INTERNET DRAFT
Category: Informational
Title: draft-ietf-aaa-solutions-01.txt
Date: November 2000

Jari Arkko
Oy LM Ericsson Ab
Pat R. Calhoun
Erik Guttman
Sun Microsystems, Inc.
Dave Nelson
Enterasys Networks, Inc.
Barney Wolff
Databus Inc.

AAA Solutions

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at:

http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at:

http://www.ietf.org/shadow.html.

This document is an individual contribution for consideration by the AAA Working Group of the Internet Engineering Task Force. Comments should be submitted to the diameter@diameter.org and aaa-wg@merit.edu mailing lists.

Distribution of this memo is unlimited.

Abstract

The AAA Design Team has issued a document that lists issues with the DIAMETER protocol. This accompanying document is intended as an archive of proposed solutions. Once accepted, these solutions will find their way into the relevant DIAMETER specifications.

Table of Contents

1.0 Introduction

The AAA Design Team has issued a document that lists issues with the DIAMETER protocol [11]. This accompanying document is intended as an archive of proposed solutions. Once accepted, these solutions will find their way into the relevant DIAMETER specifications.

Each main section should specify which DIAMETER documents its sub-sections are targetted at. Ideally, the section should also state whether the proposed text is intended to replace existing text, or added as new text.

2.0 Error Codes and Messages

Section 2.1 below is intended to replace the current section 5.2 in the Base Protocol [1] specification. Section 2.2 is intended as a new section 5.3 in the Base Protocol [1].

The existing Base Protocol defines the Result-Code AVP to be of type Complex, while the following section defines the Result-Code AVP to be of type Integer, and a separate Error-Message AVP of type String.

2.1 Result-Code AVP

The Result-Code AVP (AVP Code 268) is of type Integer and indicates whether a particular request was completed successfully or whether an error occurred. All DIAMETER messages of type *-Response or *-Answer MUST include one Result-Code AVP.

The Result Code field contains an IANA-managed 32-bit address space representing errors (see section 8.4). The DIAMETER provides five different classes of errors, all identified by the thousands digit:

- 1xxx (Informational)
- 2xxx (Success)
- 3xxx (Redirect Notification)
- 4xxx (Transient Failures)
- 5xxx (Permanent Failure)

2.1.1 Informational

Errors that fall within the Informational category are used to inform a requestor that the request cannot be immediately satisfied and a further response will be issued in the near future.

```
DIAMETER_BE_PATIENT                 1001
```

The DIAMETER server responsible for authentication and/or authorizing the user cannot satisfy the request at the moment, and will respond within the next 3 seconds.

2.1.2 Success

Errors that fall within the Success category are used to inform a peer that a request has been successfully completed.

DIAMETER_SUCCESS 2000
The Request was successfully completed.

2.1.3 Redirect Notification

Errors that fall within the Redirect Notification category are used to inform a peer that the request cannot be satisfied locally and should instead be forwarded to another server.

DIAMETER_REDIRECT_INDICATION 3001
A proxy or redirect server has determined that the request could not be satisfied locally and the initiator of the request should direct the request directly to the server, whose contact information has been added to the response. This error code MUST NOT be sent in a Message-Reject-Ind message.

2.1.4 Transient Failures

Errors that fall within the transient failures category are used to inform a peer that the request could not be satified at the time it was received, but MAY be able to satisfy the request in the future.

DIAMETER_TIME_INVALID 4001
This Result-Code value is return to inform a peer that the message received contained an invalid timestamp value (in Timestamp AVP).

DIAMETER_AUTHENTICATION_REJECTED 4002
The authentication process for the user failed, most likely due to an invalid password used by the user. Further attempts MUST only be tried after prompting the user for a new password.

DIAMETER_AUTHORIZATION_FAILED 4003
A request was received for which the user could not be authorized at this time. This error could occur when the user has already expended allowed resources, or is only permitted to

access services within a time period.

DIAMETER_UNABLE_TO_DELIVER 4004
The request could not be delivered to a host that handles the realm requested at this time.

DIAMETER_NO_END_2_END_SECURITY 4005
A proxy has detected that end-to-end security has been applied to portions of the DIAMETER message, and the proxy does not allow this security mode since it needs to alter the message by applying some local policies.

DIAMETER_CONTRADICTING_AVPS 4006
The Home DIAMETER server has detected AVPs in the request that contradicted each other, and is not willing to provide service to the user. One or more Failed-AVP MUST be present, containing the AVPs that contradicted each other.

2.1.5 Permanent Failures

Errors that fall within the permanent failures category are used to inform the peer that the request failed, and should not be attempted again.

DIAMETER_USER_UNKNOWN 5001
A request was received for a user that is unknown, therefore authentication and/or authorization failed.

DIAMETER_COMMAND_UNSUPPORTED 5002
The Request contained a Command-Code that the receiver did not recognize or support. The Message-Reject-Ind message MUST also contain a Unknown-Command-Code AVP containing the unrecognized Command-Code.

DIAMETER_AVP_UNSUPPORTED 5003
The peer received a message that contained an AVP that is not recognized or supported and was marked with the Mandatory bit. A DIAMETER message with this error MUST contain one or more Failed-AVP AVP containing the AVPs that caused the failure.

DIAMETER_REALM_NOT_SERVED 5004
A proxy or redirect server has determined that it is unable to forward the request or provide redirect information since the realm portion of the NAI requested is unknown.

DIAMETER_UNSUPPORTED_TRANSFORM 5005
A message was received that included an Integrity-Check-Value

or CMS-Data AVP [11] that made use of an unsupported transform.

DIAMETER_UNKNOWN_SESSION_ID 5006
The request or response contained an unknown Session-Id.

DIAMETER_AUTHORIZATION_REJECTED 5007
A request was received for which the user could not be authorized. This error could occur if the service requested is not permitted to the user.

DIAMETER_INVALID_AVP_VALUE 5008
The request contained an AVP with an invalid value in its data portion. A DIAMETER message with this result code MUST include the offending AVPs within a Failed-AVP AVP.

DIAMETER_MISSING_AVP 5009
The request did not contain an AVP that is considered mandatory by the Command Code definition. If this value is sent in the Result-Code AVP, a Failed-AVP AVP SHOULD be included in the message. The data portion of the Failed-AVP MUST have its AVP Code set to the value of the missing AVP.

DIAMETER_INVALID_AUTH 5010
The Request did not contain a valid Integrity-Check-Value or CMS-Data [11] AVP.

DIAMETER_LOOP_DETECTED 5011
A Proxy or Redirect server detected a loop while trying to get the message to the Home DIAMETER server. Further attempts should not be attempted until the loop has been fixed.

2.2 Error-Message AVP

The Error-Message AVP (AVP Code 281) is of type String and MAY be present if the message also contains a non-successful Result-Code AVP. The AVP MUST contain a human readable error message. The Error-Message AVP is not intended to be useful in real-time, and SHOULD NOT be expected to be parsed by network entities.

3.0 Accounting

This section contains the solutions for the Accounting related issues, described in [11]. The following issues do not yet have any proposed solutions:
- Where to place ADIF applicability statements
- Whether to strongly secure at all

- Multi-party trust, counter signatures etc in a broker proxy environment.
- Accounting-State/Accounting-Status-Ind issues from section 3.4
- Issues with polling

3.1 Universal Approach

This section describes a new structure of the accounting specification [6] in order to clarify what DIAMETER nodes need to support in different environments. If accepted, the accounting specification is split to several documents

A two-layer approach is proposed. On the base protocol layer, the basic accounting messages and AVPs are defined. On the application extension layer, the specific applications define their specific requirements on what data is included in the accounting records, when the records are produced, and what specific parts of the base accounting protocol must be used in the particular application. For instance, the Mobile IP DIAMETER extension [4] could specify its requirements on what specific attribute values are required within the ADIF records, which events should produce records, and whether strong security is required. The application extension layer should be documented in the corresponding application extension document, such as the DIAMETER Mobile IP extension.

The base accounting protocol layer is documented in the same manner as the current draft [6] stands. However, it is proposed that the parts which are mandatory and optional are clearly marked. Right now everything in the base accounting protocol as mandatory, leading to unnecessary complexity for applications that do not require all the baggage. The purpose of specifying optional parts is to provide a simple accounting base protocol that can be easily implemented, while also supporting more complex applications in optional parts.

Furthermore, batch accounting functionality is removed from the accounting protocol. Of course, operational experience and/or new requirements may later lead to the introduction of a DIAMETER batch accounting extension as well.

The following new structure for the DIAMETER accounting protocol is therefore proposed.

3.1.1 Base Accounting Protocol Layer

The current accounting protocol specification [6] is logically organized to the following parts, each with a different extension

number:

- Accounting-Base (Mandatory)

  This is the basic operations: Accounting-Request, Accounting-Answer, Accounting-Status-Ind as well as the AVPs Record-Type, ADIF-Record, Record-Number, State, and Interim-Interval would be contained here.

  The messages specified in [6] are modified to no longer require strong security, i.e. the CMS-Data AVPs in the messages are made optional.

  Support for this protocol provides real-time accounting support of single record/message, as well as Interim accounting. It is extremely simple to implement.

  The elimination of several records makes many things easier e.g. splitting of Accounting-Answers through proxies is no longer a problem.

- Polling (Optional)

  This contains Accounting-Poll-Ind and related functionality. This extension can only be used between consenting servers in a non-roaming situation due to the scalability problems involved in a global polling scenario.

3.1.2 Application Specific Accounting Layer

All DIAMETER extensions MUST have an Accounting Considerations section. The following information MUST be present in the section:

- Whether accounting is required by the application
- Which application specific events cause the production of an accounting record
- What data, and in which format, is included in the accounting records.
- Applicability statement regarding how ADIF is used for this particular application
- Which parts of the base accounting protocol layer are required and optional in this particular application

[Note that the application specific accounting can not have an own extension number since otherwise the accounting messages could not be transmitted through proxies not having special support for this particular kind of application.]

3.2 Batch Accounting

Batch accounting is removed from the protocol. The reasons for this recommendation are as follows:

1. For low granularity of batching, i.e. on the order of a second or two, the underlying network transport layer may provide sufficient batching properties, via "nageling" such that small, inefficient packet sizes are avoided.

2. For high granularity of batching, i.e. many minutes or hours, FTP may be a more appropriate protocol for the transfer of accounting data.

3. Proxy operations, in which the DIAMETER client sends accounting data to the first hop proxy, for a co-mingled collection of ultimate destinations, i.e. home accounting servers, is problematic. If the proxy server has forwarded accounting records to multiple destinations, based in NAI, and one or more of those accounting servers is not responding, the proxy server has no reasonable way to inform the DIAMETER client that only part of the accounting data has been acknowledged. This potentially creates a head of line blocking problem. The proxy's Accounting-Answer will need to be delayed until an Accounting-Answer is received for *all* of the records in the batch. This in turn will require more NAS non-volatile storage, at exactly the time when that storage is likely to be filling up. Allowing for only "real time" accounting, in which there are no co-mingled destinations, solves this problem.

4. The added complexity of batch accounting seems to outweigh its possible benefits.

3.3 Proxy Accounting Issues

The complete removal of batch accounting functionality removes the problems in dealing with partial Accounting-Answer messages.

3.4. Semantics Issues

A new transient error code is proposed in order to optimize retry behaviour in an out-of-disk-space situation:

DIAMETER_OUT_OF_SPACE   4007
A DIAMETER node received the accounting request but was unable to commit it to stable storage due to a temporary lack of space.

3.5 Bloat Issues

Given that section 3.1. above proposed that CMS-Data AVPs be made optional, it is further proposed that the ADIF-Record AVP is only included in Accounting-Answer if the CMS-Data AVP is. This would remove the bloat in environments that do not require strong security.

Therefore, it is proposed that the Accounting-Answer command be changed as follows:

```
<Accounting-Answer> ::= <DIAMETER Header, Command-Code = 272>
                        <Result-Code AVP>
                        <Host-Name AVP>
                        <Destination-NAI AVP>
                        <Grouped-AVP {
                             <Session-Id AVP> &&
                             <Accounting-Record-Number> &&
                             [<ADIF-Record AVP> &&
                             <CMS-Data AVP>]
                        }
                        [<Timestamp AVP>
                         <Nonce AVP>
                         <Integrity-Check-Value AVP>]
```

The AVPs ADIF-Record and CMS-Data MUST be present if and only if CMS-Data AVP was present in the corresponding Accounting-Request command. Therefore, if the CMS-Data AVP was not present in the request, none of the accounting data from the record would be copied to the answer. Only the Session-Id and Accounting-Record-Number AVPs would be returned in order to correlate the answer to the request.

Typically, a service provider's DIAMETER proxy would add CMS-Data AVPs to accounting requests where the business relationships call for strong security and/or non-repudiation of accounting data.

3.6 Format Issues

Given that section 3.1. above requires accounting considerations specification for each application, and requires an ADIF applicability statement in that specificaion.

It is proposed that ADIF continues as the only DIAMETER accounting record format to maximize interoperability. However, the new structure of the accounting specification allows the IETF to later revisit this decision if it proves necessary to provide other formats for special applications.

4.0 IPv6 Support

The AAA Issues [11] document described an issue where the IPv6 attributes defined in the RADIUS protocol [21] MUST be supported by the DIAMETER protocol. Each RADIUS attribute is listed in this section and the recommended DIAMETER protocol change to support this functionality. When a change is recommended to the protocol the section will contain the actual text to be included in [2].

4.1 NAS-IPv6-Address

The DIAMETER base protocol [1] defines the Host-IP-Address AVP to be of type Address, which can contain both IPv4 and IPv6 addresses. A protocol gateway server will have to identify the address type in the Host-IP-Address AVP and insert the value in the RADIUS attribute that corresponds to the IP version number.

4.2 Login-IPv6-Host

The DIAMETER NASREQ extension [2] defines the Login-IP-Host AVP to be of type Address, which can contain both IPv5 and IPv6 addresses. A protocol gateway server will have to identify the address type in the Login-IP-Host AVP and insert the value in the RADIUS attribute that corresponds to the IP version number.

4.3 Framed-Interface-Id

The Framed-Interface-Id AVP (AVP Code TBD) is of type Integer64 and contains the IPv6 interface identifier to be configured for the user.

4.4 Framed-IPv6-Prefix

The Framed-IPv6-Prefix AVP (AVP Code TBD) is of type Address and contains the IPv6 prefix to be configured for the user.

4.5 Framed-IPv6-Route

The Framed-IPv6-Route AVP (AVP Code TBD) is of type String and contains the routing information to be configured for the user on the NAS. Zero or more such AVPs MAY be present in an authorization response.

For IPv6 routes, it SHOULD contain a destination prefix optionally

followed by a slash and a decimal length specifier stating how many high order bits of the prefix to use. That is followed by a space, a gateway address, a space, and one or more metrics separated by spaces.

Whenever the gateway address is specified as zero the IP address of the user SHOULD be used as the gateway address.

5.0 Transports

With the exception of section 5.1, DIAMETER transport is For Further Study (FFS). However, the work items that have been identified by the Design Team are:

1. Is TCP appropriate as a MAY?
2. What are the Proxy behavior requirements for congestion control under SCTP?
3. Is UDP a valid transport mapping?

5.1 Failover & Recovery Sending

When DIAMETER is run over a connection-oriented transport layer that reacts sufficiently quickly to endpoint failure, a DIAMETER peer MAY rely on a failure indication from the transport. If not, the DIAMETER peer SHOULD implement its own algorithm to determine peer failure.

In either case, if the DIAMETER implementation originates requests, and has a backup peer configured or can discover one, it SHOULD send new requests to the backup peer. During this time, it SHOULD monitor the primary peer for possible recovery. When DIAMETER is run over a connection-oriented transport, the originator of the failed connection SHOULD periodically attempt to re-establish the transport connection. When DIAMETER is run over a connectionless transport, the only way to determine peer recovery is to send a dummy request. [[Such a NOP request needs to be defined.]] A typical interval for attempts to discover primary peer recovery might be 60 seconds, but a longer randomized interval is advisable where the number of clients of a single server is large, to avoid overwhelming the server as it recovers.

The health of a backup peer SHOULD also be monitored, even when it is not needed to satisfy live requests. Murphy's law implies that a backup that has not been monitored will surely be found to have failed or been misconfigured when it is most needed.

6.0 Proxies

This section contains text that is intended to replace all of section 6 in the DIAMETER Base protocol [1]. This section contains clarifications on the expected behavior of DIAMETER proxies and redirect servers, and also introduces new AVPs.

6.1 Realm-Based Message Routing

DIAMETER Message routing is done through the use of the realm portion of the Network Access Identifier (NAI), and an associated realm routing table (see section 10.0). The NAI has a format of user@realm, and DIAMETER servers have a list of locally supported realms, and MAY have a list of externally supported realms. When a message is received that includes a realm that is not locally supported, the message is proxied to the DIAMETER entity configured in the "route" table.

There are instances where the User-Name AVP is not present in authorization requests. This is typically true in networks where a request is sent to the network before the call was even answered. However, such requests MAY need to be proxied. In such cases, the first hop DIAMETER proxy MUST append the Home-Realm AVP to the DIAMETER message, by using a DNIS or ANI to Home-Realm association table.

Figure 1 depicts an example where DIA1 receives a request to authenticate user "joe@abc.com". DIA1 looks up "abc.com" in its local realm route table and determines that the message must be proxied to DIA2. DIA2 does the same check, and proxies the message to DIA3. DIA3 checks its realm route table, and determines that the realm is locally supported, and processes the authentication request, and returns the response. How the response actually makes it back to the sender of the original request is described in the next section.

```
                (Request)                   (Request)
         (User-Name=joe@abc.com)    (User-Name=joe@abc.com)
   +-------+      ------>      +-------+      ------>      +-------+
   |       |                   |       |                   |       |
   | DIA1 +-------------------+ DIA2 +-------------------+ DIA3 |
   |       |                   |       |                   |       |
   +-------+      <------      +-------+      <------      +-------+
                (Response)                  (Response)
         (User-Name=joe@abc.com)    (User-Name=joe@abc.com)

   mno.net                     xyz.com                     abc.com
```

Figure 1: Realm-Based Routing

6.2 Behavior of Proxy and Redirect Servers

This section describes the behavior of DIAMETER proxy and redirect servers in detail. In both cases, determining the next hop for a DIAMETER message is done via the Home-Realm or the User-Name AVP [1], whose syntax must comply with the Network Access Identifier (NAI) [12] specification. When present, the Home-Realm takes precedence over the User-Name AVP for routing decisions. The Home-Realm AVP, or the realm portion of the User-Name AVP is used to identify the next hop server the message must be forwarded to.

Note the processing rules contained in this section are intended to be used as general guidelines to DIAMETER developers. Certain implementations MAY use different methods than the ones described here, and still be in compliance with the protocol specification.

6.2.1 Proxy and Redirect Server handling of requests

Any request received by a DIAMETER server MUST perform a next hop lookup. Lookups are performed against what is commonly known as the Domain Routing Table (see section 10.0). A Domain Routing Table Entry contains the following fields:

- Domain Name. The Domain Name is analogous to the realm portion of the NAI. This is the field that is typically used as a primary key in the routing table lookups. Note that some implementations perform their lookups based on longest-match-from-the-right on the realm rather than requiring an exact match.
- Extension Id. It is possible for a routing entry to have a different destination based on the extension identifier of the message. This field is typically used as a secondary key field in routing table lookups.
- Local Action. The Local Action field is used to identify how a message should be treated. The following actions are supported:
    1. LOCAL - DIAMETER messages that resolve to a routing entry with the Local Action set to Local can be satisfied locally, and do not need to be forwarded to another server.
    2. PROXY - All DIAMETER messages that fall within this category MUST be forwarded to a next hop server. The local server MAY apply its local policies to the message by including new AVPs to the message prior to forwarding. See section 6.4 for more information.
    3. REDIRECT - DIAMETER messages that fall within this category MUST have the identity of the home DIAMETER server(s) appended, and returned to the sender of the message. See section 6.3 for more information.

    4. OTHER - If anyone has any ideas, please let me know what
       an "other" really is.

- Server Identifier - One or more servers the message is to be forwarded to. When the Local Action is set to PROXY, this field contains the identities of the server the message must be forwarded to. When the Local Action field is set to REDIRECT, this field contains the Home DIAMETER server(s) for the realm.

It is important to note that DIAMETER servers MUST support at least one of the PROXY, REDIRECT, or LOCAL modes of operation. Servers do not need to support all modes of operation in order to conform with the protocol specification. Server MUST NOT reorder AVPs with the same AVP Code.

6.3 Redirect Server

A Redirect Server is one that provides NAI Realm to DIAMETER Home Server address resolution. When a message is received by a peer, the Home-Realm or the realm portion of the User-Name AVP is extracted from the message, and the realm portion is used to perform a lookup in the domain routing table. Implementations SHOULD also use the Extension Id as a secondary key in the domain routing table lookup.

Successful routing table lookups will return one or more home DIAMETER server that could satisfy the message. The home servers are encoded in one or more Redirected-Host, and optional Redirect-Host-Port AVPs [1]. In the event that more than one such home server is returned, each AVP pair MUST be encapsulated within a Grouped-AVP.

```
                          +------------------+
                          |     DIAMETER     |
                          | Redirect Server  |
                          +------------------+
                           ^    |
                   Request |    | Response +
               joe@xyz.com |    | Result Code =
                           |    | Redirect
                           |    v
                         +----------+   Request    +----------+
                         | abc.net  |------------->| xyz.net  |
                         | DIAMETER |              | DIAMETER |
                         |  Server  |<-------------|  Server  |
                         +----------+   Response   +----------+
```

Figure 4: DIAMETER Redirect Server

Lastly, the Result-Code AVP is added with the value of the AVP set to DIAMETER_REDIRECT_INDICATION [1], and the message is returned to the

sender of the request. Redirect servers MAY also include the certificate of the Home server(s). These certificates are encapsulated in a CMS-Data AVP [11]. When this occurs, the server forwarding the request directly to the Home DIAMETER server SHOULD include its own certificate in the message.

6.3.1 Redirect-Host AVP

The Redirect-Host AVP (AVP Code 278) is of type Address and is returned in a response that has the Result-Code AVP set to DIAMETER_REDIRECT_REQUEST. This AVP includes the IP address of the DIAMETER host to which the request MUST be redirected. The presence of multiple Redirect-Host AVPs within the same Grouped-AVP, implies that all of the addresses MAY be used to contact the same host. When multiple AVPs are found that are un-grouped, or grouped with different Grouped-AVPs, they represent separate hosts. Upon receipt of such a Result-Code, and this AVP, a DIAMETER host SHOULD send the request directly to one of the hosts.

6.3.2 Redirect-Host-Port AVP

The Redirect-Host-Port AVP (AVP Code 277) is of type Integer32 and MAY be present when the Redirect-Host AVP is present. The absence of this AVP implies that the reserved port MUST be used.

6.4 Proxy Server

This section outlines the processing rules for DIAMETER proxy servers. A proxy server can either be stateful or stateless. A Proxy server MAY act in a stateful manner for some requests, and be stateless for others. There are two types of states that servers MAY wish to maintain; transaction and session.

Maintaining transaction state implies that a server keeps a copy of a request, which is then used when the corresponding response is received. This could be done to apply local policies to the message, or simply for auditing purposes. Maintaining session state implies that a server keeps track of all "active" users. An active user is one that has been authorized for a particular service, and the server has not received any indication that the user has relinquished access.

A stateless proxy is one that does not maintain transaction, nor session state. It frees the messages sent once acknowledgements are received by the transport layer.

A stateful proxy can be viewed as a DIAMETER Server upon receiving a request, and as a Client when forwarding the message. For all intensive purposes, stateful servers terminate an upstream "session", and initiates a downstream "session" (see figure x), and MAY provide the following features:

- Protocol translation (e.g. RADIUS <-> DIAMETER)
- Limiting resources authorized to a particular user
- Per user or transaction auditing

```
+--------+          +-----------------+          +--------+
| Client | --------> | Server | Client | -------> | Server |
+--------+          +-----------------+          +--------+
```

Figure x - Example of Stateful Proxy

A stateful proxy that maintains transaction state SHOULD release transaction information after a request's corresponding response has been forwarded towards the recipient, and has been acknowledged by the underlying transport.

A stateful proxy that maintains session state SHOULD release the session state once it is informed that a user and/or device is has relinquished access.

Home servers processing requests that include the Route-Record and/or the Proxy-State AVPs MUST return these AVPs in the same order in the corresponding response.

6.4.1 Proxying Requests

A proxy server MUST check for forwarding loops before proxying a request. A request has been looped if the server finds its own address in a Route-Record AVP (see [1] for more information on loop detection).

A DIAMETER server that proxies a request MUST append a Route-Record AVP, which includes its identity. DIAMETER Servers that receive messages MUST validate the last Route-Record AVP in the message and ensure that the host identified in the AVP is the same as the sender of the message.

A Proxy Server MAY also include the Proxy-State AVP, which is used to encode local state information. The Proxy-State AVP is guaranteed to be present in the corresponding response. If the Proxy-State AVP is present, both the Route-Record and the Proxy-State AVPs MUST be encapsulated within the Grouped-AVP AVP.

The message is then forwarded to the downstream DIAMETER server, as

identified in the Domain Routing Table.

6.4.2 Proxying Responses

A proxy server MUST only process responses whose last Route-Record AVP matches one of its addresses. Any responses that do not conform to this rule MUST be dropped. The last Route-Record AVP MUST be removed from the message before it is forwarded to the next hop, which is identified by the second to last Route-Record AVP.

6.4.2.1 Route-Record AVP

The Route-Record AVP (AVP Code 282) is of type String, and contains the Fully Qualified Domain Name of the Proxy appending the AVP to a DIAMETER message.

6.4.2.2 Proxy-State AVP

The Proxy-State AVP (AVP Code 33) [1] is used by proxy servers when forwarding requests and contains opaque data that is used by the proxy to further process the response. Such data may include AVPs that are to be added to the response, information about the downstream peer, etc.

6.4.2.3 Home-Realm AVP

The Home-Realm AVP (AVP Code 283) is of type String and contains the realm portion of the Network Access Identifier. When present, the Home-Realm AVP MUST be used to perform any message routing decisions.

6.5 Applying Local Policies

Proxies MAY apply local access policies to DIAMETER requests, or responses, by adding, changing or deleting AVPs in the messages. Proxies that apply local policies MUST NOT allow end-to-end security on any messages that traverse through it, unless security is terminated locally.

A proxy wishing to modify a DIAMETER message to enforce some local policy that detects that end-to-end security has been applied to the message MUST return a reponse to the originator with the Result-Code set to DIAMETER_NO_END_2_END_SECURITY. The originator of the request MAY re-issue the request with no end-to-end security if it falls

within its local policy.

In the event that the Home DIAMETER server receives a request with contradictory information (possibly due to some proxy adding a local policy), it MAY accept the latest AVP, or MAY return the response with the Result Code AVP set to DIAMETER_CONTRADICTING_AVPS. However, a NAS receiving a response that contains contradictory information SHOULD reject service to the user.

6.6 Hiding Network Topology

Stateful proxies forwarding requests to servers outside of their administrative domain MAY hide the internal network topology. Servers perform this by removing all Route-Record AVPs in the message, and maintains the Route-Record AVPs to add to the corresponding response. Such stateful servers MUST still add their own Route-Record AVP to the request prior to forwarding.

7.0 RADIUS Compatibility

The AAA Design Team has concluded that the protocol gateway procedures described in [x] is the correct approach. The procedures need to be validated to confirm that no errors exist.

It is possible that a new RADIUS attribute and DIAMETER AVP could be created and included when protocol translation occurs. This could be useful for troubleshooting purposes, but would have virtually no real-time benefits.

8.0 End-to-End Security

The DIAMETER Strong Security extension currently ONLY supports CMS in an asymmetric mode. It has been brought to the WGs attention that not all transactions require this level of security. The following additional security mechanisms need to be evaluated, and incorporated into the DIAMETER protocol (if they are deemed appropriate):

1. Symmetric CMS mode [22]. This allows for keys to be exchanged by DIAMETER peers via the CMS security system. However, it is not clear how such peers would agree on the keying material prior to the normal DIAMETER message flows. Further investigation is required.
2. Kerberos has been proposed as an alternative to establish dynamic security assocations (keying material) between DIAMETER peers. Further investigation needs to be completed before a

determination can be made.

9.0 Data Model

The sections which follow correspond to those in [11]. The data model 'solutions' address those issues which were identified in that document or recommend that these issues are better left unaddressed for now.

9.1 Separability of DIAMETER Header and Message

Suggestion: Use the DIAMETER message header Flags field, which is currently reserved. Assign the following flags.

```
0x04 ('E' Expect Reply) The message solicits a response.
0x02 ('I' Interrogation) The message is a Query or a Reply.
0x01 ('R' Response) The message is a response another message.
```

The following describes all combinations and their interpretation.

```
- - -    The message is an Indication
- - R    Not allowed.
- I -    Not allowed.
- I R    Not Allowed.
E - -    The message is a Request.
E - R    The message is an Answer.
E I -    The message is a Query.
E I R    The message is a Reply.
```

By examining these flags, even if a DIAMETER extension is not supported, it will still be clear what *kind* of DIAMETER message follows. Carrying this information in the message header separates the base protocol from the payload data. This has been shown to be useful in other protocols which are extensible and carry otherwise opaque data, such as SNMP. [19] The operation identifier can be used for logging, security policies, etc.

It is not clear in the current text why there is a 'Request/Answer' and is the result of the command. A 'Query' obtains some data which is returned in the 'Reply,' but no action is taken. Text clarifying the difference should be included in the base protocol specification.

9.2 Data Types supported in DIAMETER

The base data types in the current DIAMETER specification include

Data, String, Address, Integer32, Integer64, Time and Complex.

It is recommended that this list be changed to:

Address, Integer32, Unsigned32, Integer64, Unsigned64, Float32, Float64, Float128, OctetString, Grouped and List.

It is thuse recommended that the following data types no longer be supported:

String, Time and Complex.

Data will now be used for any piece of data - including a String.

Address is defined as in [1].

Unsigned32 is an unsigned 32 bit integer. Integer32 is a signed 32 bit integer. Unsigned32 replaces Time.

Unsigned64 is an unsigned 64 bit integer. Signed64 is a signed 64 bit integer.

Float32 is a 32 bit IEEE floating point number. Float64 is a 64 bit IEEE floating point number. Float128 is a 128 bit IEEE floating point number.

Group is a specific sequence of other data elements, each with their own type, defined in a particular AVP. For example, a group could be comprised of the data element of the following sequence:

{Foo AVP, Bar AVP}

A data element of this Group type MUST include both of these components, in the order specified. A Group is represented on the wire simply as each AVP is defined - one after the other. The Group type is analogous to an ASN.1 SEQUENCE [20].

List is a collection of 0 or more data elements of the same type. For example, one could have a List of Foo AVPs. A List is represented on the wire as an Unsigned32 value 'n' (which is set to the number of AVPs which follow.) The next 'n' AVPs which follow MUST be the type assigned by the DIAMETER Extension. In the case of this example, they would all be Foo AVPs. Note that the order of the data elements in the List is arbitrary (the list is not assumed to be sorted). The List type is analogous to an ASN.1 SET.

All quantities (address, signed, unsigned and floating point) are represented in network byte order.

Correct interpretation of the data element requires knowledge of the specific DIAMETER extension in which the AVP is defined. For example, Data could be anything - including a String. Unsigned32 could be any quantity - including Time. String and Time data values are not supported explicitely, according to this recommendation. Instead, they are implicit in a specific AVP definition.

9.3 Formal notation for application specific data types

Specific data type notations are in the process of being defined in separate documents (one example is provided in Appendix A). This is recommended for the following purposes:

- Allow extensible functionality for DIAMETER implementations, such as storage and type checking of new AVP types, without requiring new code,
- Allow extensible functionality for packet sniffers, debuggers management tools and human interfaces potentially without adding new code.

The formal notation is NOT intended to be used for the following purposes:

- As the definitive 'on-the-wire data representation' specification for specific AVPs
- As the definitive interpretation of the semantics of specific AVPs

In both cases, implementors must use the relevent DIAMETER Extension RFC where the DIAMETER AVP is defined.

9.4 Ordering of Data

Each AVP is self-describing (in the AVP header). It is possible for a DIAMETER implementation to accumulate all AVPs in a message without requiring the AVPs to be strictly ordered in the message.

There is a trade-off between simplicity and flexibility. Flexibility has shown itself useful in DIAMETER and RADIUS implementations in the past. It is not wise to impose arbitrary restrictions on protocols when they do not significantly simplify or result in better interoperability of protocol implementations. The recommendation is to not require strict ordering of AVPs within a message, except where this is required by a Group or List data type, as described in Section 9.2.

9.5 Semantics of the "M" bit

Some confusion has been expressed over the "M" bit in the header of DIAMETER AVPs. Concerns has been expressed about applicability of the "M" bit to non-IETF standard AVPs.

9.5.1 What does it mean?

The appropriate interpretation of the semantics of the "M", or mandatory, flag bit for DIAMETER AVPs is that the associated AVP is considered crucial to correct and secure delivery of the service specified by the collection of AVPs in a DIAMETER message. Proxies MUST NOT eliminate or modify the AVP and clients MUST NOT discard or fail to enforce the AVP.

9.5.2 From whose perspective?

The DIAMETER peer that sets an "M" bit on an AVP does so because there is a local policy that indicates that the AVP in question is crucial to the correct or secure provision of the service. These local policies are typically based on business requirements, rather than protocol or network operations requirements. AVPs with the "M" bit set are not "negotiable" by other DIAMETER peers.

9.5.3 What are correct error responses?

If a DIAMETER peer receives an AVP with the "M" bit set, and it does not recognize the AVP or does not support the service described by the AVP, it must reject the access request or treat the access response as if it were a rejection.

9.5.4 Can Vendor Specific AVPs use the "M" bit?

Vendor Specific AVPs may use the "M" bit to signify the importance of the AVP to the correct ands secure provision of service. If a DIAMETER peer rejects the DIAMETER message because it is unknown or unsupported, the rejection is to be considered correct protocol behavior, rather than an operational deficiency.

The issuer of the Vendor Specific AVP attaching the "M" bit MUST expect that rejection of service will occur in these cases, and take administrative action to correct the misconfiguration.

10.0 SNMP Support (DIAMETER MIB)

Certain work items in this area require short term attention, while some others requires longer term attention (and others are not to be done). These work items need to be identified and prioritized in a future version of this document.

11.0 Re-Authentication & Authorization

The text found in the section 11.1 is to be added in the NASREQ extension [2], while the text in section 11.2 is to be added to the DIAMETER base protocol [1].

11.1 Re-authentication/Re-authorization

The DIAMETER protocol allows for users to be periodically re-authenticated and/or re-authorized. In such instances, an AAR message would be sent with a Session-Id AVP that MUST be the same value as the one in the original authentication/authorization message. A DIAMETER server informs the NAS of the authorized session lifetime via the Authorization-Lifetime AVP.

A NAS MUST re-authenticate and/or authorize after the period provided by the server. Furthermore, it is possible for DIAMETER servers to issue an unsolicited re-authentication and/or re-authorization by issuing an AA-Challenge-Ind message to the NAS. The Session-Id AVP MUST have the same value as the original request. Upon receipt of such a message, the NAS is instructed to issue a request to re-authenticate and/or re-authorize the client.

11.2 Authorization-Lifetime AVP

The Authorization-Lifetime AVP (AVP Code TBD) is of type Integer32 and contains the maximum number of seconds of service to be provided to the user before the user is to be re-authenticated and/or re-authorized. Great care should be taken when the Authorization-Lifetime value is determined, since a low value could create significant DIAMETER traffic, which could congest both the network and the servers.

This AVP MAY be provided by the client as a hint of the maximum duration that it is willing to accept. However, the server DOES NOT have to observe the hint, and MAY return a value that is smaller than the hint. A value of zero provided by a client DOES NOT imply that service is being terminated.

12.0 Server/Resource Management State

There are several significant technical issues to be solved in the area of distributed resource management, not the least of which is recovery of state in the face of failures of clients, servers or networks. The maintenance and recovery of state may be broken down into two classes, tight consistency and loose consistency.

12.1 Loose Consistency

Loosely consistent distributed state is arguably easier to achieve in a reliable, scalable fashion. Loose consistency is characterized by:

1. delay state recovery until the information is actually needed

2. recovery of information in a directed fashion, avoiding the use of broadcast messages

3. be only as restrictive as necessary for correct network operation and substantial revenue loss avoidance.

There are two major classes of resource management that are important in the areas of applicability for DIAMETER. The first is the assignment of a network address. The second is the limitation of simultaneous login of users.

In the case of network address assignment, it is important for reasons of correct network operation to avoid assigning duplicate addresses. After a loss of state at the server, it may be possible to delay state reconciliation on any given address until that address is to be (re) assigned. One possible solution would to be to PING addresses before assignment, to determine their availability.

In the case of exclusive login, state is maintained for the single, or possibly limited multiple, login session(s) of a single user. In the event of loss of state, it may be reasonable to give the benefit of the doubt to an new user, until such time as state might be recovered. If the new user is determined to be a duplicate, that session could be terminated, by server request. It may be sufficient, in terms of limitation of potential revenue loss, to loosely control the simultaneous login.

While additional work must be done to specify the details of a loose consistency approach, it may be possible to do so within the time goals for protocol development, and yet sufficiently meet the business requirements for resource management, so as to make this a useful feature of the protocol.

12.2 Tight Consistency

The problem of obtaining reliable, scalable, distributed resource state, in a tightly consistent fashion is a difficult problem. It is not clear that there are valid underlying requirements for tight consistency, nor valid business reasons to support it at this time. This is a topic for further study.

13.0 Access Rules and Filters

The following text is intended to replace the current filter format, described in section 2.1.2 of [2].

13.1 Filter-Rule AVP

The Filter-Rule AVP (AVP Code 400) is of type String and provides filter rules that need to be configured on the NAS for the user. One or more such AVPs MAY be present in an authorization response.

Each packet can be filtered based on the following information that is associated with it:

```
Direction                          (in or out)
Source and destination IP address  (possibly masked)
Protocol
Source and destination port        (lists or ranges)
TCP flags
IP fragment flag
IP options
ICMP types
```

Rules for the appropriate direction are evaluated in order, with the first matched rule terminating the evaluation. Each packet is evaluated once. If no rule matches, the packet is dropped if the last rule evaluated was a permit, and passed if the last rule was a deny.

The filters in the Filter-Rule AVP MUST follow the format:

```
action dir proto from src to dst [options]

action       permit - Allow packets that match the rule.
             deny - Drop packets that match the rule.

dir          "in" is from the terminal, "out" is to the terminal.

proto        An IP protocol specified by number.  The "ip" keyword
             means any protocol will match.
```

```
src and dst  <address/mask> [ports]

             The <address/mask> may be specified as:
             ipno       An IPv4 or IPv6 number in dotted-quad or
                        canonical IPv6 form. Only this exact IP
                        number will match the rule.
             ipno/bits  An IP number as above with a mask width of
                        the form 1.2.3.4/24.  In this case all IP
                        numbers from 1.2.3.0 to 1.2.3.255 will
                        match.  The bit width MUST be valid for
                        the IP version and the IP number MUST NOT
                        have bits set beyond the mask.

             The sense of the match can be inverted by preceding
             an address with the not modifier, causing all other
             addresses to be matched instead.  This does not
             affect the selection of port numbers.

                The keyword "any" is 0.0.0.0/0 or the IPv6
                equivalent.  The keyword "assigned" is the address
                or set of addresses assigned to the terminal.  It
                is strongly suggested that the first rule be "deny
                in ip !assigned".  [[I would go further and state
                that this rule is mandatory and implied, so the
                NAS MUST provide source address assurance in all
                cases.  BW]]

             With the TCP and UDP protocols, optional ports may be
             specified as:

                {port|port-port}[,port[,...]]

             The `-' notation specifies a range of ports
             (including bound- aries).

             Fragmented packets which have a non-zero offset (i.e.
             not the first fragment) will never match a rule which
             has one or more port specifications.  See the frag
             option for details on matching fragmented packets.

options:
   frag    Match if the packet is a fragment and this is not the
           first fragment of the datagram.  frag may not be used
           in conjunction with either tcpflags or TCP/UDP port
           specifi- cations.

   ipoptions spec
           Match if the IP header contains the comma separated
```

list of options specified in spec. The supported IP options are:

ssrr (strict source route), lsrr (loose source route), rr (record packet route) and ts (timestamp). The absence of a particular option may be denoted with a `!'.

tcpoptions spec
Match if the TCP header contains the comma separated list of options specified in spec. The supported TCP options are:

mss (maximum segment size), window (tcp window advertisement), sack (selective ack), ts (rfc1323 timestamp) and cc (rfc1644 t/tcp connection count). The absence of a particular option may be denoted with a `!'.

established
TCP packets only. Match packets that have the RST or ACK bits set.

setup
TCP packets only. Match packets that have the SYN bit set but no ACK bit.

tcpflags spec
TCP packets only. Match if the TCP header contains the comma separated list of flags specified in spec. The supported TCP flags are:

fin, syn, rst, psh, ack and urg. The absence of a particular flag may be denoted with a `!'. A rule which contains a tcpflags specification can never match a fragmented packet which has a non-zero offset. See the frag option for details on matching fragmented packets.

icmptypes types
ICMP packets only. Match if the ICMP type is in the list types. The list may be specified as any combination of ranges or individual types separated by commas. The supported ICMP types are:

echo reply (0), destination unreachable (3), source quench (4), redirect (5), echo request (8), router advertisement (9), router solicitation (10), time-to-live exceeded (11), IP header bad (12), timestamp request (13), timestamp reply (14), information request

(15), information reply (16), address mask request (17) and address mask reply (18).

There is one kind of packet that the NAS MUST always discard, that is an IP fragment with a fragment offset of one. This is a valid packet, but it only has one use, to try to circumvent firewalls.

A NAS that is unable to interpret or apply a deny rule MUST terminate the session. A NAS that is unable to interpret or apply a permit rule MAY apply a more restrictive rule. A NAS MAY apply deny rules of its own before the supplied rules, for example to protect the NAS owner's infrastructure.

The rule syntax is a modified subset of ipfw(8) from FreeBSD, and the ipfw.c code may provide a useful base for implementations.

13.2 Non-IP Filters

Filter syntax and semantics of equivalent power should be provided for other session protocols supported by DIAMETER. Details are for further study.

14.0 AAA Server Discovery

Allowing for dynamic AAA server discovery will make it possible for simpler and more robust deployment of AAA services. In order to promote interoperable implemenations of AAA server discovery, the following mechanisms are described. These are based on existing IETF standards.

There are two cases where AAA server discovery may be performed. The first is when a AAA client needs to discover a first-hop AAA server. The second case is when an AAA server needs to discover another AAA server - for further handling of an AAA operation. In both cases, the following 'search order' is recommended:

1. The AAA implementation consults its list of static (manual) configured AAA server locations. These will be used if they exist and respond.

2. The AAA implementation uses SLPv2 [13] to discover DIAMETER services. The AAA service template [14] is included below, in section 14.1. It is recommended that SLPv2 security be deployed (this requires distributing keys to SLPv2 agents.) This is discussed further in Section 14.1.

SLPv2 will allow AAA implementations to discover the location of AAA servers in the local site, as well as their characteristics. AAA servers with specific capabilities (say support for the Accounting extension) can be requested, and only those will be discovered.

3. The AAA implementation uses DNS to request the SRV RR [15] for the '_diameter._sctp' server in a particular domain. The AAA implementation has to know in advance which domain to look for an AAA server in. This could be deduced, for example, from the 'realm' in a NAI that an AAA implementation needed to perform an AAA operation on.

   DIAMETER allows AAA peers to protect the integrity and privacy of communication as well as to perform end-point authentication. Still, it is prudent to employ DNS Security as a precaution when using DNS SRV RRs to look up the location of a DIAMETER server. [16, 17, 18]

14.1 AAA Service Template

The following service template describes the attributes used by AAA servers to advertise themselves. This simplifies the process of selecting an appropriate server to communicate with. An AAA client can request specific AAA servers based on characteristics of the AAA service desired (for example, an AAA server to use for accounting.)

Name of submitter: "Erik Guttman" <Erik.Guttman@sun.com>
Language of service template: en

Security Considerations:
AAA clients and servers use various cryptographic mechanisms to protect communication integrity, confidentiality as well as perform end-point authentication. It would thus be difficult if not impossible for an attacker to advertise itself using SLPv2 and pose as a legitimate AAA peer without proper preconfigured secrets or cryptographic keys. Still, as AAA services are vital for network operation it is important to use SLPv2 authentication to prevent an attacker from modifying or eliminating service advertisements for legitimate AAA servers.

Template text:

```
-------------------------template begins here-----------------------
template-type=service:diameter

template-version=0.0
```

```
template-description=
  The DIAMETER protocol is defined by draft-calhoun-diameter-17.txt

template-url-syntax=
  url-path= ; The standard service URL syntax is used.
            ; For example: 'service:diameter://aaa.example.com:1812

   supported-extensions= string L M
   # This attribute lists the DIAMETER extensions supported by the
   # AAA implementation.  The extensions currently defined are:
   #  Extension Name       Defined by
   #  ---------------      -----------------------------------
   #  NASREQ               draft-calhoun-diameter-nasreq-05.txt
   #  MobileIP             draft-calhoun-diameter-mobileip-11.txt
   #  Accounting           draft-calhoun-diameter-accounting-08.txt
   #  Strong Security      draft-calhoun-diameter-strong-crypto-05.txt
   #  Resource Management  draft-calhoun-diameter-res-mgmt-06.txt
   #
   # Notes:
   #   . AAA implementations support one or more extensions.
   #   . Additional extensions may be defined in the future.
   #     An updated service template will be created at that time.
   #
   NASREQ,MobileIP,Accounting,Strong Security,Resource Management

   supported-transports= string L M
   SCTP
   # This attribute lists the supported transports that the DIAMETER
   # implementation accepts.  Note that a compliant DIAMETER
   # implementation MUST support SCTP, though it MAY support other
   # transports, too.
   SCTP,TCP

-------------------------template ends here-----------------------
```

15.0 Loop Detection

This section describes how proxies detect messages that are looping through the same set of entities. This section is targetted to be numbered 6.7 in the DIAMETER Base protocol [1].

15.1 Loop Detection

When a DIAMETER Proxy or Redirect server receives a request, it MUST examine all Route-Record AVPs in the message to determine whether such an AVP already exists with the local server's identity. If an AVP with the local host's identity is found in the request, it is an

indication that the message is being looped through the same set of proxies. When such an event occurs, the DIAMETER server that detects the loop returns a response with the Result-Code AVP set to DIAMETER_LOOP_DETECTED.

16.0 IANA Considerations

DIAMETER makes extensive use of IDs (command codes, extensions, result codes, AVP attributes, Integrity-Check-Value AVP Transform code). These are collected in the base protocol specification, but defined in the DIAMETER extension docs. The Working Group needs to make sure that IANA is notified that a registry needs to be created to keep track of all DIAMETER identifiers.

17.0 Security Considerations

DIAMETER [1] is a framework providing authentication and authorization services for network access. Section 11 and 13 concern how these features could be refined or improved in subsequent work.

DIAMETER itself contains a number of security features. Section 8 discusses how these could be redesigned for less reliance on public key cryptography.

The security implications of using attribute-based service discovery to locate AAA servers is discussed in Section 14.1.

18.0 Acknowledgments

The authors would like to thank the AAA Design Team for raising the issues in [11], which were used in the creation of this document.

19.0 References

[1] P. Calhoun, A. Rubens, H. Akhtar, E. Guttman. "DIAMETER Base Protocol", draft-calhoun-diameter-17.txt, IETF work in progress, September 2000.

[2] P. Calhoun, W. Bulley, A. Rubens, J. Haag, "DIAMETER NASREQ Extension", draft-calhoun-diameter-nasreq-05.txt, IETF work in progress, September 2000.

[3] Calhoun, Zorn, Pan, Akhtar, "DIAMETER Framework", draft-

calhoun-diameter-framework-08.txt, IETF work in progress, June 2000.

[4] P. Calhoun, C. Perkins, "DIAMETER Mobile IP Extensions", draft-calhoun-diameter-mobileip-11.txt, IETF work in progress, September 2000.

[5] P. Calhoun, W. Bulley, S. Farrell, "DIAMETER Strong Security Extension", draft-calhoun-diameter-strong-crypto-05.txt (work in progress), September 2000.

[6] Arkko, Calhoun, Patel, Zorn, "DIAMETER Accounting Extension", draft-calhoun-diameter-accounting-08.txt, IETF work in progress, September 2000.

[7] P. Calhoun, A. Rubens, H. Akhtar, E. Guttman, W. Bulley, J. Haag, "DIAMETER Implementation Guidelines", draft-calhoun-diameter-impl-guide-03.txt, IETF work in progress, June 2000.

[8] P. Calhoun, N. Greene, "DIAMETER Resource Management", draft-calhoun-diameter-res-mgmt-05.txt, IETF Work in Progress, September 2000.

[9] Aboba et al, "Network Access AAA Evaluation Criteria", IETF work in progress, draft-ietf-aaa-na-reqts-07.txt, August 2000.

[10] Mitton et al, "Authentication, Authorization, and Accounting: Protocol Evaluation", IETF work in progress, draft-ietf-aaa-proto-eval-00.txt, July 2000.

[11] Calhoun et al., "AAA Problem Statements", IETF work in progress, draft-ietf-aaa-issues-01.txt, October 2000.

[12] Aboba, Beadles "The Network Access Identifier." RFC 2486. January 1999.

[13] E. Guttman, C. Perkins, J. Veizades, M. Day, "Service Location Protocol, Version 2", RFC 2608, June 1999.

[14] E. Guttman, C. Perkins, J. Kempf, "Service Templates and Service: Schemes", RFC 2609, June 1999.

[15] A. Gulbrandsen, P. Vixie, L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[16] D. Eastlake, "Domain Name System Security Extensions", RFC 2535, March 1999.

[17] D. Eastlake, "DNS Security Operational Considerations", RFC 2541, March 1999.

[18] D. Eastlake, "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, September 2000.

[19] D. Harrington et. al., "An Architecture for Describing SNMP Management Frameworks", RFC 2571, May 1999.

[20] Information processing systems - Open Systems Interconnection, "Specification of Abstract Syntax Notation One (ASN.1)", International Organization for Standardization, International Standard 8824, December 1987.

[21] B. Aboba, G. Zorn, D. Mitton, "RADIUS and IPv6", draft-aboba-radius-ipv6-02.txt, IETF Work in Progress, August 2000.

[22] S. Farrell, S. Turner, "Reuse of CMS Content Encryption Keys", draft-ietf-smime-rcek-00.txt, IETF work in progress, September 2000.

20.0 Authors' Addresses

Questions about this memo can be directed to:

Jari Arkko
Oy LM Ericsson Ab
02420 Jorvas
Finland

Phone: +358 40 5079256
E-Mail: Jari.Arkko@ericsson.com

Pat R. Calhoun
Network and Security Research Center, Sun Labs
Sun Microsystems, Inc.
15 Network Circle
Menlo Park, California, 94025
USA

Phone: +1 650-786-7733
Fax: +1 650-786-6445
E-mail: pcalhoun@eng.sun.com

Erik Guttman

Network and Security Research Center, Sun Laboratories
Sun Microsystems, Inc.
Eichhoelzelstr. 7
74915 Waibstadt
Germany

Phone: +49-7263-911-701
E-mail: erik.guttman@germany.sun.com

David B. Nelson
Enterasys Networks, Inc. (a Cabletron Systems company)
50 Minuteman Road
Andover, MA 01810-1008
USA

Phone: +1 978 684 1330
E-Mail: dnelson@enterasys.com

Barney Wolff, Pres.
Databus Inc.
15 Victor Drive
Irvington, NY 10533-1919 USA
USA

Phone: +1 914 591 5677
E-mail: barney@databus.com

21.0 Full Copyright Statement

Appendix A - DIAMETER Data Dictionary XML DTD

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE diameter [

   <!-- NOTE:  This DTD describes the set of AVPs which may be  -->
   <!-- supported by a particular DIAMETER implementation.      -->

   <!-- Since DIAMETER is extensible, not every DIAMETER        -->
   <!-- implementation will support all the AVPs described in a -->
   <!-- a document conforming to this DTD.                      -->

   <!-- This DTD describes AVPs for the purpose of simplifying  -->
   <!-- extensibility of DIAMETER implementations, protocol     -->
   <!-- analyzers and tools.  This DTD does not define the      -->
   <!-- on-the-wire presentation of AVPs.  For this information -->
   <!-- refer to the cited specifications for the base protocol -->
   <!-- and extensions.                                         -->

   <!-- ------------------------------------------------------- -->
   <!-- diameter data dictionary definition                     -->
   <!-- ------------------------------------------------------- -->

   <!-- 'diameter' describes the base protocol and the AVPs     -->
   <!-- it supports.                                            -->
   <!ELEMENT diameter ( base avp* extension+ )>

   <!-- 'base' is the url of the spec of the base protocol.     -->
   <!ELEMENT base (#PCDATA)>


   <!-- ------------------------------------------------------- -->
   <!-- extension definition                                    -->
   <!-- ------------------------------------------------------- -->

   <!ELEMENT extension ( extname doc avp+ )>

   <!-- 'extname' is the name of the extension.                 -->
   <!ELEMENT extname (#PCDATA)>

   <!-- 'doc' is the url of the spec defining the extension.    -->
   <!ELEMENT doc (#PCDATA)>


   <!-- ------------------------------------------------------- -->
   <!-- avp definition                                          -->
   <!-- ------------------------------------------------------- -->
```

```
      <!-- 'avp' describes the attributes in the extension.        -->
      <!ELEMENT avp ( name description? type value* )>

      <!-- 'name' is the name of an AVP in the DIAMETER extension. -->
      <!ELEMENT name (#PCDATA)>

      <!-- 'description' text describes the AVP.                   -->
      <!ELEMENT description (#PCDATA)>

      <!-- 'value' is a value-item/description pair defined for    -->
      <!--  the AVP.                                               -->
      <!ELEMENT value (val desc)>

      <!-- 'val' is a specific value.                         -->
      <!ELEMENT val (#PCDATA)>

      <!-- 'desc' describes the specific value.         -->
      <!ELEMENT desc (#PCDATA)>

      <!-- ------------------------------------------------------- -->
      <!--  avp type definition                                    -->
      <!-- ------------------------------------------------------- -->

      <!-- 'type' is the data type for AVPs.                       -->
      <!-- Values can be: OctetString, Unsigned32,                 -->
      <!--   Integer32, Unsigned64, Integer64, Float32, Float64,   -->
      <!--   Float128, Grouped or List.                            -->
      <!--   In the case of Grouped or List, AVPs are              -->
      <!--   nested in the AVP definition as specified below.      -->
      <!ELEMENT type (#PCDATA | grouped | list)>

      <!-- 'grouped' defines a sequence of AVPs that constitute    -->
      <!-- an AVP type collectively.  2 or more AVPs are included  -->
      <!-- in each grouped value.  An instance of an AVP with type -->
      <!-- 'grouped' is followed by instances of the given         -->
      <!--  sequence of AVPs.                                      -->
      <!ELEMENT grouped ( seq-name seq-name+)>

      <!-- 'list' defines an unordered set of AVPs of the same     -->
      <!-- type.  One AVP is included.  An AVP with type 'list'    -->
      <!-- contains an Unsigned32 number 'n', which is followed    -->
      <!-- by n instances of the AVP given.                        -->
      <!ELEMENT list (set-name)>


      <!-- ------------------------------------------------------- -->
      <!-- Mandatory avp attributes                                -->
      <!-- ------------------------------------------------------- -->
```

```
      <!-- 'code' is the reserved unsigned 32 bit ID of the avp.   -->
      <!ATTLIST avp code CDATA #REQUIRED)>

      <!-- 'may-encrypt' : is encrpytion permitted?                -->
      <!ATTLIST avp may-encrypt (true false) #REQUIRED>


      <!-- ------------------------------------------------------- -->
      <!-- Optional avp and type attributes                        -->
      <!-- ------------------------------------------------------- -->

      <!-- 'mandatory-flag' : is the 'M' flag disallowed? required?-->
      <!--   If not included, use of the flag is OPTIONAL.         -->
      <!ATTLIST avp mandatory-flag (disallowed allowed required)
        #IMPLIED "allowed" >

      <!-- 'vendor-flag' : is the 'V' flag disallowed?             -->
      <!--   If not included, use of the flag is allowed.          -->
      <!ATTLIST avp vendor-flag (disallowed allowed)
        #IMPLIED "allowed">

      <!-- 'constrained' : is the list of legal values constrained -->
      <!--   to those in the value list?                           -->
      <!ATTLIST avp constrained (true false) #IMPLIED "false">

      <!-- 'printable' is used to indicate that data of type       -->
      <!--   OctetString contains a printable string.  This        -->
      <!--   distinguishes between a printable string and opaque   -->
      <!--   octets.                                               -->
      <!ATTLIST data printable (true false) #IMPLIED "false">

      <!-- 'ip-address' is used to indicate that data of type      -->
      <!--   OctetString contains an IP address.                   -->
      <!ATTLIST data ip-address (true false) #IMPLIED "false">

      <!-- 'time' is used to indicate that data of type Unsigned32 -->
      <!--   is in fact a time value - the four most significant   -->
      <!--   bytes of an NTP timestamp. [RFC 2030]                 -->
      <!ATTLIST data time (true false) #IMPLIED "false">
   ]>

   <diameter>
     <base>
       ftp://ftp.ietf.org/internet-drafts/draft-calhoun-diameter-17.txt

       <avp code="1" may-encrypt="true">
         <name> User-Name </name>
         <type printable="true"> OctetString </type>
```

```
       </avp>


       <avp code="27" may-encrypt="true">
         <name> Session-Timeout </name>
         <type> Unsigned32 </type>
       </avp>

       <avp code="33" may-encrypt="false"
        vendor-flag="disallowed" mandatory-flag="required">
         <name> Proxy-State </name>
         <type>
           <grouped>
             <member-name> State-Creator </member-name>
             <member-name> State-Opaque </member-name>
           </grouped>
         </type>
       </avp>

       <avp code="TBD" may-encrypt="false">
         <name> State-Creator </name>
         <type ip-address="true"> OctetString </type>
         <description>
           The value of this AVP contains the address of the creator
           of the proxy state.
         </description>
       </avp>

       <avp code="TBD" may-encrypt="false">
         <name> State-Opaque </name>
         <type> OctetString </type>
         <description>
           The value of this AVP contains arbitrary state data.
         </description>
       </avp>

       <avp code="257" may-encrypt="false"
        vendor-flag="disallowed" mandatory-flag="required">
         <name> Host-IP-Address </name>
         <type ip-address="true"> OctetString </type>
       </avp>

       <avp code="258" may-encrypt="true">
         <name> Extension-Id</name>
         <type> Unsigned32 </type>
         <value>
           <val> 1 </val>
           <desc> NASREQ </desc>
```

```
    </value>
    <value>
      <val> 2 </val>
      <desc> Strong Security </desc>
    </value>
    <value>
      <val> 3 </val>
      <desc> Resource Management </desc>
    </value>
    <value>
      <val> 4 </val>
      <desc> Mobile-IP </desc>
    </value>
    <value>
      <val> 5 </val>
      <desc> Accounting </desc>
    </value>
  </avp>

  <avp code="259" may-encrypt="false">
    <name> Integrity-Check-Vector </name>
    <type>
      <grouped>
        <member-name> Transform-ID </member-name>
        <member-name> Key-ID </member-name>
        <member-name> ICV-Data </member-name>
      </grouped>
    </type>
  </avp>

  <avp code="TBD" may-encrypt="false">
    <name> Transform-ID </name>
    <type> Unsigned32 </type>
  </avp>

  <avp code="TBD" may-encrypt="false">
    <name> Key-ID </name>
    <type> Unsigned32 </type>
  </avp>

  <avp code="TBD" may-encrypt="false">
    <name> ICV-Data </name>
    <type> OctetString </type>
  </avp>

  <avp code="260" may-encrypt="false">
    <name> Encrypted-Payload </name>
    <type>
```

```
          <grouped>
            <member-name> Transform-ID </member-name>
            <member-name> Key-ID </member-name>
            <member-name> Encrypted-Payload-Data </member-name>
          </grouped>
        </type>
      </avp>

      <avp code="TBD" may-encrypt="false">
        <name> Encrypted-Payload-Data </name>
        <type> OctetString </type>
      </avp>

      <avp code="261" may-encrypt="false">
        <name> Nonce </name>
        <type> OctetString </type>
      </avp>

      <avp code="262" may-encrypt="false">
        <name> Timestamp </name>
        <type time="true"> Unsigned32 </type>
      </avp>

      <avp code="263" may-encrypt="true">
        <name> Session-Id </name>
        <type> OctetString </type>
      </avp>

      <avp code="264" may-encrypt="false"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Host-Name </name>
        <type printable="true"> OctetString </type>
      </avp>

      <avp code="266" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="disallowed">
        <name> Vendor-Name </name>
        <type printable="true"> OctetString </type>
      </avp>

      <avp code="267" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="disallowed">
        <name> Firmware-Revision </name>
        <type> Unsigned32 </type>
      </avp>

      <avp code="268" may-encrypt="false">
        <name> Result-Code </name>
```

```
      <type>
        <grouped>
          <member-name> Result-Error-Code </member-name>
          <member-name> Result-Error-String </member-name>
        </grouped>
      </type>
    </avp>

    <avp code="TBD" may-encrypt="false constrained="true">
      <name> Result-Error-Code </name>
      <type> Unsigned32 </type>
      <description>
        This  DIAMETER error codes.
      </description>
      <value>
        <val>  0 </val>
        <desc> DIAMETER_SUCCESS </desc>
      </value>
      <value>
        <val>  1 </val>
        <desc> DIAMETER_FAILURE </desc>
      </value>
      <value>
        <val>  2 </val>
        <desc> DIAMETER_POOR_REQUEST </desc>
      </value>
      <value>
        <val>  3 </val>
        <desc> DIAMETER_INVALID_AUTH </desc>
      </value>
      <value>
        <val>  4 </val>
        <desc> DIAMETER_UNKNOWN_SESSION_ID </desc>
      </value>
      <value>
        <val>  5 </val>
        <desc> DIAMETER_USER_UNKNOWN </desc>
      </value>
      <value>
        <val>  6 </val>
        <desc> DIAMETER_COMMAND_UNSUPPORTED </desc>
      </value>
      <value>
        <val>  7 </val>
        <desc> DIAMETER_TIMEOUT </desc>
      </value>
      <value>
        <val>  8 </val>
```

```
          <desc> DIAMETER_AVP_UNSUPPORTED </desc> </value>
        <value>
          <val>  9 </val>
          <desc> DIAMETER_REDIRECT_INDICATION </desc>
        </value>
        <value>
          <val> 10 </val>
          <desc> DIAMETER_REALM_NOT_SERVED </desc>
        </value>
        <value>
          <val> 11 </val>
          <desc> DIAMETER_UNSUPPORTED_TRANSFORM </desc>
        </value>
        <value>
          <val> 12 </val>
          <desc> DIAMETER_AUTHENTICATION_REJECTED </desc>
        </value>
        <value>
          <val> 13 </val>
          <desc> DIAMETER_AUTHORIZATION_REJECTED </desc>
        </value>
        <value>
          <val> 14 </val>
          <desc> DIAMETER_INVALID_AVP_VALUE </desc>
        </value>
        <value>
          <val> 15 </val>
          <desc> DIAMETER_MISSING_AVP </desc>
        </value>
        <value>
          <val> 16 </val>
          <desc> DIAMETER_ERROR_BAD_KEY </desc>
        </value>
        <value>
          <val> 17 </val>
          <desc> DIAMETER_ERROR_BAD_HOME_ADDRESS </desc>
        </value>
        <value>
          <val> 18 </val>
          <desc> DIAMETER_ERROR_TOO_BUSY </desc>
        </value>
        <value>
          <val> 19 </val>
          <desc> DIAMETER_ERROR_MIP_REPLY_FAILURE </desc>
        </value>
        <value>
          <val> 20 </val>
          <desc> DIAMETER_INVALID_CMS_DATA </desc>
```

```
          </value>
          <value>
            <val> 21 </val>
         <desc> DIAMETER_ERROR_BAD_HAR-day </desc>
          </value>
        </avp>

        <avp code="TBD" may-encrypt="false">
          <name> Result-Error-String </name>
          <type printable="true"> OctetString </type>
          <description>
            This contains an optional human readable string.  If this
            field is not included, the printable string data is of 0
            length.
          </description>
        </avp>

        <avp code="269" may-encrypt="true">
          <name> Destination-NAI </name>
          <type printable="true"> OctetString </type>
        </avp>

        <avp code="270" may-encrypt="true">
          <name> Unknown-Command-Code </name>
          <type> Unsigned32 </type>
        </avp>

        <avp code="279" may-encrypt="true">
          <name> Failed-AVP </name>
          <type> OctetString </type>
        </avp>

        <avp code="278" may-encrypt="true">
          <name> Redirect-Host </name>
          <type ip-address="true"> OctetString </type>
        </avp>

        <avp code="277" may-encrypt="true">
          <name> Redirect-Host-Port </name>
          <type> Unsigned32 </type>
        </avp>

        <avp code="280" may-encrypt="false">
          <name> Grouped </name>
          <type ip-address="true"> OctetString </type>
        </avp>

        <avp code="278" may-encrypt="false">
```

```
        <name> Redirect-Host </name>
        <type ip-address="true"> OctetString </type>
      </avp>

    </base>

    <!-- -------------------------------------------------------- -->

    <extension>

      <extname> DIAMETER NASREQ Extensions </extname>

      <doc>
         ftp://ftp.ietf.org/internet-drafts/
         draft-calhoun-diameter-nasreq-05.txt
      </doc>

      <avp code="2" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> User-Password </name>
        <type> OctetString </type>
      </avp>

      <avp code="3" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> CHAP-Password </name>
        <type> OctetString </type>
      </avp>

      <avp code="4" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> NAS-IP-Address </name>
        <type ip-address="true"> OctetString </type>
      </avp>

      <avp code="5" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> NAS-Port </name>
        <type> Unsigned32 </type>
      </avp>

      <avp code="6" may-encrypt="true" constrained="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Service-Type </name>
        <type> Unsigned32 </type>
        <value>
          <val> 1 </val>
          <desc> Login </desc>
```

```
    </value>
    <value>
      <val> 2 </val>
      <desc> Framed </desc>
    </value>
    <value>
      <val> 3 </val>
      <desc> Callback Login </desc>
    </value>
    <value>
      <val> 4 </val>
      <desc> Callback Framed </desc>
    </value>
    <value>
      <val> 5 </val>
      <desc> Outbound </desc>
    </value>
    <value>
      <val> 6 </val>
      <desc> Administrative </desc>
    </value>
    <value>
      <val> 7 </val>
      <desc> NAS Prompt </desc>
    </value>
    <value>
      <val> 8 </val>
      <desc> Authenticate Only </desc>
    </value>
    <value>
      <val> 9 </val>
      <desc> Callback NAS Prompt </desc>
    </value>
  </avp>

  <avp code="7" may-encrypt="true" constrained="true"
   vendor-flag="disallowed" mandatory-flag="required">
    <name> Framed-Protocol </name>
    <type> Unsigned32 </type>
    <value> <val> 1 </val> <desc> PPP </desc> </value>
    <value> <val> 2 </val> <desc> PPP </desc> </value>
    <value>
      <val> 3 </val>
      <desc> AppleTalk Remote Access Protocol (ARAP) </desc>
    </value>
    <value>
      <val> 4 </val>
      <desc>
```

```
         Gandalf proprietary SingleLink/MultiLink protocol
          </desc>
        </value>
        <value>
          <val> 5 </val>
          <desc> Xylogics proprietary IPX/SLIP </desc>
        </value>
        <value> <val> 6 </val> <desc> X.75 Synchronous </desc> </value>
      </avp>

      <avp code="8" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Framed-IP-Address </name>
        <type ip-address="true"> OctetString </type>
      </avp>

      <avp code="9" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Framed-IP-Netmask </name>
        <type ip-address="true"> OctetString </type>
      </avp>

      <avp code="10" may-encrypt="true" constrained="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Framed-Routing </name>
        <type> Unsigned32 </type>
        <value> <val> 0  </val> <desc> None </desc> </value>
        <value>
          <val> 1 </val>
          <desc> Send routing packets </desc>
        </value>
        <value>
          <val> 2 </val>
          <desc> Listen for routing packets </desc>
        </value>
        <value> <val> 3 </val> <desc> Send and Listen </desc> </value>
      </avp>

      <avp code="11" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Filter-Id </name>
        <type printable="true"> OctetString </type>
      </avp>

      <avp code="12" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Framed-MTU </name>
        <type> Unsigned32 </type>
```

```
        &lt;/avp&gt;

        <avp code="13" may-encrypt="true"
         vendor-flag="disallowed" mandatory-flag="required">
          &lt;name&gt; Framed-Compression &lt;/name&gt;
          &lt;type&gt; Unsigned32 &lt;/type&gt;
          &lt;value&gt;
            &lt;val&gt; 0 &lt;/val&gt;
         &lt;desc&gt; None &lt;/desc&gt;
          &lt;/value&gt;
          &lt;value&gt;
            &lt;val&gt; 1 &lt;/val&gt;
         &lt;desc&gt; VJ TCP/IP header compression &lt;/desc&gt;
          &lt;/value&gt;
          &lt;value&gt;
            &lt;val&gt; 2 &lt;/val&gt;
         &lt;desc&gt; IPX header compression &lt;/desc&gt;
          &lt;/value&gt;
          &lt;value&gt;
            &lt;val&gt; 3 &lt;/val&gt;
         &lt;desc&gt; Stac-LZS compression &lt;/desc&gt;
          &lt;/value&gt;
        &lt;/avp&gt;

        <avp code="14" may-encrypt="true"
         vendor-flag="disallowed" mandatory-flag="required">
          &lt;name&gt; Login-IP-Host &lt;/name&gt;
          &lt;type printable="true"&gt; OctetString &lt;/type&gt;
        &lt;/avp&gt;

        <avp code="15" may-encrypt="true" constrained="true"
         vendor-flag="disallowed" mandatory-flag="required">
          &lt;name&gt; Login-Service &lt;/name&gt;
          &lt;type&gt; Unsigned32 &lt;/type&gt;
          &lt;value&gt;
            &lt;val&gt; 0 &lt;/val&gt;
         &lt;desc&gt; Telnet &lt;/desc&gt;
          &lt;/value&gt;
          &lt;value&gt;
            &lt;val&gt; 1 &lt;/val&gt;
         &lt;desc&gt; Rlogin &lt;/desc&gt;
          &lt;/value&gt;
          &lt;value&gt;
            &lt;val&gt; 2 &lt;/val&gt;
         &lt;desc&gt; TCP Clear &lt;/desc&gt;
          &lt;/value&gt;
          &lt;value&gt;
            &lt;val&gt; 3 &lt;/val&gt;
```

```
         <desc> PortMaster (proprietary) </desc>
          </value>
          <value>
            <val> 4 </val>
         <desc> LAT </desc>
          </value>
          <value>
            <val> 5 </val>
         <desc> X25-PAD </desc>
          </value>
          <value>
            <val> 6 </val>
         <desc> X25-T3POS </desc>
          </value>
          <value>
            <val> 8 </val>
         <desc>
              TCP Clear Quiet (supresses any NAS-generated
              connect string)
            </desc>
          </value>

        </avp>

        <avp code="16" may-encrypt="true"
         vendor-flag="disallowed" mandatory-flag="required">
          <name> Login-TCP-Port </name>
          <type> Unsigned32 </type>
        </avp>

        <avp code="18" may-encrypt="true"
         vendor-flag="disallowed" mandatory-flag="required">
          <name> Reply-Message </name>
          <type printable="true"> OctetString </type>
        </avp>

        <avp code="19" may-encrypt="true"
         vendor-flag="disallowed" mandatory-flag="required">
          <name> Callback-Number </name>
          <type printable="true"> OctetString </type>
        </avp>

        <avp code="20" may-encrypt="true"
         vendor-flag="disallowed" mandatory-flag="required">
          <name> Callback-Id </name>
          <type printable="true"> OctetString </type>
        </avp>
```

```
<avp code="22" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Framed-IP-Route </name>
  <type printable="true"> OctetString </type>
</avp>

<avp code="23" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Framed-IPX-Route </name>
  <type printable="true"> OctetString </type>
</avp>

<avp code="28" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Idle-Timeout </name>
  <type> Unsigned32 </type>
</avp>

<avp code="30" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Called-Station-Id </name>
  <type printable="true"> OctetString </type>
</avp>

<avp code="31" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Calling-Station-Id </name>
  <type printable="true"> OctetString </type>
</avp>

<avp code="32" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> NAS-Identifier </name>
  <type printable="true"> OctetString </type>
</avp>

<avp code="34" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Login-LAT-Service </name>
  <type> Unsigned32 </type>
</avp>

<avp code="35" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Login-LAT-Node </name>
  <type printable="true"> OctetString </type>
</avp>
```

```
<avp code="36" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Login-LAT-Group </name>
  <type> OctetString </type>
</avp>

<avp code="37" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Framed-Appletalk-Link </name>
  <type> Unsigned32 </type>
</avp>

<avp code="38" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Framed-Appletalk-Network </name>
  <type> Unsigned32 </type>
</avp>

<avp code="39" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Framed-Appletalk-Zone</name>
  <type> OctetString </type>
</avp>

<avp code="60" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> CHAP-Challenge </name>
  <type> OctetString </type>
</avp>

<avp code="61" may-encrypt="true" constrained="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> NAS-Port-Type </name>
  <type> Unsigned32 </type>
  <value> <val> 0 </val> <desc> Async </desc> </value>
  <value> <val> 1 </val> <desc> Sync </desc>     </value>
  <value> <val> 2 </val> <desc> ISDN Sync </desc> </value>
  <value> <val> 3 </val> <desc> ISDN Async V.120 </desc> </value>
  <value> <val> 4 </val> <desc> ISDN Async V.110 </desc> </value>
  <value> <val> 5 </val> <desc> Virtual </desc> </value>
  <value> <val> 6 </val> <desc> PIAFS </desc> </value>
  <value>
    <val> 7 </val>
    <desc>  HDLC Clear Channel</desc>
  </value>
  <value> <val> 8 </val> <desc> X.25 </desc> </value>
  <value> <val> 9 </val> <desc> X.75 </desc> </value>
  <value> <val> 10 </val> <desc> G.3 Fax </desc> </value>
```

```
      <value>
        <val> 11 </val>
        <desc> </desc>
      </value>
      <value>
        <val> 12 </val>
        <desc> </desc>
      </value>
      <value>
        <val> 13 </val>
        <desc> </desc>
      </value>
      <value>
        <val> 14 </val>
        <desc> </desc>
      </value>
      <value> <val> 15 </val> <desc> Ethernet </desc> </value>
      <value> <val> 16 </val> <desc> xDSL </desc> </value>
      <value> <val> 17 </val> <desc> Cable </desc> </value>
      <value>
        <val> 18 </val>
        <desc> Wireless - Other </desc>
      </value>
      <value>
        <val> 19 </val>
        <desc> Wireless - IEEE 802.11  </desc>
      </value>
    </avp>

    <avp code="62" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> Port-Limit </name>
      <type> Unsigned32 </type>
    </avp>

    <avp code="63" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> Login-LAT-Port </name>
      <type printable="true"> OctetString </type>
    </avp>

    <avp code="TBD" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> Tunnel-Group </name>
      <type>
        <grouped>
          <member-name> Tunnel-Type </member-name>
          <member-name> Tunnel-Medium-Type </member-name>
```

```
          <member-name> Tunnel-Preference </member-name>
          <member-name> Tunnel-Client-Endpoint </member-name>
          <member-name> Tunnel-Server-Endpoint </member-name>
          <member-name> Tunnel-Password </member-name>
          <member-name> Tunnel-Private-Group-ID </member-name>
          <member-name> Tunnel-Assignment-Id </member-name>
          <member-name> Tunnel-Preference </member-name>
          <member-name> Tunnel-Client-Auth-ID </member-name>
          <member-name> Tunnel-Server-Auth-ID </member-name>
          </grouped>
      </type>
      <description>
        Tunnel AVPs are all accumulated into a Tunnel-Group.
        Each potential tunnel configuration is represented by
        a Tunnel-Group AVP.
      </description>
    </avp>

    <avp code="64" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> Tunnel-Type </name>
      <type> Unsigned32 </type>
      <value>
        <val> 1 </val>
     <desc> Point-to-Point Tunneling Protocol (PPTP) </desc>
      </value>
      <value>
        <val> 2 </val>
     <desc> Layer Two Forwarding (L2F) </desc>
      </value>
      <value>
        <val> 3 </val>
     <desc> Layer Two Tunneling Protocol (L2TP) </desc>
      </value>
      <value>
        <val> 4 </val>
     <desc> Ascend Tunnel Management Protocol (ATMP) </desc>
      </value>
      <value>
        <val> 5 </val>
     <desc>  Virtual Tunneling Protocol (VTP)</desc>
      </value>
      <value>
        <val> 6 </val>
     <desc>
          IP Authentication Header in the Tunnel-mode (AH)
        </desc>
      </value>
```

```
  <value>
    <val> 7 </val>
 <desc> IP-in-IP Encapsulation (IP-IP) </desc>
  </value>
  <value>
    <val> 8 </val>
 <desc> Minimal IP-in-IP Encapsulation (MIN-IP-IP) </desc>
  </value>
  <value>
    <val> 9 </val>
 <desc>
      IP Encapsulating Security Payload in the Tunnel-mode
    </desc>
  </value>
  <value>
    <val> 10 </val>
 <desc> Generic Route Encapsulation (GRE) </desc>
  </value>
  <value>
    <val> 11 </val>
 <desc> Bay Dial Virtual Services </desc>
  </value>
  <value>
    <val> 12 </val>
 <desc> IP-in-IP Tunneling </desc>
  </value>
</avp>

<avp code="65" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Tunnel-Medium-Type </name>
  <type> Unsigned32 </type>
</avp>

<avp code="66" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Tunnel-Client-Endpoint </name>
  <type printable="true"> OctetString </type>
</avp>

<avp code="67" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> Tunnel-Server-Endpoint </name>
  <type printable="true"> OctetString </type>
</avp>

<avp code="69" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
```

```
      <name> Tunnel-Password </name>
      <type printable="true"> OctetString </type>
    </avp>

    <avp code="81" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> Tunnel-Private-Group-ID</name>
      <type printable="true"> OctetString </type>
    </avp>

    <avp code="82" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> Tunnel-Assignment-Id </name>
      <type printable="true"> OctetString </type>
    </avp>

    <avp code="83" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> Tunnel-Preference </name>
      <type> Unsigned32 </type>
    </avp>

    <avp code="90" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> Tunnel-Client-Auth-ID </name>
      <type printable="true"> OctetString </type>
    </avp>

    <avp code="91" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> Tunnel-Server-Auth-ID </name>
      <type printable="true"> OctetString </type>
    </avp>

    <avp code="400" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> </name>
      <type printable="true" > OctetString </type>
    </avp>

    <avp code="401" may-encrypt="true" constrained="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> </name>
      <type> Unsigned32 </type>
      <value>
        <val> 1 </val>
        <desc> AUTHENTICATE_ONLY </desc>
      </value>
```

```
        <value>
          <val> 2 </val>
          <desc> AUTHORIZE_ONLY </desc>
        </value>
        <value>
          <val> 3 </val>
          <desc> AUTHORIZE_AUTHENTICATE </desc>
        </value>
      </avp>

      <avp code="402" may-encrypt="true" constrained="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> EAP-Payload </name>
        <type> OctetString </type>
      </avp>

    </extension>


    <!-- -------------------------------------------------------- -->

    <extension>

      <extname> DIAMETER Resource Management Extensions </extname>

      <doc>
         ftp://ftp.ietf.org/internet-drafts/
         draft-calhoun-diameter-res-mgmt-06.txt
      </doc>

      <avp code="500" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Query-Index </name>
        <type> Unsigned32 </type>
      </avp>

      <avp code="501" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Resource-Token </name>
        <type>
          <grouped>
            <member-name> Session-ID </member-name>
            <member-name> Host-Name </member-name>
            <member-name> User-Name </member-name>
            <member-name> Timestamp </member-name>
            <member-name> Extension-Id </member-name>
            <member-name> Resource-Bag </member-name>
          </grouped>
```

```
      </type>
    </avp>

    <avp code="TBD" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> Resource-Bag </name>
      <type> OctetString </type>
      <description>
        This AVP encapsulates arbitrary user data.  This could
        be in the form of a vendor specified AVP with a group
        data type, for example.
      </description>
    </avp>

  </extension>

  <!-- -------------------------------------------------------- -->

  <extension>

    <extname> DIAMETER Strong Security Extension </extname>

    <doc>
       ftp://ftp.ietf.org/internet-drafts/
       draft-calhoun-diameter-strong-crypto-05.txt
    </doc>

    <avp code="300" may-encrypt="false">
      <name> CMS-Data </name>
      <type> OctetString </type>
      <description>
        The value of this AVP is a CMS object encrypted according
        to the S/MIME v3 message specification. [RFC 2633]  The
        contents of encapsulating encrypted MIME is a set of
        DIAMETER AVPs.
      </description>
    </avp>

  </extension>

  <!-- -------------------------------------------------------- -->

  <extension>

    <extname> DIAMETER Mobile IP Extensions </extname>

    <doc>
       ftp://ftp.ietf.org/internet-drafts/
```

```
      draft-calhoun-diameter-mobileip-11.txt
    </doc>

    <avp code="320" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> MIP-Reg-Request </name>
      <type> OctetString </type>
    </avp>

    <avp code="321" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> MIP-Reg-Reply </name>
      <type> OctetString </type>
    </avp>

    <avp code="322" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> MN-AAA-Auth </name>
      <type>
        <group>
          <member-name> MN-AAA-SPI </member-name>
          <member-name> Authentication-Input-Data-Length </member-name>
          <member-name> Authenticator-Length </member-name>
          <member-name> Authenticator-Offset </member-name>
        </group>
      </type>
    </avp>

    <avp code="TBD" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> MN-AAA-SPI </name>
      <type> Unsigned32 </type>
      <description>
         The SPI which indicates the algorithm by which the targeted
         AAA server (AAAH) should attempt to validate the
         Authenticator computed by the mobile node over the
         Registration Request data
      </description>
    </avp>

    <avp code="TBD" may-encrypt="true"
     vendor-flag="disallowed" mandatory-flag="required">
      <name> Authentication-Input-Data-Length </name>
      <type> Unsigned32 </type>
      <description>
         The length in bytes of the Registration Request data (data
         portion of MIP-Reg-Request AVP) that should be used as
         input to the algorithm (indicated by the MN-AAA SPI) used
```

```
            to determine whether the Authenticator Data supplied by the
            Mobile Node is valid.
         </description>
       </avp>

       <avp code="TBD" may-encrypt="true"
        vendor-flag="disallowed" mandatory-flag="required">
         <name> Authenticator-Length </name>
         <type> Unsigned32 </type>
         <description>
            The length of the authenticator to be validated by the
            targeted AAA server (i.e., AAAH).
         </description>
       </avp>

       <avp code="TBD" may-encrypt="true"
        vendor-flag="disallowed" mandatory-flag="required">
         <name> Authenticator-Offset </name>
         <type> Unsigned32 </type>
         <description>
            The offset into the Registration Request Data, of the
            authenticator to be validated by the targeted AAA server
            (i.e., AAAH).
        </description>
       </avp>


       <avp code="325" may-encrypt="true"
        vendor-flag="disallowed" mandatory-flag="required">
         <name> MN-to-FA-Key </name>
         <type>
           <grouped>
             <member-name> Peer-SPI </member-name>
             <member-name> Mobility-SA-Key </member-name>
           </grouped>
         </type>
         <description>
            The MN-to-FA-Key AVP contains the data immediately
            following the Mobile IP extension header of the
            "Unsolicited MN-FA Key From AAA Subtype", as documented
            in draft-calhoun-mobileip-aaa-key-01.txt.
         </description>
       </avp>

       <avp code="331" may-encrypt="true"
        vendor-flag="disallowed" mandatory-flag="required">
         <name> MN-to-HA-Key </name>
         <type>
```

```
    <grouped>
      <member-name>  Peer-SPI </member-name>
      <member-name>  Mobility-SA-Key </member-name>
    </grouped>
  </type>
  <description>
     The HA-to-MN-Key AVP contains the data immediately
     following the Mobile IP extension header of the
     "Unsolicited MN-HA Key From AAA Subtype", as documented
     in draft-calhoun-mobileip-aaa-key-01.txt.
  </description>
</avp>

<avp code="326" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> FA-to-MN-Key </name>
  <type>
    <grouped>
      <member-name> Peer-SPI  </member-name>
      <member-name> Mobility-SA-Key </member-name>
    </grouped>
  </type>
</avp>

<avp code="328" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> FA-to-HA-Key </name>
  <type>
    <grouped>
      <member-name> Peer-SPI  </member-name>
      <member-name> Mobility-SA-Key </member-name>
    </grouped>
  </type>
</avp>

<avp code="332" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
  <name> HA-to-MN-Key </name>
  <type>
    <grouped>
      <member-name> Peer-SPI  </member-name>
      <member-name> Mobility-SA-Key </member-name>
    </grouped>
  </type>
</avp>

<avp code="329" may-encrypt="true"
 vendor-flag="disallowed" mandatory-flag="required">
```

```
            <name> HA-to-FA-Key </name>
            <type>
              <grouped>
                <member-name> Peer-SPI  </member-name>
                <member-name> </member-name>
              </grouped>
            </type>
          </avp>

          <avp code="TBD" may-encrypt="true"
           vendor-flag="disallowed" mandatory-flag="required">
            <name> Peer-SPI </name>
            <type> Unsigned32 </type>
            <description>
               A 32-bit opaque value, which the target MUST use to index
               all the necessary information recovered from the security
               information after it is decoded.
            </description>
          </avp>

          <avp code="TBD" may-encrypt="true"
           vendor-flag="disallowed" mandatory-flag="required">
            <name> Mobility-SA-Key </name>
            <type> OctetString </type>
            <description>
              This contains the key used to create a Mobility Security
              Association between the mobility nodes.
            </description>
          </avp>

          <avp code="324" may-encrypt="true"
           vendor-flag="disallowed" mandatory-flag="required">
            <name> FA-MN-Preferred-SPI </name>
            <type> Unsigned32 </type>
            <description>
            </description>
          </avp>

          <avp code="327" may-encrypt="true"
           vendor-flag="disallowed" mandatory-flag="required">
            <name> FA-HA-Preferred-SPI </name>
            <type> Unsigned32 </type>
            <description>
            </description>
          </avp>

          <avp code="TBD" may-encrypt="true"
           vendor-flag="disallowed" mandatory-flag="required">
```

```
          <name> </name>
          <type> </type>
          <description>
          </description>
        </avp>

        <avp code="333" may-encrypt="true"
         vendor-flag="disallowed" mandatory-flag="required">
          <name> Mobile-Node-Address </name>
          <type ip-address="true"> OctetString </type>
        </avp>

        <avp code="334" may-encrypt="true"
         vendor-flag="disallowed" mandatory-flag="required">
          <name> Home-Agent-Address </name>
          <type ip-address="true"> OctetString </type>
        </avp>

        <avp code="335" may-encrypt="true"
         vendor-flag="disallowed" mandatory-flag="required">
          <name> Previous-FA-NAI </name>
          <type printable="true"> OctetString </type>
        </avp>

        <avp code="336" may-encrypt="true"
         vendor-flag="disallowed" mandatory-flag="required">
          <name> Previous-FA-Addr </name>
          <type ip-address="true"> OctetString </type>
        </avp>

        <avp code="337" may-encrypt="true" constrained="false"
         vendor-flag="disallowed" mandatory-flag="required">
          <name> MIP-Feature-Vector </name>
          <type> Unsigned32 </type>
          <description>
            The value of this AVP is added with flag values set
         by the Foreign Agent or by the AAAF owned by the
         same administrative domain as the Foreign Agent.
          </description>
          <value>
            <val> 1 </val>
         <desc> Mobile-Node-Home-Address-Requested </desc>
          </value>
          <value>
            <val> 2 </val>
         <desc> Home-Address-Allocatable-Only-in-Home-Domain </desc>
          </value>
          <value>
```

```
      <val> 4 </val>
   <desc> Home-Agent-Requested </desc>
    </value>
    <value>
      <val> 8 </val>
   <desc> Foreign-Home-Agent-Available </desc>
    </value>
    <value>
      <val> 16 </val>
   <desc> MN-HA-Key-Request </desc>
    </value>
    <value>
      <val> 32 </val>
   <desc> MN-FA-Key-Request </desc>
    </value>
    <value>
      <val> 64 </val>
   <desc> FA-HA-Key-Request </desc>
    </value>
    <value>
      <val> 128 </val>
   <desc> Home-Agent-In-Foreign-Network </desc>
    </value>
  </avp>

</extension>

<!-- ------------------------------------------------------- -->

<extension>

  <extname> DIAMETER Accounting Extension </extname>

  <doc>
     ftp://ftp.ietf.org/internet-drafts/
     draft-calhoun-diameter-accounting-08.txt
  </doc>

  <avp code="480" may-encrypt="true" constrained="true"
   vendor-flag="disallowed" mandatory-flag="required">
    <name> Accounting-Record-Type </name>
    <type> Unsigned32 </type>
    <value>
      <val> 1 </val>
   <desc>
     EVENT_RECORD
        An Accounting Event Record is used to indicate
     that a one-time event has occurred (meaning that
```

```
          the start and end of the event are simultaneous).
          This record contains all information relevant to
          the service, and is the only record of the service.
           </desc>
         </value>
         <value>
           <val> 2 </val>
        <desc>
             START_RECORD
             An Accounting Start, Interim, and Stop Records are
          used to indicate that a service of a measurable
          length has been given.  An Accounting Start Record
          is used to initiate an accounting session, and
          contains accounting information that is relevant
             to the initiation of the session.
           </desc>
         </value>
         <value>
           <val> 3 </val>
        <desc>
             INTERIM_RECORD
             An Accounting Stop Record is sent to terminate
          an accounting session and contains cumulative
          accounting information relevant to the existing
          session.
           </desc>
         </value>
         <value>
           <val> 4 </val>
        <desc>
             STOP_RECORD
             An Accounting Stop Record is sent to terminate an
          accounting session and contains cumulative accounting
          information relevant to the existing session.
           </desc>
         </value>
       </avp>

       <avp code="481" may-encrypt="true"
        vendor-flag="disallowed" mandatory-flag="required">
         <name> ADIF-Record </name>
         <type> OctetString </type>
       </avp>

       <avp code="482" may-encrypt="true"
        vendor-flag="disallowed" mandatory-flag="required">
         <name> Accounting-Interim-Interval </name>
         <type> Unsigned32 </type>
```

```
      </avp>

      <avp code="483" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Accounting-Delivery-Max-Batch </name>
        <type> Unsigned32 </type>
      </avp>

      <avp code="484" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Accounting-Delivery-Max-Delay </name>
        <type> Unsigned32 </type>
      </avp>

      <avp code="485" may-encrypt="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Accounting-Record-Number </name>
        <type> Unsigned32 </type>
      </avp>

      <avp code="486" may-encrypt="true" constrained="true"
       vendor-flag="disallowed" mandatory-flag="required">
        <name> Accounting-State </name>
        <type> Unsigned32 </type>
        <value> <val> 1 </val> <desc> ENABLED </desc> </value>
        <value> <val> 2 </val> <desc> DISABLED </desc> </value>
      </avp>

    </extension>

  </diameter>
```