarded while in the "Completed" state. The server transaction remains in this state until Timer J fires, at which point it MUST transition to the "Terminated" state.

The server transaction MUST be destroyed the instant it enters the

"Terminated" state.

17.2.3 Matching Requests to Server Transactions

When a request is received from the network by the server, it has to be matched to an existing transaction. This is accomplished in the following manner.

The branch parameter in the topmost Via header field of the request is examined. If it is present and begins with the magic cookie "z9hG4bK", the request was generated by a client transaction compliant to this specification. Therefore, the branch parameter will be unique across all transactions sent by that client. The request matches a transaction if:

1. the branch parameter in the request is equal to the one in the top Via header field of the request that created the transaction, and

2. the sent-by value in the top Via of the request is equal to the one in the request that created the transaction, and

3. the method of the request matches the one that created the transaction, except for ACK, where the method of the request that created the transaction is INVITE.

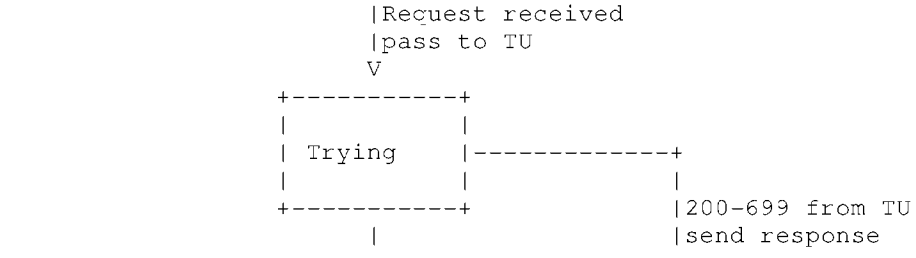
This matching rule applies to both INVITE and non-INVITE transactions alike.

The sent-by value is used as part of the matching process because there could be accidental or malicious duplication of branch parameters from different clients.

If the branch parameter in the top Via header field is not present, or does not contain the magic cookie, the following procedures are used. These exist to handle backwards compatibility with RFC 2543 compliant implementations.

The INVITE request matches a transaction if the Request-URI, To tag, From tag, Call-ID, CSeq, and top Via header field match those of the INVITE request which created the transaction. In this case, the INVITE is a retransmission of the original one that created the transaction. The ACK request matches a transaction if the Request-URI, From tag, Call-ID, CSeq number (not the method), and top Via header field match those of the INVITE request which created the transaction, and the To tag of the ACK matches the To tag of the response sent by the server transaction. Matching is done based on the matching rules defined for each of those header fields. Inclusion of the tag in the To header field in the ACK matching process helps disambiguate ACK for 2xx from ACK for other responses

at a proxy, which may have forwarded both responses (This can occur in unusual conditions. Specifically, when a proxy forked a request, and then crashes, the responses may be delivered to another proxy, which might end up forwarding multiple responses upstream). An ACK request that matches an INVITE transaction matched by a previous ACK is considered a retransmission of that previous ACK.



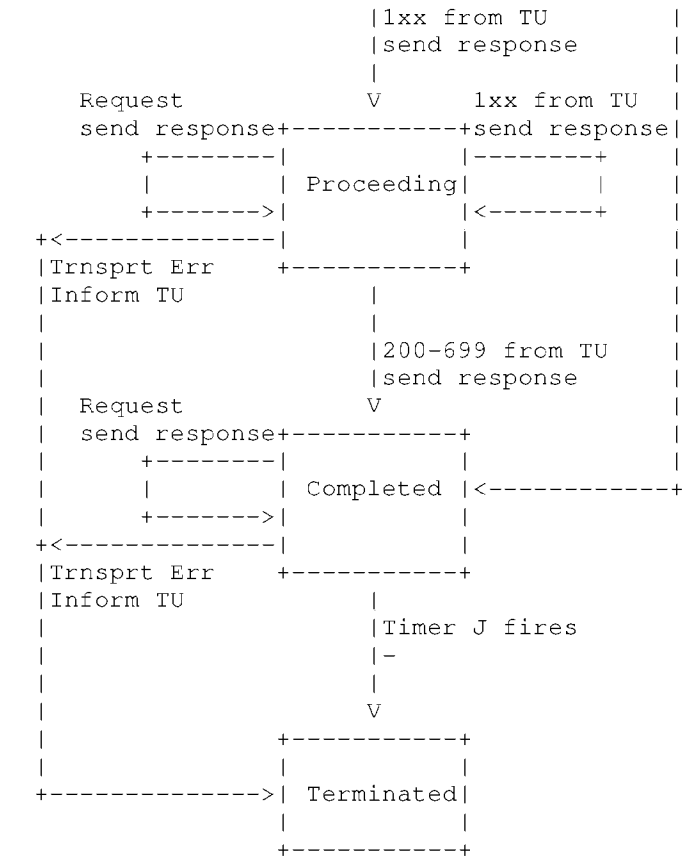


Figure 8: non-INVITE server transaction

For all other request methods, a request is matched to a transaction if the Request-URI, To tag, From tag, Call-ID, CSeq (including the method), and top Via header field match those of the request that created the transaction. Matching is done based on the matching

rules defined for each of those header fields. When a non-INVITE request matches an existing transaction, it is a retransmission of the request that created that transaction.

Because the matching rules include the Request-URI, the server cannot match a response to a transaction. When the TU passes a response to the server transaction, it must pass it to the specific server transaction for which the response is targeted.

17.2.4 Handling Transport Errors

When the server transaction sends a response to the transport layer to be sent, the following procedures are followed if the transport layer indicates a failure.

First, the procedures in [4] are followed, which attempt to deliver the response to a backup. If those should all fail, based on the definition of failure in [4], the server transaction SHOULD inform

the TU that a failure has occurred, and SHOULD transition to the terminated state.

18 Transport

The transport layer is responsible for the actual transmission of requests and responses over network transports. This includes determination of the connection to use for a request or response in the case of connection-oriented transports.

The transport layer is responsible for managing persistent connections for transport protocols like TCP and SCTP, or TLS over those, including ones opened to the transport layer. This includes connections opened by the client or server transports, so that connections are shared between client and server transport functions. These connections are indexed by the tuple formed from the address, port, and transport protocol at the far end of the connection. When a connection is opened by the transport layer, this index is set to the destination IP, port and transport. When the connection is accepted by the transport layer, this index is set to the source IP address, port number, and transport. Note that, because the source port is often ephemeral, but it cannot be known whether it is ephemeral or selected through procedures in [4], connections accepted by the transport layer will frequently not be reused. The result is that two proxies in a "peering" relationship using a connection-oriented transport frequently will have two connections in use, one for transactions initiated in each direction.

It is RECOMMENDED that connections be kept open for some implementation-defined duration after the last message was sent or received over that connection. This duration SHOULD at least equal the longest amount of time the element would need in order to bring a transaction from instantiation to the terminated state. This is to make it likely that transactions are completed over the same connection on which they are initiated (for example, request, response, and in the case of INVITE, ACK for non-2xx responses). This usually means at least 64*T1 (see Section 17.1.1.1 for a definition of T1). However, it could be larger in an element that has a TU using a large value for timer C (bullet 11 of Section 16.6), for example.

All SIP elements MUST implement UDP and TCP. SIP elements MAY implement other protocols.

> Making TCP mandatory for the UA is a substantial change from RFC 2543. It has arisen out of the need to handle larger messages, which MUST use TCP, as discussed below. Thus, even if an element never sends large messages, it may receive one and needs to be able to handle them.

18.1 Clients

18.1.1 Sending Requests

The client side of the transport layer is responsible for sending the

request and receiving responses. The user of the transport layer passes the client transport the request, an IP address, port, transport, and possibly TTL for multicast destinations.

If a request is within 200 bytes of the path MTU, or if it is larger than 1300 bytes and the path MTU is unknown, the request MUST be sent using an RFC 2914 [43] congestion controlled transport protocol, such as TCP. If this causes a change in the transport protocol from the one indicated in the top Via, the value in the top Via MUST be changed. This prevents fragmentation of messages over UDP and provides congestion control for larger messages. However, implementations MUST be able to handle messages up to the maximum datagram packet size. For UDP, this size is 65,535 bytes, including IP and UDP headers.

> The 200 byte "buffer" between the message size and the MTU accommodates the fact that the response in SIP can be larger than the request. This happens due to the addition of Record-Route header field values to the responses to INVITE, for example. With the extra buffer, the response can be about 170 bytes larger than the request, and still not be fragmented on IPv4 (about 30 bytes

> is consumed by IP/UDP, assuming no IPSec). 1300 is chosen when path MTU is not known, based on the assumption of a 1500 byte Ethernet MTU.

If an element sends a request over TCP because of these message size constraints, and that request would have otherwise been sent over UDP, if the attempt to establish the connection generates either an ICMP Protocol Not Supported, or results in a TCP reset, the element SHOULD retry the request, using UDP. This is only to provide backwards compatibility with RFC 2543 compliant implementations that do not support TCP. It is anticipated that this behavior will be deprecated in a future revision of this specification.

A client that sends a request to a multicast address MUST add the "maddr" parameter to its Via header field value containing the destination multicast address, and for IPv4, SHOULD add the "ttl" parameter with a value of 1. Usage of IPv6 multicast is not defined in this specification, and will be a subject of future standardization when the need arises.

These rules result in a purposeful limitation of multicast in SIP. Its primary function is to provide a "single-hop-discovery-like" service, delivering a request to a group of homogeneous servers, where it is only required to process the response from any one of them. This functionality is most useful for registrations. In fact, based on the transaction processing rules in Section 17.1.3, the client transaction will accept the first response, and view any others as retransmissions because they all contain the same Via branch identifier.

Before a request is sent, the client transport MUST insert a value of the "sent-by" field into the Via header field. This field contains an IP address or host name, and port. The usage of an FQDN is RECOMMENDED. This field is used for sending responses under certain conditions, described below. If the port is absent, the default value depends on the transport. It is 5060 for UDP, TCP and SCTP,

5061 for TLS.

For reliable transports, the response is normally sent on the connection on which the request was received. Therefore, the client transport MUST be prepared to receive the response on the same connection used to send the request. Under error conditions, the server may attempt to open a new connection to send the response. To handle this case, the transport layer MUST also be prepared to receive an incoming connection on the source IP address from which the request was sent and port number in the "sent-by" field. It also

MUST be prepared to receive incoming connections on any address and port that would be selected by a server based on the procedures described in Section 5 of [4].

For unreliable unicast transports, the client transport MUST be prepared to receive responses on the source IP address from which the request is sent (as responses are sent back to the source address) and the port number in the "sent-by" field. Furthermore, as with reliable transports, in certain cases the response will be sent elsewhere. The client MUST be prepared to receive responses on any address and port that would be selected by a server based on the procedures described in Section 5 of [4].

For multicast, the client transport MUST be prepared to receive responses on the same multicast group and port to which the request is sent (that is, it needs to be a member of the multicast group it sent the request to.)

If a request is destined to an IP address, port, and transport to which an existing connection is open, it is RECOMMENDED that this connection be used to send the request, but another connection MAY be opened and used.

If a request is sent using multicast, it is sent to the group address, port, and TTL provided by the transport user. If a request is sent using unicast unreliable transports, it is sent to the IP address and port provided by the transport user.

18.1.2 Receiving Responses

When a response is received, the client transport examines the top Via header field value. If the value of the "sent-by" parameter in that header field value does not correspond to a value that the client transport is configured to insert into requests, the response MUST be silently discarded.

If there are any client transactions in existence, the client transport uses the matching procedures of Section 17.1.3 to attempt to match the response to an existing transaction. If there is a match, the response MUST be passed to that transaction. Otherwise, the response MUST be passed to the core (whether it be stateless proxy, stateful proxy, or UA) for further processing. Handling of these "stray" responses is dependent on the core (a proxy will forward them, while a UA will discard, for example).

Page 1516
T-Mobile Exhibit 1002
Part 4 of 6

18.2 Servers

18.2.1 Receiving Requests

A server SHOULD be prepared to receive requests on any IP address, port and transport combination that can be the result of a DNS lookup on a SIP or SIPS URI [4] that is handed out for the purposes of communicating with that server. In this context, "handing out" includes placing a URI in a Contact header field in a REGISTER request or a redirect response, or in a Record-Route header field in a request or response. A URI can also be "handed out" by placing it on a web page or business card. It is also RECOMMENDED that a server listen for requests on the default SIP ports (5060 for TCP and UDP, 5061 for TLS over TCP) on all public interfaces. The typical exception would be private networks, or when multiple server instances are running on the same host. For any port and interface that a server listens on for UDP, it MUST listen on that same port and interface for TCP. This is because a message may need to be sent using TCP, rather than UDP, if it is too large. As a result, the converse is not true. A server need not listen for UDP on a particular address and port just because it is listening on that same address and port for TCP. There may, of course, be other reasons why a server needs to listen for UDP on a particular address and port.

When the server transport receives a request over any transport, it MUST examine the value of the "sent-by" parameter in the top Via header field value. If the host portion of the "sent-by" parameter contains a domain name, or if it contains an IP address that differs from the packet source address, the server MUST add a "received" parameter to that Via header field value. This parameter MUST contain the source address from which the packet was received. This is to assist the server transport layer in sending the response, since it must be sent to the source IP address from which the request came.

Consider a request received by the server transport which looks like, in part:

```
INVITE sip:bob@Biloxi.com SIP/2.0
Via: SIP/2.0/UDP bobspc.biloxi.com:5060
```

The request is received with a source IP address of 192.0.2.4. Before passing the request up, the transport adds a "received" parameter, so that the request would look like, in part:

```
INVITE sip:bob@Biloxi.com SIP/2.0
Via: SIP/2.0/UDP bobspc.biloxi.com:5060;received=192.0.2.4
```

Next, the server transport attempts to match the request to a server transaction. It does so using the matching rules described in Section 17.2.3. If a matching server transaction is found, the request is passed to that transaction for processing. If no match is found, the request is passed to the core, which may decide to construct a new server transaction for that request. Note that when a UAS core sends a 2xx response to INVITE, the server transaction is destroyed. This means that when the ACK arrives, there will be no matching server transaction, and based on this rule, the ACK is passed to the UAS core, where it is processed.

18.2.2 Sending Responses

The server transport uses the value of the top Via header field in order to determine where to send a response. It MUST follow the following process:

- If the "sent-protocol" is a reliable transport protocol such as TCP or SCTP, or TLS over those, the response MUST be sent using the existing connection to the source of the original request that created the transaction, if that connection is still open. This requires the server transport to maintain an association between server transactions and transport connections. If that connection is no longer open, the server SHOULD open a connection to the IP address in the "received" parameter, if present, using the port in the "sent-by" value, or the default port for that transport, if no port is specified. If that connection attempt fails, the server SHOULD use the procedures in [4] for servers in order to determine the IP address and port to open the connection and send the response to.

- Otherwise, if the Via header field value contains a "maddr" parameter, the response MUST be forwarded to the address listed there, using the port indicated in "sent-by", or port 5060 if none is present. If the address is a multicast address, the response SHOULD be sent using the TTL indicated in the "ttl" parameter, or with a TTL of 1 if that parameter is not present.

- Otherwise (for unreliable unicast transports), if the top Via has a "received" parameter, the response MUST be sent to the address in the "received" parameter, using the port indicated in the "sent-by" value, or using port 5060 if none is specified explicitly. If this fails, for example, elicits an ICMP "port unreachable" response, the procedures of Section 5 of [4] SHOULD be used to determine where to send the response.

- Otherwise, if it is not receiver-tagged, the response MUST be sent to the address indicated by the "sent-by" value, using the procedures in Section 5 of [4].

18.3 Framing

In the case of message-oriented transports (such as UDP), if the

message has a Content-Length header field, the message body is assumed to contain that many bytes. If there are additional bytes in the transport packet beyond the end of the body, they MUST be discarded. If the transport packet ends before the end of the message body, this is considered an error. If the message is a response, it MUST be discarded. If the message is a request, the element SHOULD generate a 400 (Bad Request) response. If the message has no Content-Length header field, the message body is assumed to end at the end of the transport packet.

In the case of stream-oriented transports such as TCP, the Content-Length header field indicates the size of the body. The Content-Length header field MUST be used with stream oriented transports.

18.4 Error Handling

Error handling is independent of whether the message was a request or response.

If the transport user asks for a message to be sent over an unreliable transport, and the result is an ICMP error, the behavior depends on the type of ICMP error. Host, network, port or protocol unreachable errors, or parameter problem errors SHOULD cause the transport layer to inform the transport user of a failure in sending. Source quench and TTL exceeded ICMP errors SHOULD be ignored.

If the transport user asks for a request to be sent over a reliable transport, and the result is a connection failure, the transport layer SHOULD inform the transport user of a failure in sending.

19 Common Message Components

There are certain components of SIP messages that appear in various places within SIP messages (and sometimes, outside of them) that merit separate discussion.

19.1 SIP and SIPS Uniform Resource Indicators

A SIP or SIPS URI identifies a communications resource. Like all URIs, SIP and SIPS URIs may be placed in web pages, email messages, or printed literature. They contain sufficient information to initiate and maintain a communication session with the resource.

Examples of communications resources include the following:

- a user of an online service
- an appearance on a multi-line phone
- a mailbox on a messaging system
- a PSTN number at a gateway service

- a group (such as "sales" or "helpdesk") in an organization

A SIPS URI specifies that the resource be contacted securely. This means, in particular, that TLS is to be used between the UAC and the domain that owns the URI. From there, secure communications are used to reach the user, where the specific security mechanism depends on the policy of the domain. Any resource described by a SIP URI can be "upgraded" to a SIPS URI by just changing the scheme, if it is desired to communicate with that resource securely.

19.1.1 SIP and SIPS URI Components

The "sip:" and "sips:" schemes follow the guidelines in RFC 2396 [5]. They use a form similar to the mailto URL, allowing the specification of SIP request-header fields and the SIP message-body. This makes it possible to specify the subject, media type, or urgency of sessions initiated by using a URI on a web page or in an email message. The formal syntax for a SIP or SIPS URI is presented in Section 25. Its general form, in the case of a SIP URI, is:

```
sip:user:password@host:port;uri-parameters?headers
```

The format for a SIPS URI is the same, except that the scheme is "sips" instead of sip. These tokens, and some of the tokens in their expansions, have the following meanings:

user: The identifier of a particular resource at the host being addressed. The term "host" in this context frequently refers to a domain. The "userinfo" of a URI consists of this user field, the password field, and the @ sign following them. The userinfo part of a URI is optional and MAY be absent when the

destination host does not have a notion of users or when the host itself is the resource being identified. If the @ sign is present in a SIP or SIPS URI, the user field MUST NOT be empty.

If the host being addressed can process telephone numbers, for instance, an Internet telephony gateway, a telephone-subscriber field defined in RFC 2806 [9] MAY be used to populate the user field. There are special escaping rules for encoding telephone-subscriber fields in SIP and SIPS URIs described in Section 19.1.2.

password: A password associated with the user. While the SIP and SIPS URI syntax allows this field to be present, its use is NOT RECOMMENDED, because the passing of authentication information in clear text (such as URIs) has proven to be a security risk in almost every case where it has been used. For instance, transporting a PIN number in this field exposes the PIN.

Note that the password field is just an extension of the user portion. Implementations not wishing to give special significance to the password portion of the field MAY simply treat "user:password" as a single string.

host: The host providing the SIP resource. The host part contains either a fully-qualified domain name or numeric IPv4 or IPv6

address. Using the fully-qualified domain name form is RECOMMENDED whenever possible.

port: The port number where the request is to be sent.

URI parameters: Parameters affecting a request constructed from the URI.

URI parameters are added after the hostport component and are separated by semi-colons.

URI parameters take the form:

parameter-name "=" parameter-value

Even though an arbitrary number of URI parameters may be included in a URI, any given parameter-name MUST NOT appear more than once.

This extensible mechanism includes the transport, maddr, ttl, user, method and lr parameters.

The transport parameter determines the transport mechanism to be used for sending SIP messages, as specified in [4]. SIP can use any network transport protocol. Parameter names are defined for UDP (RFC 768 [14]), TCP (RFC 761 [15]), and SCTP (RFC 2960 [16]). For a SIPS URI, the transport parameter MUST indicate a reliable transport.

The maddr parameter indicates the server address to be contacted for this user, overriding any address derived from the host field. When an maddr parameter is present, the port and transport components of the URI apply to the address indicated in the maddr parameter value. [4] describes the proper interpretation of the transport, maddr, and hostport in order to obtain the destination address, port, and transport for sending a request.

The maddr field has been used as a simple form of loose source routing. It allows a URI to specify a proxy that must be traversed en-route to the destination. Continuing to use the maddr parameter this way is strongly discouraged (the mechanisms that enable it are deprecated). Implementations should instead use the Route mechanism described in this document, establishing a pre-existing route set if necessary (see Section 8.1.1.1). This provides a full URI to describe the node to be traversed.

The ttl parameter determines the time-to-live value of the UDP multicast packet and MUST only be used if maddr is a multicast address and the transport protocol is UDP. For example, to specify a call to alice@atlanta.com using multicast to 239.255.255.1 with a ttl of 15, the following URI would be used:

sip:alice@atlanta.com;maddr=239.255.255.1;ttl=15

The set of valid telephone-subscriber strings is a subset of valid user strings. The user URI parameter exists to distinguish telephone numbers from user names that happen to look like telephone numbers. If the user string contains a telephone number formatted as a telephone-subscriber, the user parameter value "phone" SHOULD be present. Even without this parameter, recipients of SIP and SIPS URIs MAY interpret the pre-@ part as a telephone number if local restrictions on the name space for user name allow it.

The method of the SIP request constructed from the URI can be specified with the method parameter.

The lr parameter, when present, indicates that the element responsible for this resource implements the routing mechanisms specified in this document. This parameter will be used in the URIs proxies place into Record-Route header field values, and may appear in the URIs in a pre-existing route set.

This parameter is used to achieve backwards compatibility with systems implementing the strict-routing mechanisms of RFC 2543 and the rfc2543bis drafts up to bis-05. An element preparing to send a request based on a URI not containing this parameter can assume the receiving element implements strict-routing and reformat the message to preserve the information in the Request-URI.

Since the uri-parameter mechanism is extensible, SIP elements MUST silently ignore any uri-parameters that they do not understand.

Headers: Header fields to be included in a request constructed from the URI.

Headers fields in the SIP request can be specified with the "?" mechanism within a URI. The header names and values are encoded in ampersand separated hname = hvalue pairs. The special hname "body" indicates that the associated hvalue is the message-body of the SIP request.

Table 1 summarizes the use of SIP and SIPS URI components based on the context in which the URI appears. The external column describes URIs appearing anywhere outside of a SIP message, for instance on a web page or business card. Entries marked "m" are mandatory, those marked "o" are optional, and those marked "-" are not allowed. Elements processing URIs SHOULD ignore any disallowed components if they are present. The second column indicates the default value of an optional element if it is not present. "--" indicates that the element is either not optional, or has no default value.

URIs in Contact header fields have different restrictions depending on the context in which the header field appears. One set applies to messages that establish and maintain dialogs (INVITE and its 200 (OK) response). The other applies to registration and redirection messages (REGISTER, its 200 (OK) response, and 3xx class responses to any method).

19.1.2 Character Escaping Requirements

	default	Req.-URI	To	From	reg./redir. Contact	dialog Contact/ R-R/Route	external
user	--	o	o	o	o	o	o
password	--	o	o	o	o	o	o
host	--	m	m	m	m	m	m
port	(1)	o	-	-	o	o	o
user-param	ip	o	o	o	o	o	o
method	INVITE	-	-	-	-	-	o
maddr-param	--	o	-	-	o	o	o
ttl-param	1	o	-	-	o	-	o
transp.-param	(2)	o	-	-	o	o	o
lr-param	--	o	-	-	-	o	o
other-param	--	o	o	o	o	o	o
headers	--	-	-	-	o	-	o

(1): The default port value is transport and scheme dependent. The default is 5060 for sip: using UDP, TCP, or SCTP. The default is 5061 for sip: using TLS over TCP and sips: over TCP.

(2): The default transport is scheme dependent. For sip:, it is UDP. For sips:, it is TCP.

Table 1: Use and default values of URI components for SIP header field values, Request-URI and references

SIP follows the requirements and guidelines of RFC 2396 [5] when defining the set of characters that must be escaped in a SIP URI, and uses its ""%" HEX HEX" mechanism for escaping. From RFC 2396 [5]:

> The set of characters actually reserved within any given URI component is defined by that component. In general, a character is reserved if the semantics of the URI changes if the character is replaced with its escaped US-ASCII encoding [5]. Excluded US-ASCII characters (RFC 2396 [5]), such as space and control characters and characters used as URI delimiters, also MUST be escaped. URIs MUST NOT contain unescaped space and control characters.

For each component, the set of valid BNF expansions defines exactly which characters may appear unescaped. All other characters MUST be escaped.

For example, "@" is not in the set of characters in the user component, so the user "j@s0n" must have at least the @ sign encoded, as in "j%40s0n".

Expanding the hname and hvalue tokens in Section 25 show that all URI reserved characters in header field names and values MUST be escaped.

The telephone-subscriber subset of the user component has special escaping considerations. The set of characters not reserved in the RFC 2806 [9] description of telephone-subscriber contains a number of characters in various syntax elements that need to be escaped when used in SIP URIs. Any characters occurring in a telephone-subscriber that do not appear in an expansion of the BNF for the user rule MUST be escaped.

Note that character escaping is not allowed in the host component of a SIP or SIPS URI (the % character is not valid in its expansion). This is likely to change in the future as requirements for Internationalized Domain Names are finalized. Current implementations MUST NOT attempt to improve robustness by treating received escaped characters in the host component as literally equivalent to their unescaped counterpart. The behavior required to meet the requirements of IDN may be significantly different.

19.1.3 Example SIP and SIPS URIs

```
sip:alice@atlanta.com
sip:alice:secretword@atlanta.com;transport=tcp
sips:alice@atlanta.com?subject=project%20x&priority=urgent
sip:+1-212-555-1212:1234@gateway.com;user=phone
sips:1212@gateway.com
sip:alice@192.0.2.4
sip:atlanta.com;method=REGISTER?to=alice%40atlanta.com
sip:alice;day=tuesday@atlanta.com
```

The last sample URI above has a user field value of "alice;day=tuesday". The escaping rules defined above allow a semicolon to appear unescaped in this field. For the purposes of this protocol, the field is opaque. The structure of that value is only useful to the SIP element responsible for the resource.

19.1.4 URI Comparison

Some operations in this specification require determining whether two SIP or SIPS URIs are equivalent. In this specification, registrars need to compare bindings in Contact URIs in REGISTER requests (see Section 10.3.). SIP and SIPS URIs are compared for equality according to the following rules:

- A SIP and SIPS URI are never equivalent.

- Comparison of the userinfo of SIP and SIPS URIs is case-sensitive. This includes userinfo containing passwords or formatted as telephone-subscribers. Comparison of all other components of the URI is case-insensitive unless explicitly defined otherwise.

T-Mobile Exhibit 1002
Part 4 of 6

- The ordering of parameters and header fields is not significant in comparing SIP and SIPS URIs.

- Characters other than those in the "reserved" set (see RFC 2396 [5]) are equivalent to their ""%" HEX HEX" encoding.

- An IP address that is the result of a DNS lookup of a host name does not match that host name.

- For two URIs to be equal, the user, password, host, and port components must match.

  A URI omitting the user component will not match a URI that includes one. A URI omitting the password component will not match a URI that includes one.

  A URI omitting any component with a default value will not match a URI explicitly containing that component with its default value. For instance, a URI omitting the optional port component will not match a URI explicitly declaring port 5060. The same is true for the transport-parameter, ttl-parameter, user-parameter, and method components.

  Defining sip:user@host to not be equivalent to sip:user@host:5060 is a change from RFC 2543. When deriving addresses from URIs, equivalent addresses are expected from equivalent URIs. The URI sip:user@host:5060 will always resolve to port 5060. The URI sip:user@host may resolve to other ports through the DNS SRV mechanisms detailed in [4].

- URI uri-parameter components are compared as follows:

  - Any uri-parameter appearing in both URIs must match.

  - A user, ttl, or method uri-parameter appearing in only one URI never matches, even if it contains the default value.

  - A URI that includes an maddr parameter will not match a URI that contains no maddr parameter.

  - All other uri-parameters appearing in only one URI are ignored when comparing the URIs.

- URI header components are never ignored. Any present header component MUST be present in both URIs and match for the URIs to match. The matching rules are defined for each header field in Section 20.

The URIs within each of the following sets are equivalent:

```
sip:%61lice@atlanta.com;transport=TCP
sip:alice@AtLanTa.CoM;Transport=tcp

sip:carol@chicago.com
sip:carol@chicago.com;newparam=5
sip:carol@chicago.com;security=on
```

```
sip:biloxi.com;transport=tcp;method=REGISTER?to=sip:bob%40biloxi.com
sip:biloxi.com;method=REGISTER;transport=tcp?to=sip:bob%40biloxi.com

sip:alice@atlanta.com?subject=project%20x&priority=urgent
sip:alice@atlanta.com?priority=urgent&subject=project%20x
```

The URIs within each of the following sets are not equivalent:

```
SIP:ALICE@AtLanTa.CoM;Transport=udp             (different usernames)
sip:alice@AtLanTa.CoM;Transport=UDP

sip:bob@biloxi.com                   (can resolve to different ports)
sip:bob@biloxi.com:5060

sip:bob@biloxi.com              (can resolve to different transports)
sip:bob@biloxi.com;transport=udp

sip:bob@biloxi.com     (can resolve to different port and transports)
sip:bob@biloxi.com:6000;transport=tcp

sip:carol@chicago.com                      (different header component)
sip:carol@chicago.com?Subject=next%20meeting

sip:bob@phone21.boxesbybob.com   (even though that's what
sip:bob@192.0.2.4                 phone21.boxesbybob.com resolves to)
```

Note that equality is not transitive:

- sip:carol@chicago.com and sip:carol@chicago.com;security=on are equivalent
- sip:carol@chicago.com and sip:carol@chicago.com;security=off are equivalent

- sip:carol@chicago.com;security=on and sip:carol@chicago.com;security=off are not equivalent

19.1.5 Forming Requests from a URI

An implementation needs to take care when forming requests directly from a URI. URIs from business cards, web pages, and even from sources inside the protocol such as registered contacts may contain inappropriate header fields or body parts.

An implementation MUST include any provided transport, maddr, ttl, or user parameter in the Request-URI of the formed request. If the URI contains a method parameter, its value MUST be used as the method of the request. The method parameter MUST NOT be placed in the Request-URI. Unknown URI parameters MUST be placed in the message's Request-URI.

An implementation SHOULD treat the presence of any headers or body parts in the URI as a desire to include them in the message, and choose to honor the request on a per-component basis.

An implementation SHOULD NOT honor these obviously dangerous header fields: From, Call-ID, CSeq, Via, and Record-Route.

An implementation SHOULD NOT honor any requested Route header field values in order to not be used as an unwitting agent in malicious attacks.

An implementation SHOULD NOT honor requests to include header fields that may cause it to falsely advertise its location or capabilities. These include: Accept, Accept-Encoding, Accept-Language, Allow, Contact (in its dialog usage), Organization, Supported, and User-Agent.

An implementation SHOULD verify the accuracy of any requested descriptive header fields, including: Content-Disposition, Content-Encoding, Content-Language, Content-Length, Content-Type, Date, Mime-Version, and Timestamp.

If the request formed from constructing a message from a given URI is not a valid SIP request, the URI is invalid. An implementation MUST NOT proceed with transmitting the request. It should instead pursue the course of action due an invalid URI in the context it occurs.

> The constructed request can be invalid in many ways. These include, but are not limited to, syntax error in header fields, invalid combinations of URI parameters, or an incorrect description of the message body.

Sending a request formed from a given URI may require capabilities unavailable to the implementation. The URI might indicate use of an unimplemented transport or extension, for example. An implementation SHOULD refuse to send these requests rather than modifying them to match their capabilities. An implementation MUST NOT send a request requiring an extension that it does not support.

> For example, such a request can be formed through the presence of a Require header parameter or a method URI parameter with an unknown or explicitly unsupported value.

19.1.6 Relating SIP URIs and tel URLs

When a tel URL (RFC 2806 [9]) is converted to a SIP or SIPS URI, the entire telephone-subscriber portion of the tel URL, including any parameters, is placed into the userinfo part of the SIP or SIPS URI.

Thus, tel:+358-555-1234567;postd=pp22 becomes

> sip:+358-555-1234567;postd=pp22@foo.com;user=phone

or

> sips:+358-555-1234567;postd=pp22@foo.com;user=phone

not

> sip:+358-555-1234567@foo.com;postd=pp22;user=phone

or

> sips:+358-555-1234567@foo.com;postd=pp22;user=phone

In general, equivalent "tel" URLs converted to SIP or SIPS URIs in

this fashion may not produce equivalent SIP or SIPS URIs. The userinfo of SIP and SIPS URIs are compared as a case-sensitive string. Variance in case-insensitive portions of tel URLs and reordering of tel URL parameters does not affect tel URL equivalence, but does affect the equivalence of SIP URIs formed from them.

For example,

```
tel:+358-555-1234567;postd=pp22
tel:+358-555-1234567;POSTD=PP22
```

are equivalent, while

```
sip:+358-555-1234567;postd=pp22@foo.com;user=phone
sip:+358-555-1234567;POSTD=PP22@foo.com;user=phone
```


are not.

Likewise,

```
tel:+358-555-1234567;postd=pp22;isub=1411
tel:+358-555-1234567;isub=1411;postd=pp22
```

are equivalent, while

```
sip:+358-555-1234567;postd=pp22;isub=1411@foo.com;user=phone
sip:+358-555-1234567;isub=1411;postd=pp22@foo.com;user=phone
```

are not.

To mitigate this problem, elements constructing telephone-subscriber fields to place in the userinfo part of a SIP or SIPS URI SHOULD fold any case-insensitive portion of telephone-subscriber to lower case, and order the telephone-subscriber parameters lexically by parameter name, excepting isdn-subaddress and post-dial, which occur first and in that order. (All components of a tel URL except for future-extension parameters are defined to be compared case-insensitive.)

Following this suggestion, both

```
tel:+358-555-1234567;postd=pp22
tel:+358-555-1234567;POSTD=PP22
```

become

```
sip:+358-555-1234567;postd=pp22@foo.com;user=phone
```

and both

```
tel:+358-555-1234567;tsp=a.b;phone-context=5
tel:+358-555-1234567;phone-context=5;tsp=a.b
```

become

```
sip:+358-555-1234567;phone-context=5;tsp=a.b@foo.com;user=phone
```

19.2 Option Tags

Option tags are unique identifiers used to designate new options (extensions) in SIP. These tags are used in Require (Section 20.32), Proxy-Require (Section 20.29), Supported (Section 20.37) and Unsupported (Section 20.40) header fields. Note that these options appear as parameters in those header fields in an option-tag = token form (see Section 25 for the definition of token).

Option tags are defined in standards track RFCs. This is a change from past practice, and is instituted to ensure continuing multi-vendor interoperability (see discussion in Section 20.32 and Section 20.37). An IANA registry of option tags is used to ensure easy reference.

19.3 Tags

The "tag" parameter is used in the To and From header fields of SIP messages. It serves as a general mechanism to identify a dialog, which is the combination of the Call-ID along with two tags, one from each participant in the dialog. When a UA sends a request outside of a dialog, it contains a From tag only, providing "half" of the dialog ID. The dialog is completed from the response(s), each of which contributes the second half in the To header field. The forking of SIP requests means that multiple dialogs can be established from a single request. This also explains the need for the two-sided dialog identifier; without a contribution from the recipients, the originator could not disambiguate the multiple dialogs established from a single request.

When a tag is generated by a UA for insertion into a request or response, it MUST be globally unique and cryptographically random with at least 32 bits of randomness. A property of this selection requirement is that a UA will place a different tag into the From header of an INVITE than it would place into the To header of the response to the same INVITE. This is needed in order for a UA to invite itself to a session, a common case for "hairpinning" of calls in PSTN gateways. Similarly, two INVITEs for different calls will have different From tags, and two responses for different calls will have different To tags.

Besides the requirement for global uniqueness, the algorithm for generating a tag is implementation-specific. Tags are helpful in fault tolerant systems, where a dialog is to be recovered on an alternate server after a failure. A UAS can select the tag in such a way that a backup can recognize a request as part of a dialog on the failed server, and therefore determine that it should attempt to recover the dialog and any other state associated with it.

20 Header Fields

The general syntax for header fields is covered in Section 7.3. This section lists the full set of header fields along with notes on syntax, meaning, and usage. Throughout this section, we use [HX.Y] to refer to Section X.Y of the current HTTP/1.1 specification RFC 2616 [8]. Examples of each header field are given.

Information about header fields in relation to methods and proxy processing is summarized in Tables 2 and 3.

The "where" column describes the request and response types in which the header field can be used. Values in this column are:

R: header field may only appear in requests;

r: header field may only appear in responses;

2xx, 4xx, etc.: A numerical value or range indicates response codes with which the header field can be used;

c: header field is copied from the request to the response.

An empty entry in the "where" column indicates that the header field may be present in all requests and responses.

The "proxy" column describes the operations a proxy may perform on a header field:

a: A proxy can add or concatenate the header field if not present.

m: A proxy can modify an existing header field value.

d: A proxy can delete a header field value.

r: A proxy must be able to read the header field, and thus this header field cannot be encrypted.

The next six columns relate to the presence of a header field in a method:

c: Conditional; requirements on the header field depend on the context of the message.

m: The header field is mandatory.

m*: The header field SHOULD be sent, but clients/servers need to be prepared to receive messages without that header field.

o: The header field is optional.

t: The header field SHOULD be sent, but clients/servers need to be prepared to receive messages without that header field.

If a stream-based protocol (such as TCP) is used as a transport, then the header field MUST be sent.

*: The header field is required if the message body is not empty. See Sections 20.14, 20.15 and 7.4 for details.

-: The header field is not applicable.

"Optional" means that an element MAY include the header field in a request or response, and a UA MAY ignore the header field if present in the request or response (The exception to this rule is the Require header field discussed in 20.32). A "mandatory" header field MUST be present in a request, and MUST be understood by the UAS receiving the request. A mandatory response header field MUST be present in the response, and the header field MUST be understood by the UAC processing the response. "Not applicable" means that the header field MUST NOT be present in a request. If one is placed in a request by mistake, it MUST be ignored by the UAS receiving the request. Similarly, a header field labeled "not applicable" for a response means that the UAS MUST NOT place the header field in the response, and the UAC MUST ignore the header field in the response.

A UA SHOULD ignore extension header parameters that are not understood.

A compact form of some common header field names is also defined for use when overall message size is an issue.

The Contact, From, and To header fields contain a URI. If the URI contains a comma, question mark or semicolon, 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.

20.1 Accept

The Accept header field follows the syntax defined in [H14.1]. The semantics are also identical, with the exception that if no Accept header field is present, the server SHOULD assume a default value of application/sdp.

An empty Accept header field means that no formats are acceptable.

Example:

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

Table 2: Summary of header fields, A--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

Table 3: Summary of header fields, P--Z; (1): copied with possible addition of tag

Accept: application/sdp;level=1, application/x-private, text/html

20.2 Accept-Encoding

The Accept-Encoding header field is similar to Accept, but restricts the content-codings [H3.5] that are acceptable in the response. See [H14.3]. The semantics in SIP are identical to those defined in [H14.3].

An empty Accept-Encoding header field is permissible. It is equivalent to Accept-Encoding: identity, that is, only the identity encoding, meaning no encoding, is permissible.

If no Accept-Encoding header field is present, the server SHOULD assume a default value of identity.

This differs slightly from the HTTP definition, which indicates that when not present, any encoding can be used, but the identity encoding is preferred.

Example:

Accept-Encoding: gzip

20.3 Accept-Language

The Accept-Language header field is used in requests to indicate the preferred languages for reason phrases, session descriptions, or status responses carried as message bodies in the response. If no Accept-Language header field is present, the server SHOULD assume all languages are acceptable to the client.

The Accept-Language header field follows the syntax defined in [H14.4]. The rules for ordering the languages based on the "q" parameter apply to SIP as well.

Example:

Accept-Language: da, en-gb;q=0.8, en;q=0.7

20.4 Alert-Info

When present in an INVITE request, the Alert-Info header field specifies an alternative ring tone to the UAS. When present in a 180 (Ringing) response, the Alert-Info header field specifies an alternative ringback tone to the UAC. A typical usage is for a proxy

to insert this header field to provide a distinctive ring feature.

The Alert-Info header field can introduce security risks. These risks and the ways to handle them are discussed in Section 20.9, which discusses the Call-Info header field since the risks are identical.

In addition, a user SHOULD be able to disable this feature selectively.

This helps prevent disruptions that could result from the use of this header field by untrusted elements.

Example:

```
Alert-Info: <http://www.example.com/sounds/moo.wav>
```


20.5 Allow

The Allow header field lists the set of methods supported by the UA generating the message.

All methods, including ACK and CANCEL, understood by the UA MUST be included in the list of methods in the Allow header field, when present. The absence of an Allow header field MUST NOT be interpreted to mean that the UA sending the message supports no methods. Rather, it implies that the UA is not providing any information on what methods it supports.

Supplying an Allow header field in responses to methods other than OPTIONS reduces the number of messages needed.

Example:

```
Allow: INVITE, ACK, OPTIONS, CANCEL, BYE
```

20.6 Authentication-Info

The Authentication-Info header field provides for mutual authentication with HTTP Digest. A UAS MAY include this header field in a 2xx response to a request that was successfully authenticated using digest based on the Authorization header field.

Syntax and semantics follow those specified in RFC 2617 [17].

Example:

```
Authentication-Info: nextnonce="47364c23432d2e131a5fb210812c"
```

20.7 Authorization

The Authorization header field contains authentication credentials of a UA. Section 22.2 overviews the use of the Authorization header field, and Section 22.4 describes the syntax and semantics when used with HTTP authentication.

This header field, along with Proxy-Authorization, breaks the general rules about multiple header field values. Although not a comma-separated list, this header field name may be present multiple times, and MUST NOT be combined into a single header line using the usual rules described in Section 7.3.

In the example below, there are no quotes around the Digest parameter:

```
Authorization: Digest username="Alice", realm="atlanta.com",
 nonce="84a4cc6f3082121f32b42a2187831a9e",
 response="7587245234b3434cc3412213e5f113a5432"
```

20.8 Call-ID

The Call-ID header field uniquely identifies a particular invitation or all registrations of a particular client. A single multimedia conference can give rise to several calls with different Call-IDs, for example, if a user invites a single individual several times to the same (long-running) conference. Call-IDs are case-sensitive and are simply compared byte-by-byte.

The compact form of the Call-ID header field is i.

Examples:

```
Call-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6@biloxi.com
i:f81d4fae-7dec-11d0-a765-00a0c91e6bf6@192.0.2.4
```

20.9 Call-Info

The Call-Info header field provides additional information about the caller or callee, depending on whether it is found in a request or response. The purpose of the URI is described by the "purpose" parameter. The "icon" parameter designates an image suitable as an iconic representation of the caller or callee. The "info" parameter describes the caller or callee in general, for example, through a web page. The "card" parameter provides a business card, for example, in vCard [36] or LDIF [37] formats. Additional tokens can be registered using IANA and the procedures in Section 27.

Use of the Call-Info header field can pose a security risk. If a callee fetches the URIs provided by a malicious caller, the callee may be at risk for displaying inappropriate or offensive content, dangerous or illegal content, and so on. Therefore, it is RECOMMENDED that a UA only render the information in the Call-Info header field if it can verify the authenticity of the element that originated the header field and trusts that element. This need not be the peer UA; a proxy can insert this header field into requests.

Example:

```
Call-Info: <http://wwww.example.com/alice/photo.jpg> ;purpose=icon,
  <http://www.example.com/alice/> ;purpose=info
```

20.10 Contact

A Contact header field value provides a URI whose meaning depends on the type of request or response it is in.

A Contact header field value can contain a display name, a URI with URI parameters, and header parameters.

This document defines the Contact parameters "q" and "expires". These parameters are only used when the Contact is present in a REGISTER request or response, or in a 3xx response. Additional parameters may be defined in other specifications.

When the header field value contains a display name, the URI including all URI parameters is enclosed in "<" and ">". If no "<" and ">" are present, all parameters after the URI are header parameters, not URI parameters. The display name can be tokens, or a quoted string, if a larger character set is desired.

Even if the "display-name" is empty, the "name-addr" form MUST be used if the "addr-spec" contains a comma, semicolon, or question mark. There may or may not be LWS between the display-name and the "<".

These rules for parsing a display name, URI and URI parameters, and header parameters also apply for the header fields To and From.

The Contact header field has a role similar to the Location header field in HTTP. However, the HTTP header field only allows one address, unquoted. Since URIs can contain commas and semicolons as reserved characters, they can be mistaken for header or parameter delimiters, respectively.

The compact form of the Contact header field is m (for "moved").

Examples:

```
Contact: "Mr. Watson" <sip:watson@worcester.bell-telephone.com>
   ;q=0.7; expires=3600,
   "Mr. Watson" <mailto:watson@bell-telephone.com> ;q=0.1
m: <sips:bob@192.0.2.4>;expires=60
```


20.11 Content-Disposition

The Content-Disposition header field describes how the message body or, for multipart messages, a message body part is to be interpreted by the UAC or UAS. This SIP header field extends the MIME Content-Type (RFC 2183 [18]).

Several new "disposition-types" of the Content-Disposition header are defined by SIP. The value "session" indicates that the body part describes a session, for either calls or early (pre-call) media. The value "render" indicates that the body part should be displayed or otherwise rendered to the user. Note that the value "render" is used rather than "inline" to avoid the connotation that the MIME body is displayed as a part of the rendering of the entire message (since the MIME bodies of SIP messages oftentimes are not displayed to users). For backward-compatibility, if the Content-Disposition header field is missing, the server SHOULD assume bodies of Content-Type application/sdp are the disposition "session", while other content types are "render".

The disposition type "icon" indicates that the body part contains an image suitable as an iconic representation of the caller or callee that could be rendered informationally by a user agent when a message has been received, or persistently while a dialog takes place. The value "alert" indicates that the body part contains information, such as an audio clip, that should be rendered by the user agent in an attempt to alert the user to the receipt of a request, generally a request that initiates a dialog; this alerting body could for example be rendered as a ring tone for a phone call after a 180 Ringing provisional response has been sent.

Any MIME body with a "disposition-type" that renders content to the user should only be processed when a message has been properly authenticated.

The handling parameter, handling-param, describes how the UAS should react if it receives a message body whose content type or disposition type it does not understand. The parameter has defined values of "optional" and "required". If the handling parameter is missing, the value "required" SHOULD be assumed. The handling parameter is described in RFC 3204 [19].

If this header field is missing, the MIME type determines the default content disposition. If there is none, "render" is assumed.

Example:

```
Content-Disposition: session
```


20.12 Content-Encoding

The Content-Encoding header field is used as a modifier to the "media-type". When present, its value indicates what additional content codings have been applied to the entity-body, and thus what decoding mechanisms MUST be applied in order to obtain the media-type referenced by the Content-Type header field. Content-Encoding is primarily used to allow a body to be compressed without losing the identity of its underlying media type.

If multiple encodings have been applied to an entity-body, the content codings MUST be listed in the order in which they were applied.

All content-coding values are case-insensitive. IANA acts as a registry for content-coding value tokens. See [H3.5] for a definition of the syntax for content-coding.

Clients MAY apply content encodings to the body in requests. A server MAY apply content encodings to the bodies in responses. The server MUST only use encodings listed in the Accept-Encoding header field in the request.

The compact form of the Content-Encoding header field is e. Examples:

```
Content-Encoding: gzip
e: tar
```

20.13 Content-Language

See [H14.12]. Example:

```
Content-Language: fr
```

20.14 Content-Length

The Content-Length header field indicates the size of the message-body, in decimal number of octets, sent to the recipient. Applications SHOULD use this field to indicate the size of the message-body to be transferred, regardless of the media type of the entity. If a stream-based protocol (such as TCP) is used as transport, the header field MUST be used.

The size of the message-body does not include the CRLF separating header fields and body. Any Content-Length greater than or equal to zero is a valid value. If no body is present in a message, then the Content-Length header field value MUST be set to zero.

The ability to omit Content-Length simplifies the creation of cgi-like scripts that dynamically generate responses.

The compact form of the header field is l.

Examples:

```
Content-Length: 349
l: 173
```

20.15 Content-Type

The Content-Type header field indicates the media type of the message-body sent to the recipient. The "media-type" element is defined in [H3.7]. The Content-Type header field MUST be present if the body is not empty. If the body is empty, and a Content-Type header field is present, it indicates that the body of the specific type has zero length (for example, an empty audio file).

The compact form of the header field is c.

Examples:

```
Content-Type: application/sdp
c: text/html; charset=ISO-8859-4
```

20.16 CSeq

A CSeq header field in a request contains a single decimal sequence number and the request method. The sequence number MUST be expressible as a 32-bit unsigned integer. The method part of CSeq is case-sensitive. The CSeq header field serves to order transactions within a dialog, to provide a means to uniquely identify transactions, and to differentiate between new requests and request retransmissions. Two CSeq header fields are considered equal if the sequence number and the request method are identical. Example:

```
CSeq: 4711 INVITE
```

20.17 Date

The Date header field contains the date and time. Unlike HTTP/1.1, SIP only supports the most recent RFC 1123 [20] format for dates. As in [H3.3], SIP restricts the time zone in SIP-date to "GMT", while RFC 1123 allows any time zone. An RFC 1123 date is case-sensitive.

The Date header field reflects the time when the request or response is first sent.

The Date header field can be used by simple end systems without a battery-backed clock to acquire a notion of current time. However, in its GMT form, it requires clients to know their offset from GMT.

Example:

```
Date: Sat, 13 Nov 2010 23:29:00 GMT
```

20.18 Error-Info

The Error-Info header field provides a pointer to additional information about the error status response.

SIP UACs have user interface capabilities ranging from pop-up windows and audio on PC softclients to audio-only on "black" phones or endpoints connected via gateways. Rather than forcing a server generating an error to choose between sending an error status code with a detailed reason phrase and playing an audio recording, the Error-Info header field allows both to be sent. The UAC then has the choice of which error indicator to render to the caller.

A UAC MAY treat a SIP or SIPS URI in an Error-Info header field as if it were a Contact in a redirect and generate a new INVITE, resulting in a recorded announcement session being established. A non-SIP URI MAY be rendered to the user.

Examples:

```
SIP/2.0 404 The number you have dialed is not in service
Error-Info: <sip:not-in-service-recording@atlanta.com>
```

20.19 Expires

The Expires header field gives the relative time after which the message (or content) expires.

The precise meaning of this is method dependent.

The expiration time in an INVITE does not affect the duration of the actual session that may result from the invitation. Session description protocols may offer the ability to express time limits on the session duration, however.

The value of this field is an integral number of seconds (in decimal) between 0 and (2**32)-1, measured from the receipt of the request.

Example:

```
Expires: 5
```

20.20 From

The From header field indicates the initiator of the request. This may be different from the initiator of the dialog. Requests sent by the callee to the caller use the callee's address in the From header field.

The optional "display-name" is meant to be rendered by a human user interface. A system SHOULD use the display name "Anonymous" if the identity of the client is to remain hidden. Even if the "display-name" is empty, the "name-addr" form MUST be used if the "addr-spec" contains a comma, question mark, or semicolon. Syntax issues are discussed in Section 7.3.1.

Two From header fields are equivalent if their URIs match, and their parameters match. Extension parameters in one header field, not present in the other are ignored for the purposes of comparison. This means that the display name and presence or absence of angle brackets do not affect matching.

See Section 20.10 for the rules for parsing a display name, URI and URI parameters, and header field parameters.

The compact form of the From header field is f.

Examples:

```
From: "A. G. Bell" <sip:agb@bell-telephone.com> ;tag=a48s
From: sip:+12125551212@server.phone2net.com;tag=887s
f: Anonymous <sip:c8oqz84zk7z@privacy.org>;tag=hyh8
```

20.21 In-Reply-To