Network Working Group   P. Calhoun
Request for Comments: 3588   Airespace, Inc.
Category: Standards Track   J. Loughney
                                                                Nokia
                                                            E. Guttman
                                                Sun Microsystems, Inc.
                                                              G. Zorn
                                                   Cisco Systems, Inc.
                                                             J. Arkko
                                                             Ericsson
                                                       September 2003

Diameter Base Protocol

Status of this Memo

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

Abstract

   The Diameter base protocol is intended to provide an Authentication,
   Authorization and Accounting (AAA) framework for applications such as
   network access or IP mobility. Diameter is also intended to work in
   both local Authentication, Authorization & Accounting and roaming
   situations. This document specifies the message format, transport,
   error reporting, accounting and security services to be used by all
   Diameter applications. The Diameter base application needs to be
   supported by all Diameter implementations.

Conventions Used In This Document

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in BCP 14, RFC 2119
   [KEYWORD].

Table of Contents

1. Introduction

   Authentication, Authorization and Accounting (AAA) protocols such as
   TACACS [TACACS] and RADIUS [RADIUS] were initially deployed to
   provide dial-up PPP [PPP] and terminal server access. Over time,
   with the growth of the Internet and the introduction of new access
   technologies, including wireless, DSL, Mobile IP and Ethernet,
   routers and network access servers (NAS) have increased in complexity
   and density, putting new demands on AAA protocols.

   Network access requirements for AAA protocols are summarized in
   [AAAREQ]. These include:

   Failover
      [RADIUS] does not define failover mechanisms, and as a result,
      failover behavior differs between implementations. In order to
      provide well defined failover behavior, Diameter supports
      application-layer acknowledgements, and defines failover
      algorithms and the associated state machine. This is described in
      Section 5.5 and [AAATRANS].

   Transmission-level security
      [RADIUS] defines an application-layer authentication and integrity
      scheme that is required only for use with Response packets. While
      [RADEXT] defines an additional authentication and integrity
      mechanism, use is only required during Extensible Authentication
      Protocol (EAP) sessions. While attribute-hiding is supported,
      [RADIUS] does not provide support for per-packet confidentiality.
      In accounting, [RADACCT] assumes that replay protection is
      provided by the backend billing server, rather than within the
      protocol itself.

      While [RFC3162] defines the use of IPsec with RADIUS, support for
      IPsec is not required. Since within [IKE] authentication occurs
      only within Phase 1 prior to the establishment of IPsec SAs in
      Phase 2, it is typically not possible to define separate trust or
      authorization schemes for each application. This limits the
      usefulness of IPsec in inter-domain AAA applications (such as
      roaming) where it may be desirable to define a distinct
      certificate hierarchy for use in a AAA deployment. In order to
      provide universal support for transmission-level security, and
      enable both intra- and inter-domain AAA deployments, IPsec support
      is mandatory in Diameter, and TLS support is optional. Security
      is discussed in Section 13.

Reliable transport
   RADIUS runs over UDP, and does not define retransmission behavior;
   as a result, reliability varies between implementations. As
   described in [ACCMGMT], this is a major issue in accounting, where
   packet loss may translate directly into revenue loss. In order to
   provide well defined transport behavior, Diameter runs over
   reliable transport mechanisms (TCP, SCTP) as defined in
   [AAATRANS].

Agent support
   [RADIUS] does not provide for explicit support for agents,
   including Proxies, Redirects and Relays. Since the expected
   behavior is not defined, it varies between implementations.
   Diameter defines agent behavior explicitly; this is described in
   Section 2.8.

Server-initiated messages
   While RADIUS server-initiated messages are defined in [DYNAUTH],
   support is optional. This makes it difficult to implement
   features such as unsolicited disconnect or
   reauthentication/reauthorization on demand across a heterogeneous
   deployment. Support for server-initiated messages is mandatory in
   Diameter, and is described in Section 8.

Auditability
   RADIUS does not define data-object security mechanisms, and as a
   result, untrusted proxies may modify attributes or even packet
   headers without being detected. Combined with lack of support for
   capabilities negotiation, this makes it very difficult to
   determine what occurred in the event of a dispute. While
   implementation of data object security is not mandatory within
   Diameter, these capabilities are supported, and are described in
   [AAACMS].

Transition support
   While Diameter does not share a common protocol data unit (PDU)
   with RADIUS, considerable effort has been expended in enabling
   backward compatibility with RADIUS, so that the two protocols may
   be deployed in the same network. Initially, it is expected that
   Diameter will be deployed within new network devices, as well as
   within gateways enabling communication between legacy RADIUS
   devices and Diameter agents. This capability, described in
   [NASREQ], enables Diameter support to be added to legacy networks,
   by addition of a gateway or server speaking both RADIUS and
   Diameter.

   In addition to addressing the above requirements, Diameter also
   provides support for the following:

   Capability negotiation
      RADIUS does not support error messages, capability negotiation, or
      a mandatory/non-mandatory flag for attributes. Since RADIUS
      clients and servers are not aware of each other's capabilities,
      they may not be able to successfully negotiate a mutually
      acceptable service, or in some cases, even be aware of what
      service has been implemented. Diameter includes support for error
      handling (Section 7), capability negotiation (Section 5.3), and
      mandatory/non-mandatory attribute-value pairs (AVPs) (Section
      4.1).

   Peer discovery and configuration
      RADIUS implementations typically require that the name or address
      of servers or clients be manually configured, along with the
      corresponding shared secrets. This results in a large
      administrative burden, and creates the temptation to reuse the
      RADIUS shared secret, which can result in major security
      vulnerabilities if the Request Authenticator is not globally and
      temporally unique as required in [RADIUS]. Through DNS, Diameter
      enables dynamic discovery of peers. Derivation of dynamic session
      keys is enabled via transmission-level security.

   Roaming support
      The ROAMOPS WG provided a survey of roaming implementations
      [ROAMREV], detailed roaming requirements [ROAMCRIT], defined the
      Network Access Identifier (NAI) [NAI], and documented existing
      implementations (and imitations) of RADIUS-based roaming
      [PROXYCHAIN]. In order to improve scalability, [PROXYCHAIN]
      introduced the concept of proxy chaining via an intermediate
      server, facilitating roaming between providers. However, since
      RADIUS does not provide explicit support for proxies, and lacks
      auditability and transmission-level security features, RADIUS-
      based roaming is vulnerable to attack from external parties as
      well as susceptible to fraud perpetrated by the roaming partners
      themselves. As a result, it is not suitable for wide-scale
      deployment on the Internet [PROXYCHAIN]. By providing explicit
      support for inter-domain roaming and message routing (Sections 2.7
      and 6), auditability [AAACMS], and transmission-layer security
      (Section 13) features, Diameter addresses these limitations and
      provides for secure and scalable roaming.

   In the decade since AAA protocols were first introduced, the
   capabilities of Network Access Server (NAS) devices have increased
   substantially. As a result, while Diameter is a considerably more
   sophisticated protocol than RADIUS, it remains feasible to implement

within embedded devices, given improvements in processor speeds and
the widespread availability of embedded IPsec and TLS
implementations.

1.1. Diameter Protocol

   The Diameter base protocol provides the following facilities:

   - Delivery of AVPs (attribute value pairs)
   - Capabilities negotiation
   - Error notification
   - Extensibility, through addition of new commands and AVPs (required
      in [AAAREQ]).
   - Basic services necessary for applications, such as handling of
      user sessions or accounting

   All data delivered by the protocol is in the form of an AVP. Some of
   these AVP values are used by the Diameter protocol itself, while
   others deliver data associated with particular applications that
   employ Diameter. AVPs may be added arbitrarily to Diameter messages,
   so long as the required AVPs are included and AVPs that are
   explicitly excluded are not included. AVPs are used by the base
   Diameter protocol to support the following required features:

   - Transporting of user authentication information, for the purposes
      of enabling the Diameter server to authenticate the user.

   - Transporting of service specific authorization information,
      between client and servers, allowing the peers to decide whether a
      user's access request should be granted.

   - Exchanging resource usage information, which MAY be used for
      accounting purposes, capacity planning, etc.

   - Relaying, proxying and redirecting of Diameter messages through a
      server hierarchy.

   The Diameter base protocol provides the minimum requirements needed
   for a AAA protocol, as required by [AAAREQ]. The base protocol may
   be used by itself for accounting purposes only, or it may be used
   with a Diameter application, such as Mobile IPv4 [DIAMMIP], or
   network access [NASREQ]. It is also possible for the base protocol
   to be extended for use in new applications, via the addition of new
   commands or AVPs. At this time the focus of Diameter is network
   access and accounting applications. A truly generic AAA protocol
   used by many applications might provide functionality not provided by
   Diameter. Therefore, it is imperative that the designers of new
   applications understand their requirements before using Diameter.

See Section 2.4 for more information on Diameter applications.

Any node can initiate a request. In that sense, Diameter is a peer-
to-peer protocol. In this document, a Diameter Client is a device at
the edge of the network that performs access control, such as a
Network Access Server (NAS) or a Foreign Agent (FA). A Diameter
client generates Diameter messages to request authentication,
authorization, and accounting services for the user. A Diameter
agent is a node that does not authenticate and/or authorize messages
locally; agents include proxies, redirects and relay agents. A
Diameter server performs authentication and/or authorization of the
user. A Diameter node MAY act as an agent for certain requests while
acting as a server for others.

The Diameter protocol also supports server-initiated messages, such
as a request to abort service to a particular user.

1.1.1. Description of the Document Set

Currently, the Diameter specification consists of a base
specification (this document), Transport Profile [AAATRANS] and
applications: Mobile IPv4 [DIAMMIP], and NASREQ [NASREQ].

The Transport Profile document [AAATRANS] discusses transport layer
issues that arise with AAA protocols and recommendations on how to
overcome these issues. This document also defines the Diameter
failover algorithm and state machine.

The Mobile IPv4 [DIAMMIP] application defines a Diameter application
that allows a Diameter server to perform AAA functions for Mobile
IPv4 services to a mobile node.

The NASREQ [NASREQ] application defines a Diameter Application that
allows a Diameter server to be used in a PPP/SLIP Dial-Up and
Terminal Server Access environment. Consideration was given for
servers that need to perform protocol conversion between Diameter and
RADIUS.

In summary, this document defines the base protocol specification for
AAA, which includes support for accounting. The Mobile IPv4 and the
NASREQ documents describe applications that use this base
specification for Authentication, Authorization and Accounting.

1.2. Approach to Extensibility

   The Diameter protocol is designed to be extensible, using several
   mechanisms, including:

      - Defining new AVP values
      - Creating new AVPs
      - Creating new authentication/authorization applications
      - Creating new accounting applications
      - Application authentication procedures

   Reuse of existing AVP values, AVPs and Diameter applications are
   strongly recommended. Reuse simplifies standardization and
   implementation and avoids potential interoperability issues. It is
   expected that command codes are reused; new command codes can only be
   created by IETF Consensus (see Section 11.2.1).

1.2.1. Defining New AVP Values

   New applications should attempt to reuse AVPs defined in existing
   applications when possible, as opposed to creating new AVPs. For
   AVPs of type Enumerated, an application may require a new value to
   communicate some service-specific information.

   In order to allocate a new AVP value, a request MUST be sent to IANA
   [IANA], along with an explanation of the new AVP value. IANA
   considerations for Diameter are discussed in Section 11.

1.2.2. Creating New AVPs

   When no existing AVP can be used, a new AVP should be created. The
   new AVP being defined MUST use one of the data types listed in
   Section 4.2.

   In the event that a logical grouping of AVPs is necessary, and
   multiple "groups" are possible in a given command, it is recommended
   that a Grouped AVP be used (see Section 4.4).

   In order to create a new AVP, a request MUST be sent to IANA, with a
   specification for the AVP. The request MUST include the commands
   that would make use of the AVP.

1.2.3. Creating New Authentication Applications

   Every Diameter application specification MUST have an IANA assigned
   Application Identifier (see Section 2.4) or a vendor specific
   Application Identifier.

   Should a new Diameter usage scenario find itself unable to fit within
   an existing application without requiring major changes to the
   specification, it may be desirable to create a new Diameter
   application. Major changes to an application include:

   - Adding new AVPs to the command, which have the "M" bit set.

   - Requiring a command that has a different number of round trips to
      satisfy a request (e.g., application foo has a command that
      requires one round trip, but new application bar has a command
      that requires two round trips to complete).

   - Adding support for an authentication method requiring definition
      of new AVPs for use with the application. Since a new EAP
      authentication method can be supported within Diameter without
      requiring new AVPs, addition of EAP methods does not require the
      creation of a new authentication application.

   Creation of a new application should be viewed as a last resort. An
   implementation MAY add arbitrary non-mandatory AVPs to any command
   defined in an application, including vendor-specific AVPs without
   needing to define a new application. Please refer to Section 11.1.1
   for details.

   In order to justify allocation of a new application identifier,
   Diameter applications MUST define one Command Code, or add new
   mandatory AVPs to the ABNF.

   The expected AVPs MUST be defined in an ABNF [ABNF] grammar (see
   Section 3.2). If the Diameter application has accounting
   requirements, it MUST also specify the AVPs that are to be present in
   the Diameter Accounting messages (see Section 9.3). However, just
   because a new authentication application id is required, does not
   imply that a new accounting application id is required.

   When possible, a new Diameter application SHOULD reuse existing
   Diameter AVPs, in order to avoid defining multiple AVPs that carry
   similar information.

1.2.4. Creating New Accounting Applications

   There are services that only require Diameter accounting. Such
   services need to define the AVPs carried in the Accounting-Request
   (ACR)/ Accounting-Answer (ACA) messages, but do not need to define
   new command codes. An implementation MAY add arbitrary non-mandatory
   AVPs (AVPs with the "M" bit not set) to any command defined in an

application, including vendor-specific AVPs, without needing to
define a new accounting application. Please refer to Section 11.1.1
for details.

Application Identifiers are still required for Diameter capability
exchange. Every Diameter accounting application specification MUST
have an IANA assigned Application Identifier (see Section 2.4) or a
vendor specific Application Identifier.

Every Diameter implementation MUST support accounting. Basic
accounting support is sufficient to handle any application that uses
the ACR/ACA commands defined in this document, as long as no new
mandatory AVPs are added. A mandatory AVP is defined as one which
has the "M" bit set when sent within an accounting command,
regardless of whether it is required or optional within the ABNF for
the accounting application.

The creation of a new accounting application should be viewed as a
last resort and MUST NOT be used unless a new command or additional
mechanisms (e.g., application defined state machine) is defined
within the application, or new mandatory AVPs are added to the ABNF.

Within an accounting command, setting the "M" bit implies that a
backend server (e.g., billing server) or the accounting server itself
MUST understand the AVP in order to compute a correct bill. If the
AVP is not relevant to the billing process, when the AVP is included
within an accounting command, it MUST NOT have the "M" bit set, even
if the "M" bit is set when the same AVP is used within other Diameter
commands (i.e., authentication/authorization commands).

A DIAMETER base accounting implementation MUST be configurable to
advertise supported accounting applications in order to prevent the
accounting server from accepting accounting requests for unbillable
services. The combination of the home domain and the accounting
application Id can be used in order to route the request to the
appropriate accounting server.

When possible, a new Diameter accounting application SHOULD attempt
to reuse existing AVPs, in order to avoid defining multiple AVPs that
carry similar information.

If the base accounting is used without any mandatory AVPs, new
commands or additional mechanisms (e.g., application defined state
machine), then the base protocol defined standard accounting
application Id (Section 2.4) MUST be used in ACR/ACA commands.

1.2.5. Application Authentication Procedures

   When possible, applications SHOULD be designed such that new
   authentication methods MAY be added without requiring changes to the
   application. This MAY require that new AVP values be assigned to
   represent the new authentication transform, or any other scheme that
   produces similar results. When possible, authentication frameworks,
   such as Extensible Authentication Protocol [EAP], SHOULD be used.

1.3. Terminology

   AAA
      Authentication, Authorization and Accounting.

   Accounting
      The act of collecting information on resource usage for the
      purpose of capacity planning, auditing, billing or cost
      allocation.

   Accounting Record
      An accounting record represents a summary of the resource
      consumption of a user over the entire session. Accounting servers
      creating the accounting record may do so by processing interim
      accounting events or accounting events from several devices
      serving the same user.

   Authentication
      The act of verifying the identity of an entity (subject).

   Authorization
      The act of determining whether a requesting entity (subject) will
      be allowed access to a resource (object).

   AVP
      The Diameter protocol consists of a header followed by one or more
      Attribute-Value-Pairs (AVPs). An AVP includes a header and is
      used to encapsulate protocol-specific data (e.g., routing
      information) as well as authentication, authorization or
      accounting information.

   Broker
      A broker is a business term commonly used in AAA infrastructures.
      A broker is either a relay, proxy or redirect agent, and MAY be
      operated by roaming consortiums. Depending on the business model,
      a broker may either choose to deploy relay agents or proxy
      agents.

Diameter Agent
   A Diameter Agent is a Diameter node that provides either relay,
   proxy, redirect or translation services.

Diameter Client
   A Diameter Client is a device at the edge of the network that
   performs access control. An example of a Diameter client is a
   Network Access Server (NAS) or a Foreign Agent (FA).

Diameter Node
   A Diameter node is a host process that implements the Diameter
   protocol, and acts either as a Client, Agent or Server.

Diameter Peer
   A Diameter Peer is a Diameter Node to which a given Diameter Node
   has a direct transport connection.

Diameter Security Exchange
   A Diameter Security Exchange is a process through which two
   Diameter nodes establish end-to-end security.

Diameter Server
   A Diameter Server is one that handles authentication,
   authorization and accounting requests for a particular realm. By
   its very nature, a Diameter Server MUST support Diameter
   applications in addition to the base protocol.

Downstream
   Downstream is used to identify the direction of a particular
   Diameter message from the home server towards the access device.

End-to-End Security
   TLS and IPsec provide hop-by-hop security, or security across a
   transport connection. When relays or proxy are involved, this
   hop-by-hop security does not protect the entire Diameter user
   session. End-to-end security is security between two Diameter
   nodes, possibly communicating through Diameter Agents. This
   security protects the entire Diameter communications path from the
   originating Diameter node to the terminating Diameter node.

Home Realm
   A Home Realm is the administrative domain with which the user
   maintains an account relationship.

Home Server
   See Diameter Server.

   Interim accounting
      An interim accounting message provides a snapshot of usage during
      a user's session. It is typically implemented in order to provide
      for partial accounting of a user's session in the case of a device
      reboot or other network problem prevents the reception of a
      session summary message or session record.

   Local Realm
      A local realm is the administrative domain providing services to a
      user. An administrative domain MAY act as a local realm for
      certain users, while being a home realm for others.

   Multi-session
      A multi-session represents a logical linking of several sessions.
      Multi-sessions are tracked by using the Acct-Multi-Session-Id. An
      example of a multi-session would be a Multi-link PPP bundle. Each
      leg of the bundle would be a session while the entire bundle would
      be a multi-session.

   Network Access Identifier
      The Network Access Identifier, or NAI [NAI], is used in the
      Diameter protocol to extract a user's identity and realm. The
      identity is used to identify the user during authentication and/or
      authorization, while the realm is used for message routing
      purposes.

   Proxy Agent or Proxy
      In addition to forwarding requests and responses, proxies make
      policy decisions relating to resource usage and provisioning.
      This is typically accomplished by tracking the state of NAS
      devices. While proxies typically do not respond to client
      Requests prior to receiving a Response from the server, they may
      originate Reject messages in cases where policies are violated.
      As a result, proxies need to understand the semantics of the
      messages passing through them, and may not support all Diameter
      applications.

   Realm
      The string in the NAI that immediately follows the '@' character.
      NAI realm names are required to be unique, and are piggybacked on
      the administration of the DNS namespace. Diameter makes use of
      the realm, also loosely referred to as domain, to determine
      whether messages can be satisfied locally, or whether they must be
      routed or redirected. In RADIUS, realm names are not necessarily
      piggybacked on the DNS namespace but may be independent of it.

Real-time Accounting
   Real-time accounting involves the processing of information on
   resource usage within a defined time window. Time constraints are
   typically imposed in order to limit financial risk.

Relay Agent or Relay
   Relays forward requests and responses based on routing-related
   AVPs and realm routing table entries. Since relays do not make
   policy decisions, they do not examine or alter non-routing AVPs.
   As a result, relays never originate messages, do not need to
   understand the semantics of messages or non-routing AVPs, and are
   capable of handling any Diameter application or message type.
   Since relays make decisions based on information in routing AVPs
   and realm forwarding tables they do not keep state on NAS resource
   usage or sessions in progress.

Redirect Agent
   Rather than forwarding requests and responses between clients and
   servers, redirect agents refer clients to servers and allow them
   to communicate directly. Since redirect agents do not sit in the
   forwarding path, they do not alter any AVPs transiting between
   client and server. Redirect agents do not originate messages and
   are capable of handling any message type, although they may be
   configured only to redirect messages of certain types, while
   acting as relay or proxy agents for other types. As with proxy
   agents, redirect agents do not keep state with respect to sessions
   or NAS resources.

Roaming Relationships
   Roaming relationships include relationships between companies and
   ISPs, relationships among peer ISPs within a roaming consortium,
   and relationships between an ISP and a roaming consortium.

Security Association
   A security association is an association between two endpoints in
   a Diameter session which allows the endpoints to communicate with
   integrity and confidentially, even in the presence of relays
   and/or proxies.

Session
   A session is a related progression of events devoted to a
   particular activity. Each application SHOULD provide guidelines
   as to when a session begins and ends. All Diameter packets with
   the same Session-Identifier are considered to be part of the same
   session.

Session state
     A stateful agent is one that maintains session state information,
     by keeping track of all authorized active sessions. Each
     authorized session is bound to a particular service, and its state
     is considered active either until it is notified otherwise, or by
     expiration.

Sub-session
     A sub-session represents a distinct service (e.g., QoS or data
     characteristics) provided to a given session. These services may
     happen concurrently (e.g., simultaneous voice and data transfer
     during the same session) or serially. These changes in sessions
     are tracked with the Accounting-Sub-Session-Id.

Transaction state
     The Diameter protocol requires that agents maintain transaction
     state, which is used for failover purposes. Transaction state
     implies that upon forwarding a request, the Hop-by-Hop identifier
     is saved; the field is replaced with a locally unique identifier,
     which is restored to its original value when the corresponding
     answer is received. The request's state is released upon receipt
     of the answer. A stateless agent is one that only maintains
     transaction state.

Translation Agent
     A translation agent is a stateful Diameter node that performs
     protocol translation between Diameter and another AAA protocol,
     such as RADIUS.

Transport Connection
     A transport connection is a TCP or SCTP connection existing
     directly between two Diameter peers, otherwise known as a Peer-
     to-Peer Connection.

Upstream
     Upstream is used to identify the direction of a particular
     Diameter message from the access device towards the home server.

User
     The entity requesting or using some resource, in support of which
     a Diameter client has generated a request.

2. Protocol Overview

   The base Diameter protocol may be used by itself for accounting
   applications, but for use in authentication and authorization it is
   always extended for a particular application. Two Diameter
   applications are defined by companion documents: NASREQ [NASREQ],

Mobile IPv4 [DIAMMIP]. These applications are introduced in this
document but specified elsewhere. Additional Diameter applications
MAY be defined in the future (see Section 11.3).

Diameter Clients MUST support the base protocol, which includes
accounting. In addition, they MUST fully support each Diameter
application that is needed to implement the client's service, e.g.,
NASREQ and/or Mobile IPv4. A Diameter Client that does not support
both NASREQ and Mobile IPv4, MUST be referred to as "Diameter X
Client" where X is the application which it supports, and not a
"Diameter Client".

Diameter Servers MUST support the base protocol, which includes
accounting. In addition, they MUST fully support each Diameter
application that is needed to implement the intended service, e.g.,
NASREQ and/or Mobile IPv4. A Diameter Server that does not support
both NASREQ and Mobile IPv4, MUST be referred to as "Diameter X
Server" where X is the application which it supports, and not a
"Diameter Server".

Diameter Relays and redirect agents are, by definition, protocol
transparent, and MUST transparently support the Diameter base
protocol, which includes accounting, and all Diameter applications.

Diameter proxies MUST support the base protocol, which includes
accounting. In addition, they MUST fully support each Diameter
application that is needed to implement proxied services, e.g.,
NASREQ and/or Mobile IPv4. A Diameter proxy which does not support
also both NASREQ and Mobile IPv4, MUST be referred to as "Diameter X
Proxy" where X is the application which it supports, and not a
"Diameter Proxy".

The base Diameter protocol concerns itself with capabilities
negotiation, how messages are sent and how peers may eventually be
abandoned. The base protocol also defines certain rules that apply
to all exchanges of messages between Diameter nodes.

Communication between Diameter peers begins with one peer sending a
message to another Diameter peer. The set of AVPs included in the
message is determined by a particular Diameter application. One AVP
that is included to reference a user's session is the Session-Id.

The initial request for authentication and/or authorization of a user
would include the Session-Id. The Session-Id is then used in all
subsequent messages to identify the user's session (see Section 8 for
more information). The communicating party may accept the request,
or reject it by returning an answer message with the Result-Code AVP

set to indicate an error occurred. The specific behavior of the
Diameter server or client receiving a request depends on the Diameter
application employed.

Session state (associated with a Session-Id) MUST be freed upon
receipt of the Session-Termination-Request, Session-Termination-
Answer, expiration of authorized service time in the Session-Timeout
AVP, and according to rules established in a particular Diameter
application.

2.1. Transport

Transport profile is defined in [AAATRANS].

The base Diameter protocol is run on port 3868 of both TCP [TCP] and
SCTP [SCTP] transport protocols.

Diameter clients MUST support either TCP or SCTP, while agents and
servers MUST support both. Future versions of this specification MAY
mandate that clients support SCTP.

A Diameter node MAY initiate connections from a source port other
than the one that it declares it accepts incoming connections on, and
MUST be prepared to receive connections on port 3868. A given
Diameter instance of the peer state machine MUST NOT use more than
one transport connection to communicate with a given peer, unless
multiple instances exist on the peer in which case a separate
connection per process is allowed.

When no transport connection exists with a peer, an attempt to
connect SHOULD be periodically made. This behavior is handled via
the Tc timer, whose recommended value is 30 seconds. There are
certain exceptions to this rule, such as when a peer has terminated
the transport connection stating that it does not wish to
communicate.

When connecting to a peer and either zero or more transports are
specified, SCTP SHOULD be tried first, followed by TCP. See Section
5.2 for more information on peer discovery.

Diameter implementations SHOULD be able to interpret ICMP protocol
port unreachable messages as explicit indications that the server is
not reachable, subject to security policy on trusting such messages.
Diameter implementations SHOULD also be able to interpret a reset
from the transport and timed-out connection attempts.

If Diameter receives data up from TCP that cannot be parsed or
identified as a Diameter error made by the peer, the stream is
compromised and cannot be recovered. The transport connection MUST
be closed using a RESET call (send a TCP RST bit) or an SCTP ABORT
message (graceful closure is compromised).

2.1.1. SCTP Guidelines

The following are guidelines for Diameter implementations that
support SCTP:

1. For interoperability: All Diameter nodes MUST be prepared to
   receive Diameter messages on any SCTP stream in the association.

2. To prevent blocking: All Diameter nodes SHOULD utilize all SCTP
   streams available to the association to prevent head-of-the-line
   blocking.

2.2. Securing Diameter Messages

Diameter clients, such as Network Access Servers (NASes) and Mobility
Agents MUST support IP Security [SECARCH], and MAY support TLS [TLS].
Diameter servers MUST support TLS and IPsec. The Diameter protocol
MUST NOT be used without any security mechanism (TLS or IPsec).

It is suggested that IPsec can be used primarily at the edges and in
intra-domain traffic, such as using pre-shared keys between a NAS a
local AAA proxy. This also eases the requirements on the NAS to
support certificates. It is also suggested that inter-domain traffic
would primarily use TLS. See Sections 13.1 and 13.2 for more details
on IPsec and TLS usage.

2.3. Diameter Application Compliance

Application Identifiers are advertised during the capabilities
exchange phase (see Section 5.3). For a given application,
advertising support of an application implies that the sender
supports all command codes, and the AVPs specified in the associated
ABNFs, described in the specification.

An implementation MAY add arbitrary non-mandatory AVPs to any command
defined in an application, including vendor-specific AVPs. Please
refer to Section 11.1.1 for details.

2.4. Application Identifiers

   Each Diameter application MUST have an IANA assigned Application
   Identifier (see Section 11.3). The base protocol does not require an
   Application Identifier since its support is mandatory. During the
   capabilities exchange, Diameter nodes inform their peers of locally
   supported applications. Furthermore, all Diameter messages contain
   an Application Identifier, which is used in the message forwarding
   process.

   The following Application Identifier values are defined:

      Diameter Common Messages   0
      NASREQ   1 [NASREQ]
      Mobile-IP   2 [DIAMMIP]
      Diameter Base Accounting   3
      Relay   0xffffffff

   Relay and redirect agents MUST advertise the Relay Application
   Identifier, while all other Diameter nodes MUST advertise locally
   supported applications. The receiver of a Capabilities Exchange
   message advertising Relay service MUST assume that the sender
   supports all current and future applications.

   Diameter relay and proxy agents are responsible for finding an
   upstream server that supports the application of a particular
   message. If none can be found, an error message is returned with the
   Result-Code AVP set to DIAMETER_UNABLE_TO_DELIVER.

2.5. Connections vs. Sessions

   This section attempts to provide the reader with an understanding of
   the difference between connection and session, which are terms used
   extensively throughout this document.

   A connection is a transport level connection between two peers, used
   to send and receive Diameter messages. A session is a logical
   concept at the application layer, and is shared between an access
   device and a server, and is identified via the Session-Id AVP

```
         +--------+              +-------+              +--------+
         | Client |              | Relay |              | Server |
         +--------+              +-------+              +--------+
              <---------->           <---------->
           peer connection A    peer connection B

              <----------------------------->
                      User session x
```
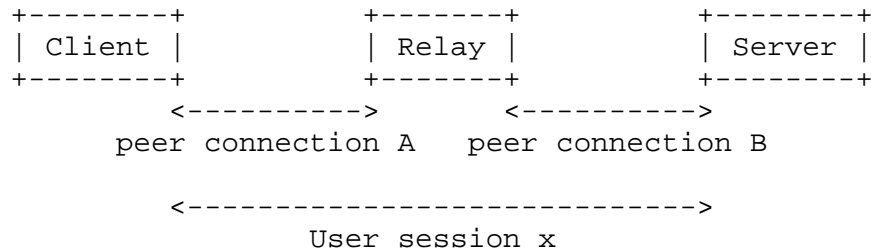
Figure 1: Diameter connections and sessions

In the example provided in Figure 1, peer connection A is established between the Client and its local Relay. Peer connection B is established between the Relay and the Server. User session X spans from the Client via the Relay to the Server. Each "user" of a service causes an auth request to be sent, with a unique session identifier. Once accepted by the server, both the client and the server are aware of the session. It is important to note that there is no relationship between a connection and a session, and that Diameter messages for multiple sessions are all multiplexed through a single connection.

2.6. Peer Table

The Diameter Peer Table is used in message forwarding, and referenced by the Realm Routing Table. A Peer Table entry contains the following fields:

Host identity
   Following the conventions described for the DiameterIdentity
   derived AVP data format in Section 4.4. This field contains the
   contents of the Origin-Host (Section 6.3) AVP found in the CER or
   CEA message.

StatusT
   This is the state of the peer entry, and MUST match one of the
   values listed in Section 5.6.

Static or Dynamic
   Specifies whether a peer entry was statically configured, or
   dynamically discovered.

Expiration time
   Specifies the time at which dynamically discovered peer table
   entries are to be either refreshed, or expired.

   TLS Enabled
      Specifies whether TLS is to be used when communicating with the
      peer.

   Additional security information, when needed (e.g., keys,
   certificates)

2.7. Realm-Based Routing Table

   All Realm-Based routing lookups are performed against what is
   commonly known as the Realm Routing Table (see Section 12). A Realm
   Routing Table Entry contains the following fields:

   Realm Name
      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.

   Application Identifier
      An application is identified by a vendor id and an application id.
      For all IETF standards track Diameter applications, the vendor id
      is zero. A route entry can have a different destination based on
      the application identification AVP of the message. This field
      MUST be 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 route entry with
         the Local Action set to Local can be satisfied locally, and do
         not need to be routed to another server.

      2. RELAY - All Diameter messages that fall within this category
         MUST be routed to a next hop server, without modifying any
         non-routing AVPs. See Section 6.1.8 for relaying guidelines

      3. PROXY - All Diameter messages that fall within this category
         MUST be routed 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 routing. See Section 6.1.8 for
         proxying guidelines.

      4. 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 6.1.7
         for redirect guidelines.

   Server Identifier
      One or more servers the message is to be routed to. These servers
      MUST also be present in the Peer table. When the Local Action is
      set to RELAY or PROXY, this field contains the identity of the
      server(s) the message must be routed to. When the Local Action
      field is set to REDIRECT, this field contains the identity of one
      or more servers the message should be redirected to.

   Static or Dynamic
      Specifies whether a route entry was statically configured, or
      dynamically discovered.

   Expiration time
      Specifies the time which a dynamically discovered route table
      entry expires.

   It is important to note that Diameter agents MUST support at least
   one of the LOCAL, RELAY, PROXY or REDIRECT modes of operation.
   Agents do not need to support all modes of operation in order to
   conform with the protocol specification, but MUST follow the protocol
   compliance guidelines in Section 2. Relay agents MUST NOT reorder
   AVPs, and proxies MUST NOT reorder AVPs.

   The routing table MAY include a default entry that MUST be used for
   any requests not matching any of the other entries. The routing
   table MAY consist of only such an entry.

   When a request is routed, the target server MUST have advertised the
   Application Identifier (see Section 2.4) for the given message, or
   have advertised itself as a relay or proxy agent. Otherwise, an
   error is returned with the Result-Code AVP set to
   DIAMETER_UNABLE_TO_DELIVER.

2.8. Role of Diameter Agents

   In addition to client and servers, the Diameter protocol introduces
   relay, proxy, redirect, and translation agents, each of which is
   defined in Section 1.3. These Diameter agents are useful for several
   reasons:

   - They can distribute administration of systems to a configurable
      grouping, including the maintenance of security associations.

   - They can be used for concentration of requests from an number of
      co-located or distributed NAS equipment sets to a set of like user
      groups.

   - They can do value-added processing to the requests or responses.

- They can be used for load balancing.

- A complex network will have multiple authentication sources, they
  can sort requests and forward towards the correct target.

The Diameter protocol requires that agents maintain transaction
state, which is used for failover purposes. Transaction state
implies that upon forwarding a request, its Hop-by-Hop identifier is
saved; the field is replaced with a locally unique identifier, which
is restored to its original value when the corresponding answer is
received. The request's state is released upon receipt of the
answer. A stateless agent is one that only maintains transaction
state.

The Proxy-Info AVP allows stateless agents to add local state to a
Diameter request, with the guarantee that the same state will be
present in the answer. However, the protocol's failover procedures
require that agents maintain a copy of pending requests.

A stateful agent is one that maintains session state information; by
keeping track of all authorized active sessions. Each authorized
session is bound to a particular service, and its state is considered
active either until it is notified otherwise, or by expiration. Each
authorized session has an expiration, which is communicated by
Diameter servers via the Session-Timeout AVP.

Maintaining session state MAY be useful in certain applications, such
as:

- Protocol translation (e.g., RADIUS <-> Diameter)

- Limiting resources authorized to a particular user

- Per user or transaction auditing

A Diameter agent MAY act in a stateful manner for some requests and
be stateless for others. A Diameter implementation MAY act as one
type of agent for some requests, and as another type of agent for
others.

2.8.1. Relay Agents

Relay Agents are Diameter agents that accept requests and route
messages to other Diameter nodes based on information found in the
messages (e.g., Destination-Realm). This routing decision is
performed using a list of supported realms, and known peers. This is
known as the Realm Routing Table, as is defined further in Section
2.7.

Relays MAY be used to aggregate requests from multiple Network Access
Servers (NASes) within a common geographical area (POP). The use of
Relays is advantageous since it eliminates the need for NASes to be
configured with the necessary security information they would
otherwise require to communicate with Diameter servers in other
realms. Likewise, this reduces the configuration load on Diameter
servers that would otherwise be necessary when NASes are added,
changed or deleted.

Relays modify Diameter messages by inserting and removing routing
information, but do not modify any other portion of a message.
Relays SHOULD NOT maintain session state but MUST maintain
transaction state.

```
+------+    --------->    +------+    --------->    +------+
|      |    1. Request    |      |    2. Request    |      |
| NAS  |                  | DRL  |                  | HMS  |
|      |    4. Answer     |      |    3. Answer     |      |
+------+    <---------    +------+    <---------    +------+
example.net               example.net               example.com
```
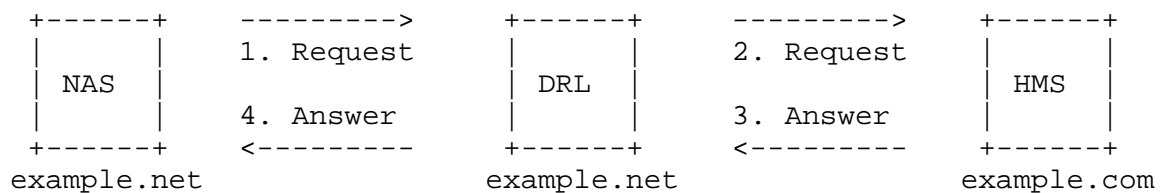
Figure 2: Relaying of Diameter messages

The example provided in Figure 2 depicts a request issued from NAS,
which is an access device, for the user bob@example.com. Prior to
issuing the request, NAS performs a Diameter route lookup, using
"example.com" as the key, and determines that the message is to be
relayed to DRL, which is a Diameter Relay. DRL performs the same
route lookup as NAS, and relays the message to HMS, which is
example.com's Home Diameter Server. HMS identifies that the request
can be locally supported (via the realm), processes the
authentication and/or authorization request, and replies with an
answer, which is routed back to NAS using saved transaction state.

Since Relays do not perform any application level processing, they
provide relaying services for all Diameter applications, and
therefore MUST advertise the Relay Application Identifier.

2.8.2. Proxy Agents

Similarly to relays, proxy agents route Diameter messages using the
Diameter Routing Table. However, they differ since they modify
messages to implement policy enforcement. This requires that proxies
maintain the state of their downstream peers (e.g., access devices)
to enforce resource usage, provide admission control, and
provisioning.

It is important to note that although proxies MAY provide a value-add
function for NASes, they do not allow access devices to use end-to-
end security, since modifying messages breaks authentication.

Proxies MAY be used in call control centers or access ISPs that
provide outsourced connections, they can monitor the number and types
of ports in use, and make allocation and admission decisions
according to their configuration.

Proxies that wish to limit resources MUST maintain session state.
All proxies MUST maintain transaction state.

Since enforcing policies requires an understanding of the service
being provided, Proxies MUST only advertise the Diameter applications
they support.

2.8.3. Redirect Agents

Redirect agents are useful in scenarios where the Diameter routing
configuration needs to be centralized. An example is a redirect
agent that provides services to all members of a consortium, but does
not wish to be burdened with relaying all messages between realms.
This scenario is advantageous since it does not require that the
consortium provide routing updates to its members when changes are
made to a member's infrastructure.

Since redirect agents do not relay messages, and only return an
answer with the information necessary for Diameter agents to
communicate directly, they do not modify messages. Since redirect
agents do not receive answer messages, they cannot maintain session
state. Further, since redirect agents never relay requests, they are
not required to maintain transaction state.

The example provided in Figure 3 depicts a request issued from the
access device, NAS, for the user bob@example.com. The message is
forwarded by the NAS to its relay, DRL, which does not have a routing
entry in its Diameter Routing Table for example.com. DRL has a
default route configured to DRD, which is a redirect agent that
returns a redirect notification to DRL, as well as HMS' contact
information. Upon receipt of the redirect notification, DRL
establishes a transport connection with HMS, if one doesn't already
exist, and forwards the request to it.

```
                             +------+
                             |      |
                             | DRD  |
                             |      |
                             +------+
                              ^    |
           2. Request         |    |  3. Redirection
                              |    |     Notification
                              |    v
   +------+    --------->    +------+    --------->    +------+
   |      |    1. Request    |      |    4. Request    |      |
   | NAS  |                  | DRL  |                  | HMS  |
   |      |    6. Answer     |      |    5. Answer     |      |
   +------+    <---------    +------+    <---------    +------+
   example.net                example.net               example.com
```
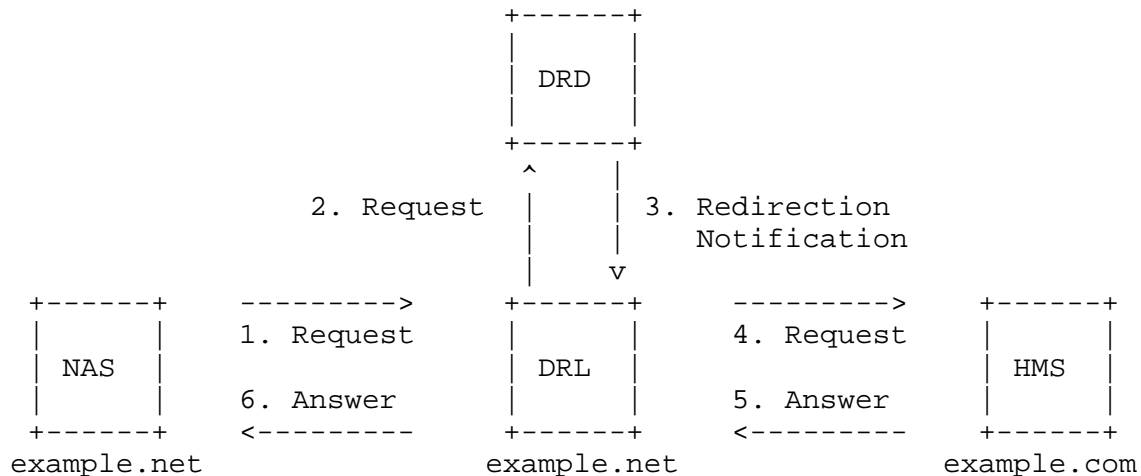
                 Figure 3: Redirecting a Diameter Message

   Since redirect agents do not perform any application level
   processing, they provide relaying services for all Diameter
   applications, and therefore MUST advertise the Relay Application
   Identifier.

2.8.4. Translation Agents

   A translation agent is a device that provides translation between two
   protocols (e.g., RADIUS<->Diameter, TACACS+<->Diameter). Translation
   agents are likely to be used as aggregation servers to communicate
   with a Diameter infrastructure, while allowing for the embedded
   systems to be migrated at a slower pace.

   Given that the Diameter protocol introduces the concept of long-lived
   authorized sessions, translation agents MUST be session stateful and
   MUST maintain transaction state.

   Translation of messages can only occur if the agent recognizes the
   application of a particular request, and therefore translation agents
   MUST only advertise their locally supported applications.

```
   +------+    --------->    +------+    --------->    +------+
   |      |   RADIUS Request |      |  Diameter Request |      |
   | NAS  |                  | TLA  |                  | HMS  |
   |      |   RADIUS Answer  |      |  Diameter Answer |      |
   +------+    <---------    +------+    <---------    +------+
   example.net                example.net               example.com
```
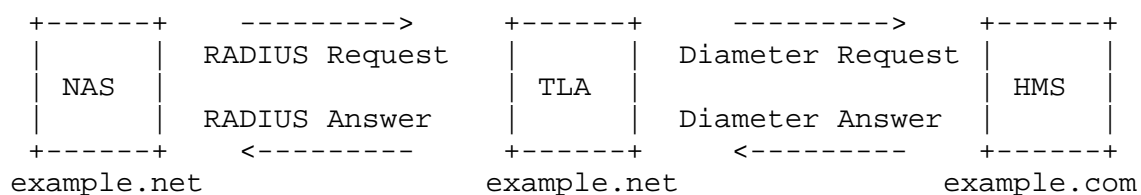
                Figure 4: Translation of RADIUS to Diameter

2.9. End-to-End Security Framework

   End-to-end security services include confidentiality and message
   origin authentication. These services are provided by supporting AVP
   integrity and confidentiality between two peers, communicating
   through agents.

   End-to-end security is provided via the End-to-End security
   extension, described in [AAACMS]. The circumstances requiring the
   use of end-to-end security are determined by policy on each of the
   peers. Security policies, which are not the subject of
   standardization, may be applied by next hop Diameter peer or by
   destination realm. For example, where TLS or IPsec transmission-
   level security is sufficient, there may be no need for end-to-end
   security.

   End-to-end security policies include:

   - Never use end-to-end security.

   - Use end-to-end security on messages containing sensitive AVPs.
      Which AVPs are sensitive is determined by service provider policy.
      AVPs containing keys and passwords should be considered sensitive.
      Accounting AVPs may be considered sensitive. Any AVP for which
      the P bit may be set or which may be encrypted may be considered
      sensitive.

   - Always use end-to-end security.

   It is strongly RECOMMENDED that all Diameter implementations support
   end-to-end security.

2.10. Diameter Path Authorization

   As noted in Section 2.2, Diameter requires transmission level
   security to be used on each connection (TLS or IPsec). Therefore,
   each connection is authenticated, replay and integrity protected and
   confidential on a per-packet basis.

   In addition to authenticating each connection, each connection as
   well as the entire session MUST also be authorized. Before
   initiating a connection, a Diameter Peer MUST check that its peers
   are authorized to act in their roles. For example, a Diameter peer
   may be authentic, but that does not mean that it is authorized to act
   as a Diameter Server advertising a set of Diameter applications.

Prior to bringing up a connection, authorization checks are performed
at each connection along the path. Diameter capabilities negotiation
(CER/CEA) also MUST be carried out, in order to determine what
Diameter applications are supported by each peer. Diameter sessions
MUST be routed only through authorized nodes that have advertised
support for the Diameter application required by the session.

As noted in Section 6.1.8, a relay or proxy agent MUST append a
Route-Record AVP to all requests forwarded. The AVP contains the
identity of the peer the request was received from.

The home Diameter server, prior to authorizing a session, MUST check
the Route-Record AVPs to make sure that the route traversed by the
request is acceptable. For example, administrators within the home
realm may not wish to honor requests that have been routed through an
untrusted realm. By authorizing a request, the home Diameter server
is implicitly indicating its willingness to engage in the business
transaction as specified by the contractual relationship between the
server and the previous hop. A DIAMETER_AUTHORIZATION_REJECTED error
message (see Section 7.1.5) is sent if the route traversed by the
request is unacceptable.

A home realm may also wish to check that each accounting request
message corresponds to a Diameter response authorizing the session.
Accounting requests without corresponding authorization responses
SHOULD be subjected to further scrutiny, as should accounting
requests indicating a difference between the requested and provided
service.

Similarly, the local Diameter agent, on receiving a Diameter response
authorizing a session, MUST check the Route-Record AVPs to make sure
that the route traversed by the response is acceptable. At each
step, forwarding of an authorization response is considered evidence
of a willingness to take on financial risk relative to the session.
A local realm may wish to limit this exposure, for example, by
establishing credit limits for intermediate realms and refusing to
accept responses which would violate those limits. By issuing an
accounting request corresponding to the authorization response, the
local realm implicitly indicates its agreement to provide the service
indicated in the authorization response. If the service cannot be
provided by the local realm, then a DIAMETER_UNABLE_TO_COMPLY error
message MUST be sent within the accounting request; a Diameter client
receiving an authorization response for a service that it cannot
perform MUST NOT substitute an alternate service, and then send
accounting requests for the alternate service instead.

3. Diameter Header

   A summary of the Diameter header format is shown below. The fields
   are transmitted in network byte order.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    Version    |                 Message Length                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | command flags |                  Command-Code                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         Application-ID                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Hop-by-Hop Identifier                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      End-to-End Identifier                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  AVPs ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-
```
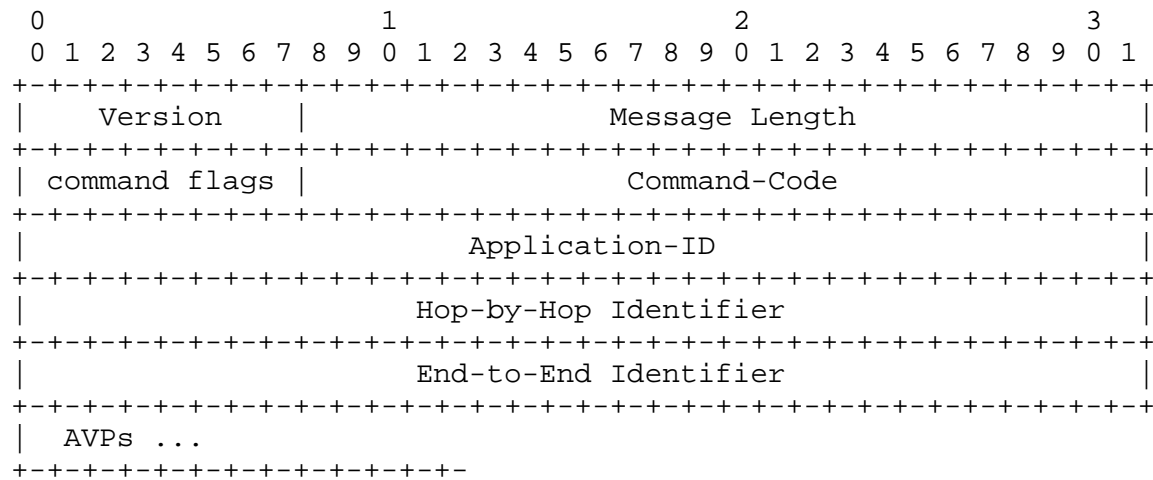
   Version
      This Version field MUST be set to 1 to indicate Diameter Version
      1.

   Message Length
      The Message Length field is three octets and indicates the length
      of the Diameter message including the header fields.

   Command Flags
      The Command Flags field is eight bits. The following bits are
      assigned:

```
    0 1 2 3 4 5 6 7
   +-+-+-+-+-+-+-+-+
   |R P E T r r r r|
   +-+-+-+-+-+-+-+-+
```
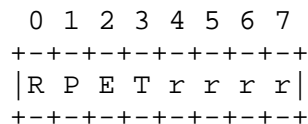
      R(equest)   - If set, the message is a request. If cleared, the
                    message is an answer.
      P(roxiable) - If set, the message MAY be proxied, relayed or
                    redirected. If cleared, the message MUST be
                    locally processed.
      E(rror)   - If set, the message contains a protocol error,
                    and the message will not conform to the ABNF
                    described for this command. Messages with the 'E'

                         bit set are commonly referred to as error
                         messages. This bit MUST NOT be set in request
                         messages. See Section 7.2.
         T(Potentially re-transmitted message)
                       - This flag is set after a link failover procedure,
                         to aid the removal of duplicate requests. It is
                         set when resending requests not yet acknowledged,
                         as an indication of a possible duplicate due to a
                         link failure. This bit MUST be cleared when
                         sending a request for the first time, otherwise
                         the sender MUST set this flag. Diameter agents
                         only need to be concerned about the number of
                         requests they send based on a single received
                         request; retransmissions by other entities need
                         not be tracked. Diameter agents that receive a
                         request with the T flag set, MUST keep the T flag
                         set in the forwarded request. This flag MUST NOT
                         be set if an error answer message (e.g., a
                         protocol error) has been received for the earlier
                         message. It can be set only in cases where no
                         answer has been received from the server for a
                         request and the request is sent again. This flag
                         MUST NOT be set in answer messages.

         r(eserved) - these flag bits are reserved for future use, and
                         MUST be set to zero, and ignored by the receiver.

      Command-Code
         The Command-Code field is three octets, and is used in order to
         communicate the command associated with the message. The 24-bit
         address space is managed by IANA (see Section 11.2.1).

         Command-Code values 16,777,214 and 16,777,215 (hexadecimal values
         FFFFFE -FFFFFF) are reserved for experimental use (See Section
         11.3).

      Application-ID
         Application-ID is four octets and is used to identify to which
         application the message is applicable for. The application can be
         an authentication application, an accounting application or a
         vendor specific application. See Section 11.3 for the possible
         values that the application-id may use.

         The application-id in the header MUST be the same as what is
         contained in any relevant AVPs contained in the message.

Hop-by-Hop Identifier
   The Hop-by-Hop Identifier is an unsigned 32-bit integer field (in
   network byte order) and aids in matching requests and replies.
   The sender MUST ensure that the Hop-by-Hop identifier in a request
   is unique on a given connection at any given time, and MAY attempt
   to ensure that the number is unique across reboots. The sender of
   an Answer message MUST ensure that the Hop-by-Hop Identifier field
   contains the same value that was found in the corresponding
   request. The Hop-by-Hop identifier is normally a monotonically
   increasing number, whose start value was randomly generated. An
   answer message that is received with an unknown Hop-by-Hop
   Identifier MUST be discarded.

End-to-End Identifier
   The End-to-End Identifier is an unsigned 32-bit integer field (in
   network byte order) and is used to detect duplicate messages.
   Upon reboot implementations MAY set the high order 12 bits to
   contain the low order 12 bits of current time, and the low order
   20 bits to a random value. Senders of request messages MUST
   insert a unique identifier on each message. The identifier MUST
   remain locally unique for a period of at least 4 minutes, even
   across reboots. The originator of an Answer message MUST ensure
   that the End-to-End Identifier field contains the same value that
   was found in the corresponding request. The End-to-End Identifier
   MUST NOT be modified by Diameter agents of any kind. The
   combination of the Origin-Host (see Section 6.3) and this field is
   used to detect duplicates. Duplicate requests SHOULD cause the
   same answer to be transmitted (modulo the hop-by-hop Identifier
   field and any routing AVPs that may be present), and MUST NOT
   affect any state that was set when the original request was
   processed. Duplicate answer messages that are to be locally
   consumed (see Section 6.2) SHOULD be silently discarded.

AVPs
   AVPs are a method of encapsulating information relevant to the
   Diameter message. See Section 4 for more information on AVPs.

3.1. Command Codes

   Each command Request/Answer pair is assigned a command code, and the
   sub-type (i.e., request or answer) is identified via the 'R' bit in
   the Command Flags field of the Diameter header.

   Every Diameter message MUST contain a command code in its header's
   Command-Code field, which is used to determine the action that is to
   be taken for a particular message. The following Command Codes are
   defined in the Diameter base protocol:

   Command-Name   Abbrev.   Code   Reference

   Abort-Session-Request   ASR   274   8.5.1
   Abort-Session-Answer   ASA   274   8.5.2
   Accounting-Request   ACR   271   9.7.1
   Accounting-Answer   ACA   271   9.7.2
   Capabilities-Exchange-   CER   257   5.3.1
      Request
   Capabilities-Exchange-   CEA   257   5.3.2
      Answer
   Device-Watchdog-Request   DWR   280   5.5.1
   Device-Watchdog-Answer   DWA   280   5.5.2
   Disconnect-Peer-Request   DPR   282   5.4.1
   Disconnect-Peer-Answer   DPA   282   5.4.2
   Re-Auth-Request   RAR   258   8.3.1
   Re-Auth-Answer   RAA   258   8.3.2
   Session-Termination-   STR   275   8.4.1
      Request
   Session-Termination-   STA   275   8.4.2
      Answer

3.2. Command Code ABNF specification

   Every Command Code defined MUST include a corresponding ABNF
   specification, which is used to define the AVPs that MUST or MAY be
   present. The following format is used in the definition:

   command-def   = command-name "::=" diameter-message

   command-name   = diameter-name

   diameter-name   = ALPHA *(ALPHA / DIGIT / "-")

   diameter-message = header [*fixed] [*required] [*optional]
                      [*fixed]

   header   = "<" Diameter-Header:" command-id
                     [r-bit] [p-bit] [e-bit] [application-id]">"

   application-id = 1*DIGIT

   command-id   = 1*DIGIT
                     ; The Command Code assigned to the command

   r-bit   = ", REQ"
                     ; If present, the 'R' bit in the Command
                     ; Flags is set, indicating that the message
                     ; is a request, as opposed to an answer.

   p-bit   = ", PXY"
                     ; If present, the 'P' bit in the Command
                     ; Flags is set, indicating that the message
                     ; is proxiable.

   e-bit   = ", ERR"
                     ; If present, the 'E' bit in the Command
                     ; Flags is set, indicating that the answer
                     ; message contains a Result-Code AVP in
                     ; the "protocol error" class.

   fixed   = [qual] "<" avp-spec ">"
                     ; Defines the fixed position of an AVP

   required   = [qual] "{" avp-spec "}"
                     ; The AVP MUST be present and can appear
                     ; anywhere in the message.

```
optional            = [qual] "[" avp-name "]"
                      ; The avp-name in the 'optional' rule cannot
                      ; evaluate to any AVP Name which is included
                      ; in a fixed or required rule.  The AVP can
                      ; appear anywhere in the message.

qual                = [min] "*" [max]
                      ; See ABNF conventions, RFC 2234 Section 6.6.
                      ; The absence of any qualifiers depends on whether
                      ; it precedes a fixed, required, or optional
                      ; rule.  If a fixed or required rule has no
                      ; qualifier, then exactly one such AVP MUST
                      ; be present.  If an optional rule has no
                      ; qualifier, then 0 or 1 such AVP may be
                      ; present.
                      ;
                      ; NOTE:  "[" and "]" have a different meaning
                      ; than in ABNF (see the optional rule, above).
                      ; These braces cannot be used to express
                      ; optional fixed rules (such as an optional
                      ; ICV at the end).  To do this, the convention
                      ; is '0*1fixed'.

min                 = 1*DIGIT
                      ; The minimum number of times the element may
                      ; be present.  The default value is zero.

max                 = 1*DIGIT
                      ; The maximum number of times the element may
                      ; be present.  The default value is infinity.  A
                      ; value of zero implies the AVP MUST NOT be
                      ; present.

avp-spec            = diameter-name
                      ; The avp-spec has to be an AVP Name, defined
                      ; in the base or extended Diameter
                      ; specifications.

avp-name            = avp-spec / "AVP"
                      ; The string "AVP" stands for *any* arbitrary
                      ; AVP Name, which does not conflict with the
                      ; required or fixed position AVPs defined in
                      ; the command code definition.
```

The following is a definition of a fictitious command code:

```
Example-Request ::= < "Diameter-Header: 9999999, REQ, PXY >
                     { User-Name }
                 * { Origin-Host }
                 * [ AVP
```

3.3. Diameter Command Naming Conventions

   Diameter command names typically includes one or more English words
   followed by the verb Request or Answer. Each English word is
   delimited by a hyphen. A three-letter acronym for both the request
   and answer is also normally provided.

   An example is a message set used to terminate a session. The command
   name is Session-Terminate-Request and Session-Terminate-Answer, while
   the acronyms are STR and STA, respectively.

   Both the request and the answer for a given command share the same
   command code. The request is identified by the R(equest) bit in the
   Diameter header set to one (1), to ask that a particular action be
   performed, such as authorizing a user or terminating a session. Once
   the receiver has completed the request it issues the corresponding
   answer, which includes a result code that communicates one of the
   following:

   - The request was successful

   - The request failed

   - An additional request must be sent to provide information the peer
      requires prior to returning a successful or failed answer.

   - The receiver could not process the request, but provides
      information about a Diameter peer that is able to satisfy the
      request, known as redirect.

   Additional information, encoded within AVPs, MAY also be included in
   answer messages.

4. Diameter AVPs

   Diameter AVPs carry specific authentication, accounting,
   authorization, routing and security information as well as
   configuration details for the request and reply.

   Some AVPs MAY be listed more than once. The effect of such an AVP is
   specific, and is specified in each case by the AVP description.

Each AVP of type OctetString MUST be padded to align on a 32-bit
boundary, while other AVP types align naturally. A number of zero-
valued bytes are added to the end of the AVP Data field till a word
boundary is reached. The length of the padding is not reflected in
the AVP Length field.

4.1. AVP Header

   The fields in the AVP header MUST be sent in network byte order. The
   format of the header is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           AVP Code                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |V M P r r r r r|                  AVP Length                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Vendor-ID (opt)                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    Data ...
   +-+-+-+-+-+-+-+-+-+
```
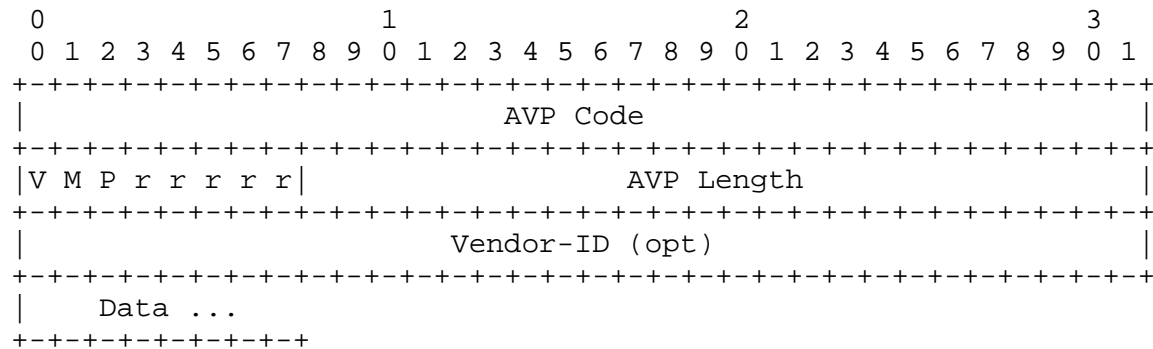
   AVP Code
      The AVP Code, combined with the Vendor-Id field, identifies the
      attribute uniquely. AVP numbers 1 through 255 are reserved for
      backward compatibility with RADIUS, without setting the Vendor-Id
      field. AVP numbers 256 and above are used for Diameter, which are
      allocated by IANA (see Section 11.1).

   AVP Flags
      The AVP Flags field informs the receiver how each attribute must
      be handled. The 'r' (reserved) bits are unused and SHOULD be set
      to 0. Note that subsequent Diameter applications MAY define
      additional bits within the AVP Header, and an unrecognized bit
      SHOULD be considered an error. The 'P' bit indicates the need for
      encryption for end-to-end security.

      The 'M' Bit, known as the Mandatory bit, indicates whether support
      of the AVP is required. If an AVP with the 'M' bit set is
      received by a Diameter client, server, proxy, or translation agent
      and either the AVP or its value is unrecognized, the message MUST
      be rejected. Diameter Relay and redirect agents MUST NOT reject
      messages with unrecognized AVPs.

The 'M' bit MUST be set according to the rules defined for the AVP
containing it. In order to preserve interoperability, a Diameter
implementation MUST be able to exclude from a Diameter message any
Mandatory AVP which is neither defined in the base Diameter
protocol nor in any of the Diameter Application specifications
governing the message in which it appears. It MAY do this in one
of the following ways:

1) If a message is rejected because it contains a Mandatory AVP
   which is neither defined in the base Diameter standard nor in
   any of the Diameter Application specifications governing the
   message in which it appears, the implementation may resend the
   message without the AVP, possibly inserting additional standard
   AVPs instead.

2) A configuration option may be provided on a system wide, per
   peer, or per realm basis that would allow/prevent particular
   Mandatory AVPs to be sent. Thus an administrator could change
   the configuration to avoid interoperability problems.

Diameter implementations are required to support all Mandatory
AVPs which are allowed by the message's formal syntax and defined
either in the base Diameter standard or in one of the Diameter
Application specifications governing the message.

AVPs with the 'M' bit cleared are informational only and a
receiver that receives a message with such an AVP that is not
supported, or whose value is not supported, MAY simply ignore the
AVP.

The 'V' bit, known as the Vendor-Specific bit, indicates whether
the optional Vendor-ID field is present in the AVP header. When
set the AVP Code belongs to the specific vendor code address
space.

Unless otherwise noted, AVPs will have the following default AVP
Flags field settings:

     The 'M' bit MUST be set. The 'V' bit MUST NOT be set.

AVP Length
   The AVP Length field is three octets, and indicates the number of
   octets in this AVP including the AVP Code, AVP Length, AVP Flags,
   Vendor-ID field (if present) and the AVP data. If a message is
   received with an invalid attribute length, the message SHOULD be
   rejected.

4.1.1. Optional Header Elements

   The AVP Header contains one optional field. This field is only
   present if the respective bit-flag is enabled.

   Vendor-ID
      The Vendor-ID field is present if the 'V' bit is set in the AVP
      Flags field. The optional four-octet Vendor-ID field contains the
      IANA assigned "SMI Network Management Private Enterprise Codes"
      [ASSIGNNO] value, encoded in network byte order. Any vendor
      wishing to implement a vendor-specific Diameter AVP MUST use their
      own Vendor-ID along with their privately managed AVP address
      space, guaranteeing that they will not collide with any other
      vendor's vendor-specific AVP(s), nor with future IETF
      applications.

      A vendor ID value of zero (0) corresponds to the IETF adopted AVP
      values, as managed by the IANA. Since the absence of the vendor
      ID field implies that the AVP in question is not vendor specific,
      implementations MUST NOT use the zero (0) vendor ID.

4.2. Basic AVP Data Formats

   The Data field is zero or more octets and contains information
   specific to the Attribute. The format and length of the Data field
   is determined by the AVP Code and AVP Length fields. The format of
   the Data field MUST be one of the following base data types or a data
   type derived from the base data types. In the event that a new Basic
   AVP Data Format is needed, a new version of this RFC must be created.

   OctetString
      The data contains arbitrary data of variable length. Unless
      otherwise noted, the AVP Length field MUST be set to at least 8
      (12 if the 'V' bit is enabled). AVP Values of this type that are
      not a multiple of four-octets in length is followed by the
      necessary padding so that the next AVP (if any) will start on a
      32-bit boundary.

   Integer32
      32 bit signed value, in network byte order. The AVP Length field
      MUST be set to 12 (16 if the 'V' bit is enabled).

   Integer64
      64 bit signed value, in network byte order. The AVP Length field
      MUST be set to 16 (20 if the 'V' bit is enabled).

   Unsigned32
      32 bit unsigned value, in network byte order. The AVP Length
      field MUST be set to 12 (16 if the 'V' bit is enabled).

   Unsigned64
      64 bit unsigned value, in network byte order. The AVP Length
      field MUST be set to 16 (20 if the 'V' bit is enabled).

   Float32
      This represents floating point values of single precision as
      described by [FLOATPOINT]. The 32-bit value is transmitted in
      network byte order. The AVP Length field MUST be set to 12 (16 if
      the 'V' bit is enabled).

   Float64
      This represents floating point values of double precision as
      described by [FLOATPOINT]. The 64-bit value is transmitted in
      network byte order. The AVP Length field MUST be set to 16 (20 if
      the 'V' bit is enabled).

   Grouped
      The Data field is specified as a sequence of AVPs. Each of these
      AVPs follows - in the order in which they are specified -
      including their headers and padding. The AVP Length field is set
      to 8 (12 if the 'V' bit is enabled) plus the total length of all
      included AVPs, including their headers and padding. Thus the AVP
      length field of an AVP of type Grouped is always a multiple of 4.

4.3. Derived AVP Data Formats

   In addition to using the Basic AVP Data Formats, applications may
   define data formats derived from the Basic AVP Data Formats. An
   application that defines new AVP Derived Data Formats MUST include
   them in a section entitled "AVP Derived Data Formats", using the same
   format as the definitions below. Each new definition must be either
   defined or listed with a reference to the RFC that defines the
   format.

   The below AVP Derived Data Formats are commonly used by applications.

   Address
      The Address format is derived from the OctetString AVP Base
      Format. It is a discriminated union, representing, for example a
      32-bit (IPv4) [IPV4] or 128-bit (IPv6) [IPV6] address, most
      significant octet first. The first two octets of the Address

AVP represents the AddressType, which contains an Address Family
defined in [IANAADFAM]. The AddressType is used to discriminate
the content and format of the remaining octets.

Time
   The Time format is derived from the OctetString AVP Base Format.
   The string MUST contain four octets, in the same format as the
   first four bytes are in the NTP timestamp format. The NTP
   Timestamp format is defined in chapter 3 of [SNTP].

   This represents the number of seconds since 0h on 1 January 1900
   with respect to the Coordinated Universal Time (UTC).

   On 6h 28m 16s UTC, 7 February 2036 the time value will overflow.
   SNTP [SNTP] describes a procedure to extend the time to 2104.
   This procedure MUST be supported by all DIAMETER nodes.

UTF8String
   The UTF8String format is derived from the OctetString AVP Base
   Format. This is a human readable string represented using the
   ISO/IEC IS 10646-1 character set, encoded as an OctetString using
   the UTF-8 [UFT8] transformation format described in RFC 2279.

   Since additional code points are added by amendments to the 10646
   standard from time to time, implementations MUST be prepared to
   encounter any code point from 0x00000001 to 0x7fffffff. Byte
   sequences that do not correspond to the valid encoding of a code
   point into UTF-8 charset or are outside this range are prohibited.

   The use of control codes SHOULD be avoided. When it is necessary
   to represent a new line, the control code sequence CR LF SHOULD be
   used.

   The use of leading or trailing white space SHOULD be avoided.

   For code points not directly supported by user interface hardware
   or software, an alternative means of entry and display, such as
   hexadecimal, MAY be provided.

   For information encoded in 7-bit US-ASCII, the UTF-8 charset is
   identical to the US-ASCII charset.

   UTF-8 may require multiple bytes to represent a single character /
   code point; thus the length of an UTF8String in octets may be
   different from the number of characters encoded.

   Note that the AVP Length field of an UTF8String is measured in
   octets, not characters.

   DiameterIdentity
      The DiameterIdentity format is derived from the OctetString AVP
      Base Format.

         DiameterIdentity = FQDN

      DiameterIdentity value is used to uniquely identify a Diameter
      node for purposes of duplicate connection and routing loop
      detection.

      The contents of the string MUST be the FQDN of the Diameter node.
      If multiple Diameter nodes run on the same host, each Diameter
      node MUST be assigned a unique DiameterIdentity. If a Diameter
      node can be identified by several FQDNs, a single FQDN should be
      picked at startup, and used as the only DiameterIdentity for that
      node, whatever the connection it is sent on.

   DiameterURI

      The DiameterURI MUST follow the Uniform Resource Identifiers (URI)
      syntax [URI] rules specified below:

      "aaa://" FQDN [port] [transport] [protocol]

                      ; No transport security

      "aaas://" FQDN [port] [transport] [protocol]

                      ; Transport security used

      FQDN   = Fully Qualified Host Name

      port   = ":" 1*DIGIT

                      ; One of the ports used to listen for
                      ; incoming connections.
                      ; If absent,
                      ; the default Diameter port (3868) is
                      ; assumed.

      transport   = ";transport=" transport-protocol

                      ; One of the transports used to listen
                      ; for incoming connections. If absent,
                      ; the default SCTP [SCTP] protocol is
                      ; assumed. UDP MUST NOT be used when
                      ; the aaa-protocol field is set to
                      ; diameter.

```
   transport-protocol = ( "tcp" / "sctp" / "udp" )

   protocol          = ";protocol=" aaa-protocol

                 ; If absent, the default AAA protocol
                 ; is diameter.

   aaa-protocol      = ( "diameter" / "radius" / "tacacs+" )
```

   The following are examples of valid Diameter host identities:

```
   aaa://host.example.com;transport=tcp
   aaa://host.example.com:6666;transport=tcp
   aaa://host.example.com;protocol=diameter
   aaa://host.example.com:6666;protocol=diameter
   aaa://host.example.com:6666;transport=tcp;protocol=diameter
   aaa://host.example.com:1813;transport=udp;protocol=radius
```

   Enumerated
      Enumerated is derived from the Integer32 AVP Base Format. The
      definition contains a list of valid values and their
      interpretation and is described in the Diameter application
      introducing the AVP.

   IPFilterRule
      The IPFilterRule format is derived from the OctetString AVP Base
      Format. It uses the ASCII charset. Packets may 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.

IPFilterRule filters 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.
                          For a match to occur, the same IP
                          version must be present in the
                          packet that was used in describing
                          the IP address. To test for a
                          particular IP version, the bits part
                          can be set to zero. 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. For IPv4,
                          a typical first rule is often "deny
                          in ip! assigned"

              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.

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

                        {port/port-port}[,ports[,...]]

                    The '-' notation specifies a range of ports
                    (including boundaries).

                    Fragmented packets that have a non-zero offset
                    (i.e., not the first fragment) will never match
                    a rule that 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 specifications.

               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
        that contains a tcpflags specification can never
        match a fragmented packet that 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. Both the numeric values and
        the symbolic values listed below can be used. 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 access device 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.

   An access device that is unable to interpret or apply a deny rule
   MUST terminate the session. An access device that is unable to
   interpret or apply a permit rule MAY apply a more restrictive
   rule. An access device MAY apply deny rules of its own before the
   supplied rules, for example to protect the access device 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.

   QoSFilterRule
      The QosFilterRule format is derived from the OctetString AVP Base
      Format. It uses the ASCII charset. Packets may be marked or
      metered 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)
         DSCP values   (no mask or range)

      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 treated as best
      effort. An access device that is unable to interpret or apply a
      QoS rule SHOULD NOT terminate the session.

      QoSFilterRule filters MUST follow the format:

      action dir proto from src to dst [options]

                 tag   - Mark packet with a specific DSCP
                          [DIFFSERV]. The DSCP option MUST be
                          included.
                 meter - Meter traffic. The metering options
                          MUST be included.

   dir   The format is as described under IPFilterRule.

                 proto   The format is as described under
                 IPFilterRule.

                 src and dst The format is as described under
                 IPFilterRule.

4.4. Grouped AVP Values

   The Diameter protocol allows AVP values of type 'Grouped.' This
   implies that the Data field is actually a sequence of AVPs. It is
   possible to include an AVP with a Grouped type within a Grouped type,
   that is, to nest them. AVPs within an AVP of type Grouped have the
   same padding requirements as non-Grouped AVPs, as defined in Section
   4.

   The AVP Code numbering space of all AVPs included in a Grouped AVP is
   the same as for non-grouped AVPs. Further, if any of the AVPs
   encapsulated within a Grouped AVP has the 'M' (mandatory) bit set,
   the Grouped AVP itself MUST also include the 'M' bit set.

   Every Grouped AVP defined MUST include a corresponding grammar, using
   ABNF [ABNF] (with modifications), as defined below.

```
      grouped-avp-def  = name "::=" avp

      name-fmt         = ALPHA *(ALPHA / DIGIT / "-")

      name             = name-fmt
                           ; The name has to be the name of an AVP,
                           ; defined in the base or extended Diameter
                           ; specifications.

      avp              = header  [ *fixed] [ *required] [ *optional]
                           [ *fixed]

      header           = "<" "AVP-Header:" avpcode [vendor] ">"

      avpcode          = 1*DIGIT
                           ; The AVP Code assigned to the Grouped AVP

      vendor           = 1*DIGIT
                           ; The Vendor-ID assigned to the Grouped AVP.
                           ; If absent, the default value of zero is
                           ; used.
```

4.4.1. Example AVP with a Grouped Data type

   The Example-AVP (AVP Code 999999) is of type Grouped and is used to
   clarify how Grouped AVP values work. The Grouped Data field has the
   following ABNF grammar:

```
      Example-AVP  ::= < AVP Header: 999999 >
                         { Origin-Host }
                      1*{ Session-Id }
                       *[ AVP ]
```

   An Example-AVP with Grouped Data follows.

   The Origin-Host AVP is required (Section 6.3). In this case:

```
      Origin-Host = "example.com".
```

   One or more Session-Ids must follow. Here there are two:

      Session-Id =
        "grump.example.com:33041;23432;893;0AF3B81"

      Session-Id =
        "grump.example.com:33054;23561;2358;0AF3B82"

   optional AVPs included are

      Recovery-Policy = <binary>
        2163bc1d0ad82371f6bc09484133c3f09ad74a0dd5346d54195a7cf0b35
        2cabc881839a4fdcfbc1769e2677a4c1fb499284c5f70b48f58503a45c5
        c2d6943f82d5930f2b7c1da640f476f0e9c9572a50db8ea6e51e1c2c7bd
        f8bb43dc995144b8dbe297ac739493946803e1cee3e15d9b765008a1b2a
        cf4ac777c80041d72c01e691cf751dbf86e85f509f3988e5875dc905119
        26841f00f0e29a6d1ddc1a842289d440268681e052b30fb638045f7779c
        1d873c784f054f688f5001559ecff64865ef975f3e60d2fd7966b8c7f92

      Futuristic-Acct-Record = <binary>
        fe19da5802acd98b07a5b86cb4d5d03f0314ab9ef1ad0b67111ff3b90a0
        57fe29620bf3585fd2dd9fcc38ce62f6cc208c6163c008f4258d1bc88b8
        17694a74ccad3ec69269461b14b2e7a4c111fb239e33714da207983f58c
        41d018d56fe938f3cbf089aac12a912a2f0d1923a9390e5f789cb2e5067
        d3427475e49968f841

   The data for the optional AVPs is represented in hex since the format
   of these AVPs is neither known at the time of definition of the
   Example-AVP group, nor (likely) at the time when the example instance
   of this AVP is interpreted - except by Diameter implementations which
   support the same set of AVPs. The encoding example illustrates how
   padding is used and how length fields are calculated. Also note that
   AVPs may be present in the Grouped AVP value which the receiver
   cannot interpret (here, the Recover-Policy and Futuristic-Acct-Record
   AVPs).

This AVP would be encoded as follows:

```
          0       1       2       3       4       5       6       7
      +-------+-------+-------+-------+-------+-------+-------+-------+
    0 |       Example AVP Header (AVP Code = 999999), Length = 468  |
      +-------+-------+-------+-------+-------+-------+-------+-------+
    8 |       Origin-Host AVP Header (AVP Code = 264), Length = 19  |
      +-------+-------+-------+-------+-------+-------+-------+-------+
   16 | 'e'   | 'x'   | 'a'   | 'm'   | 'p'   | 'l'   | 'e'   | '.'   |
      +-------+-------+-------+-------+-------+-------+-------+-------+
   24 | 'c'   | 'o'   | 'm'   |Padding|    Session-Id AVP Header     |
      +-------+-------+-------+-------+-------+-------+-------+-------+
   32 | (AVP Code = 263), Length = 50 | 'g'   | 'r'   | 'u'   | 'm'   |
      +-------+-------+-------+-------+-------+-------+-------+-------+
                                . . .
      +-------+-------+-------+-------+-------+-------+-------+-------+
   64 | 'A'   | 'F'   | '3'   | 'B'   | '8'   | '1'   |Padding|Padding|
      +-------+-------+-------+-------+-------+-------+-------+-------+
   72 |     Session-Id AVP Header (AVP Code = 263), Length = 51     |
      +-------+-------+-------+-------+-------+-------+-------+-------+
   80 | 'g'   | 'r'   | 'u'   | 'm'   | 'p'   | '.'   | 'e'   | 'x'   |
      +-------+-------+-------+-------+-------+-------+-------+-------+
                                . . .
      +-------+-------+-------+-------+-------+-------+-------+-------+
  104 | '0'   | 'A'   | 'F'   | '3'   | 'B'   | '8'   | '2'   |Padding|
      +-------+-------+-------+-------+-------+-------+-------+-------+
  112 |   Recovery-Policy Header (AVP Code = 8341), Length = 223    |
      +-------+-------+-------+-------+-------+-------+-------+-------+
  120 | 0x21  | 0x63  | 0xbc  | 0x1d  | 0x0a  | 0xd8  | 0x23  | 0x71  |
      +-------+-------+-------+-------+-------+-------+-------+-------+
                                . . .
      +-------+-------+-------+-------+-------+-------+-------+-------+
  320 | 0x2f  | 0xd7  | 0x96  | 0x6b  | 0x8c  | 0x7f  | 0x92  |Padding|
      +-------+-------+-------+-------+-------+-------+-------+-------+
  328 | Futuristic-Acct-Record Header (AVP Code = 15930), Length = 137|
      +-------+-------+-------+-------+-------+-------+-------+-------+
  336 | 0xfe  | 0x19  | 0xda  | 0x58  | 0x02  | 0xac  | 0xd9  | 0x8b  |
      +-------+-------+-------+-------+-------+-------+-------+-------+
                                . . .
      +-------+-------+-------+-------+-------+-------+-------+-------+
  464 | 0x41  |Padding|Padding|Padding|
      +-------+-------+-------+-------+
```

4.5. Diameter Base Protocol AVPs

   The following table describes the Diameter AVPs defined in the base
   protocol, their AVP Code values, types, possible flag values and
   whether the AVP MAY be encrypted. For the originator of a Diameter
   message, "Encr" (Encryption) means that if a message containing that
   AVP is to be sent via a Diameter agent (proxy, redirect or relay)
   then the message MUST NOT be sent unless there is end-to-end security
   between the originator and the recipient and integrity /
   confidentiality protection is offered for this AVP OR the originator
   has locally trusted configuration that indicates that end-to-end
   security is not needed. Similarly, for the originator of a Diameter
   message, a "P" in the "MAY" column means that if a message containing
   that AVP is to be sent via a Diameter agent (proxy, redirect or
   relay) then the message MUST NOT be sent unless there is end-to-end
   security between the originator and the recipient or the originator
   has locally trusted configuration that indicates that end-to-end
   security is not needed.

   Due to space constraints, the short form DiamIdent is used to
   represent DiameterIdentity.

	AVP	Section		AVP Flag rules				
Attribute Name	Code	Defined	Data Type	MUST	MAY	SHLD NOT	MUST NOT	Encr
Acct-Interim-Interval	85	9.8.2	Unsigned32	M	P		V	Y
Accounting-Realtime-Required	483	9.8.7	Enumerated	M	P		V	Y
Acct-Multi-Session-Id	50	9.8.5	UTF8String	M	P		V	Y
Accounting-Record-Number	485	9.8.3	Unsigned32	M	P		V	Y
Accounting-Record-Type	480	9.8.1	Enumerated	M	P		V	Y
Accounting-Session-Id	44	9.8.4	OctetString	M	P		V	Y
Accounting-Sub-Session-Id	287	9.8.6	Unsigned64	M	P		V	Y
Acct-Application-Id	259	6.9	Unsigned32	M	P		V	N
Auth-Application-Id	258	6.8	Unsigned32	M	P		V	N
Auth-Request-Type	274	8.7	Enumerated	M	P		V	N
Authorization-Lifetime	291	8.9	Unsigned32	M	P		V	N
Auth-Grace-Period	276	8.10	Unsigned32	M	P		V	N
Auth-Session-State	277	8.11	Enumerated	M	P		V	N
Re-Auth-Request-Type	285	8.12	Enumerated	M	P		V	N
Class	25	8.20	OctetString	M	P		V	Y
Destination-Host	293	6.5	DiamIdent	M	P		V	N
Destination-Realm	283	6.6	DiamIdent	M	P		V	N
Disconnect-Cause	273	5.4.3	Enumerated	M	P		V	N
E2E-Sequence AVP	300	6.15	Grouped	M	P		V	Y
Error-Message	281	7.3	UTF8String		P		V,M	N
Error-Reporting-Host	294	7.4	DiamIdent		P		V,M	N
Event-Timestamp	55	8.21	Time	M	P		V	N
Experimental-Result	297	7.6	Grouped	M	P		V	N

| | | | | AVP Flag rules | | | | |
| | AVP | Section | | MUST | MAY | SHLD | MUST | MAY |
Attribute Name	Code	Defined	Data Type			NOT	NOT	Encr
Experimental-Result-Code	298	7.7	Unsigned32	M	P		V	N
Failed-AVP	279	7.5	Grouped	M	P		V	N
Firmware-Revision	267	5.3.4	Unsigned32				P,V,M	N
Host-IP-Address	257	5.3.5	Address	M	P		V	N
Inband-Security-Id	299	6.10	Unsigned32	M	P		V	N
Multi-Round-Time-Out	272	8.19	Unsigned32	M	P		V	Y
Origin-Host	264	6.3	DiamIdent	M	P		V	N
Origin-Realm	296	6.4	DiamIdent	M	P		V	N
Origin-State-Id	278	8.16	Unsigned32	M	P		V	N
Product-Name	269	5.3.7	UTF8String				P,V,M	N
Proxy-Host	280	6.7.3	DiamIdent	M			P,V	N
Proxy-Info	284	6.7.2	Grouped	M			P,V	N
Proxy-State	33	6.7.4	OctetString	M			P,V	N
Redirect-Host	292	6.12	DiamURI	M	P		V	N
Redirect-Host-Usage	261	6.13	Enumerated	M	P		V	N
Redirect-Max-Cache-Time	262	6.14	Unsigned32	M	P		V	N
Result-Code	268	7.1	Unsigned32	M	P		V	N
Route-Record	282	6.7.1	DiamIdent	M			P,V	N
Session-Id	263	8.8	UTF8String	M	P		V	Y
Session-Timeout	27	8.13	Unsigned32	M	P		V	N
Session-Binding	270	8.17	Unsigned32	M	P		V	Y
Session-Server-Failover	271	8.18	Enumerated	M	P		V	Y
Supported-Vendor-Id	265	5.3.6	Unsigned32	M	P		V	N
Termination-Cause	295	8.15	Enumerated	M	P		V	N
User-Name	1	8.14	UTF8String	M	P		V	Y
Vendor-Id	266	5.3.3	Unsigned32	M	P		V	N
Vendor-Specific-Application-Id	260	6.11	Grouped	M	P		V	N

5. Diameter Peers

   This section describes how Diameter nodes establish connections and
   communicate with peers.

5.1. Peer Connections

   Although a Diameter node may have many possible peers that it is able
   to communicate with, it may not be economical to have an established
   connection to all of them. At a minimum, a Diameter node SHOULD have
   an established connection with two peers per realm, known as the
   primary and secondary peers. Of course, a node MAY have additional
   connections, if it is deemed necessary. Typically, all messages for
   a realm are sent to the primary peer, but in the event that failover
   procedures are invoked, any pending requests are sent to the
   secondary peer. However, implementations are free to load balance
   requests between a set of peers.

   Note that a given peer MAY act as a primary for a given realm, while
   acting as a secondary for another realm.

   When a peer is deemed suspect, which could occur for various reasons,
   including not receiving a DWA within an allotted timeframe, no new
   requests should be forwarded to the peer, but failover procedures are
   invoked. When an active peer is moved to this mode, additional
   connections SHOULD be established to ensure that the necessary number
   of active connections exists.

   There are two ways that a peer is removed from the suspect peer list:

   1. The peer is no longer reachable, causing the transport connection
      to be shutdown. The peer is moved to the closed state.

   2. Three watchdog messages are exchanged with accepted round trip
      times, and the connection to the peer is considered stabilized.

      In the event the peer being removed is either the primary or
      secondary, an alternate peer SHOULD replace the deleted peer, and
      assume the role of either primary or secondary.

5.2. Diameter Peer Discovery

   Allowing for dynamic Diameter agent discovery will make it possible
   for simpler and more robust deployment of Diameter services. In
   order to promote interoperable implementations of Diameter peer
   discovery, the following mechanisms are described. These are based

on existing IETF standards. The first option (manual configuration)
MUST be supported by all DIAMETER nodes, while the latter two options
(SRVLOC and DNS) MAY be supported.

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

1. The Diameter implementation consults its list of static (manually)
   configured Diameter agent locations. These will be used if they
   exist and respond.

2. The Diameter implementation uses SLPv2 [SLP] to discover Diameter
   services. The Diameter service template [TEMPLATE] is included in
   Appendix A.

   It is recommended that SLPv2 security be deployed (this requires
   distributing keys to SLPv2 agents). This is discussed further in
   Appendix A. SLPv2 security SHOULD be used (requiring distribution
   of keys to SLPv2 agents) in order to ensure that discovered peers
   are authorized for their roles. SLPv2 is discussed further in
   Appendix A.

3. The Diameter implementation performs a NAPTR query for a server in
   a particular realm. The Diameter implementation has to know in
   advance which realm to look for a Diameter agent in. This could
   be deduced, for example, from the 'realm' in a NAI that a Diameter
   implementation needed to perform a Diameter operation on.

   3.1 The services relevant for the task of transport protocol
       selection are those with NAPTR service fields with values
       "AAA+D2x", where x is a letter that corresponds to a transport
       protocol supported by the domain. This specification defines
       D2T for TCP and D2S for SCTP. We also establish an IANA
       registry for NAPTR service name to transport protocol
       mappings.

       These NAPTR records provide a mapping from a domain, to the
       SRV record for contacting a server with the specific transport
       protocol in the NAPTR services field. The resource record
       will contain an empty regular expression and a replacement
       value, which is the SRV record for that particular transport
       protocol. If the server supports multiple transport
       protocols, there will be multiple NAPTR records, each with a
       different service value. As per RFC 2915 [NAPTR], the client

discards any records whose services fields are not applicable.
For the purposes of this specification, several rules are
defined.

3.2 A client MUST discard any service fields that identify a
resolution service whose value is not "D2X", for values of X
that indicate transport protocols supported by the client.
The NAPTR processing as described in RFC 2915 will result in
discovery of the most preferred transport protocol of the
server that is supported by the client, as well as an SRV
record for the server.

The domain suffixes in the NAPTR replacement field SHOULD
match the domain of the original query.

4. If no NAPTR records are found, the requester queries for those
address records for the destination address,
'_diameter._sctp'.realm or '_diameter._tcp'.realm. Address
records include A RR's, AAAA RR's or other similar records, chosen
according to the requestor's network protocol capabilities. If
the DNS server returns no address records, the requestor gives up.

If the server is using a site certificate, the domain name in the
query and the domain name in the replacement field MUST both be
valid based on the site certificate handed out by the server in
the TLS or IKE exchange. Similarly, the domain name in the SRV
query and the domain name in the target in the SRV record MUST
both be valid based on the same site certificate. Otherwise, an
attacker could modify the DNS records to contain replacement
values in a different domain, and the client could not validate
that this was the desired behavior, or the result of an attack

Also, the Diameter Peer MUST check to make sure that the
discovered peers are authorized to act in its role.
Authentication via IKE or TLS, or validation of DNS RRs via DNSSEC
is not sufficient to conclude this. For example, a web server may
have obtained a valid TLS certificate, and secured RRs may be
included in the DNS, but this does not imply that it is authorized
to act as a Diameter Server.

Authorization can be achieved for example, by configuration of a
Diameter Server CA. Alternatively this can be achieved by
definition of OIDs within TLS or IKE certificates so as to signify
Diameter Server authorization.

A dynamically discovered peer causes an entry in the Peer Table (see
Section 2.6) to be created. Note that entries created via DNS MUST
expire (or be refreshed) within the DNS TTL. If a peer is discovered

outside of the local realm, a routing table entry (see Section 2.7)
for the peer's realm is created. The routing table entry's
expiration MUST match the peer's expiration value.

5.3. Capabilities Exchange

When two Diameter peers establish a transport connection, they MUST
exchange the Capabilities Exchange messages, as specified in the peer
state machine (see Section 5.6). This message allows the discovery
of a peer's identity and its capabilities (protocol version number,
supported Diameter applications, security mechanisms, etc.)

The receiver only issues commands to its peers that have advertised
support for the Diameter application that defines the command. A
Diameter node MUST cache the supported applications in order to
ensure that unrecognized commands and/or AVPs are not unnecessarily
sent to a peer.

A receiver of a Capabilities-Exchange-Req (CER) message that does not
have any applications in common with the sender MUST return a
Capabilities-Exchange-Answer (CEA) with the Result-Code AVP set to
DIAMETER_NO_COMMON_APPLICATION, and SHOULD disconnect the transport
layer connection. Note that receiving a CER or CEA from a peer
advertising itself as a Relay (see Section 2.4) MUST be interpreted
as having common applications with the peer.

Similarly, a receiver of a Capabilities-Exchange-Req (CER) message
that does not have any security mechanisms in common with the sender
MUST return a Capabilities-Exchange-Answer (CEA) with the Result-Code
AVP set to DIAMETER_NO_COMMON_SECURITY, and SHOULD disconnect the
transport layer connection.

CERs received from unknown peers MAY be silently discarded, or a CEA
MAY be issued with the Result-Code AVP set to DIAMETER_UNKNOWN_PEER.
In both cases, the transport connection is closed. If the local
policy permits receiving CERs from unknown hosts, a successful CEA
MAY be returned. If a CER from an unknown peer is answered with a
successful CEA, the lifetime of the peer entry is equal to the
lifetime of the transport connection. In case of a transport
failure, all the pending transactions destined to the unknown peer
can be discarded.

The CER and CEA messages MUST NOT be proxied, redirected or relayed.

Since the CER/CEA messages cannot be proxied, it is still possible
that an upstream agent receives a message for which it has no
available peers to handle the application that corresponds to the
Command-Code. In such instances, the 'E' bit is set in the answer

message (see Section 7.) with the Result-Code AVP set to
DIAMETER_UNABLE_TO_DELIVER to inform the downstream to take action
(e.g., re-routing request to an alternate peer).

With the exception of the Capabilities-Exchange-Request message, a
message of type Request that includes the Auth-Application-Id or
Acct-Application-Id AVPs, or a message with an application-specific
command code, MAY only be forwarded to a host that has explicitly
advertised support for the application (or has advertised the Relay
Application Identifier).

5.3.1. Capabilities-Exchange-Request

The Capabilities-Exchange-Request (CER), indicated by the Command-
Code set to 257 and the Command Flags' 'R' bit set, is sent to
exchange local capabilities. Upon detection of a transport failure,
this message MUST NOT be sent to an alternate peer.

When Diameter is run over SCTP [SCTP], which allows for connections
to span multiple interfaces and multiple IP addresses, the
Capabilities-Exchange-Request message MUST contain one Host-IP-
Address AVP for each potential IP address that MAY be locally used
when transmitting Diameter messages.

Message Format

```
   <CER> ::= < Diameter Header: 257, REQ >
             { Origin-Host }
             { Origin-Realm }
          1* { Host-IP-Address }
             { Vendor-Id }
             { Product-Name }
             [ Origin-State-Id ]
           * [ Supported-Vendor-Id ]
           * [ Auth-Application-Id ]
           * [ Inband-Security-Id ]
           * [ Acct-Application-Id ]
           * [ Vendor-Specific-Application-Id ]
             [ Firmware-Revision ]
           * [ AVP ]
```

5.3.2. Capabilities-Exchange-Answer

The Capabilities-Exchange-Answer (CEA), indicated by the Command-Code
set to 257 and the Command Flags' 'R' bit cleared, is sent in
response to a CER message.

When Diameter is run over SCTP [SCTP], which allows connections to
span multiple interfaces, hence, multiple IP addresses, the
Capabilities-Exchange-Answer message MUST contain one Host-IP-Address
AVP for each potential IP address that MAY be locally used when
transmitting Diameter messages.

Message Format

```
<CEA> ::= < Diameter Header: 257 >
             { Result-Code }
             { Origin-Host }
             { Origin-Realm }
          1* { Host-IP-Address }
             { Vendor-Id }
             { Product-Name }
             [ Origin-State-Id ]
             [ Error-Message ]
           * [ Failed-AVP ]
           * [ Supported-Vendor-Id ]
           * [ Auth-Application-Id ]
           * [ Inband-Security-Id ]
           * [ Acct-Application-Id ]
           * [ Vendor-Specific-Application-Id ]
             [ Firmware-Revision ]
           * [ AVP ]
```

5.3.3. Vendor-Id AVP

   The Vendor-Id AVP (AVP Code 266) is of type Unsigned32 and contains
   the IANA "SMI Network Management Private Enterprise Codes" [ASSIGNNO]
   value assigned to the vendor of the Diameter application. In
   combination with the Supported-Vendor-Id AVP (Section 5.3.6), this
   MAY be used in order to know which vendor specific attributes may be
   sent to the peer. It is also envisioned that the combination of the
   Vendor-Id, Product-Name (Section 5.3.7) and the Firmware-Revision
   (Section 5.3.4) AVPs MAY provide very useful debugging information.

   A Vendor-Id value of zero in the CER or CEA messages is reserved and
   indicates that this field is ignored.

5.3.4. Firmware-Revision AVP

   The Firmware-Revision AVP (AVP Code 267) is of type Unsigned32 and is
   used to inform a Diameter peer of the firmware revision of the
   issuing device.

For devices that do not have a firmware revision (general purpose
computers running Diameter software modules, for instance), the
revision of the Diameter software module may be reported instead.

5.3.5. Host-IP-Address AVP

The Host-IP-Address AVP (AVP Code 257) is of type Address and is used
to inform a Diameter peer of the sender's IP address. All source
addresses that a Diameter node expects to use with SCTP [SCTP] MUST
be advertised in the CER and CEA messages by including a Host-IP-
Address AVP for each address. This AVP MUST ONLY be used in the CER
and CEA messages.

5.3.6. Supported-Vendor-Id AVP

The Supported-Vendor-Id AVP (AVP Code 265) is of type Unsigned32 and
contains the IANA "SMI Network Management Private Enterprise Codes"
[ASSIGNNO] value assigned to a vendor other than the device vendor.
This is used in the CER and CEA messages in order to inform the peer
that the sender supports (a subset of) the vendor-specific AVPs
defined by the vendor identified in this AVP.

5.3.7. Product-Name AVP

The Product-Name AVP (AVP Code 269) is of type UTF8String, and
contains the vendor assigned name for the product. The Product-Name
AVP SHOULD remain constant across firmware revisions for the same
product.

5.4. Disconnecting Peer connections

When a Diameter node disconnects one of its transport connections,
its peer cannot know the reason for the disconnect, and will most
likely assume that a connectivity problem occurred, or that the peer
has rebooted. In these cases, the peer may periodically attempt to
reconnect, as stated in Section 2.1. In the event that the
disconnect was a result of either a shortage of internal resources,
or simply that the node in question has no intentions of forwarding
any Diameter messages to the peer in the foreseeable future, a
periodic connection request would not be welcomed. The
Disconnection-Reason AVP contains the reason the Diameter node issued
the Disconnect-Peer-Request message.

The Disconnect-Peer-Request message is used by a Diameter node to
inform its peer of its intent to disconnect the transport layer, and
that the peer shouldn't reconnect unless it has a valid reason to do
so (e.g., message to be forwarded). Upon receipt of the message, the

   Disconnect-Peer-Answer is returned, which SHOULD contain an error if
   messages have recently been forwarded, and are likely in flight,
   which would otherwise cause a race condition.

   The receiver of the Disconnect-Peer-Answer initiates the transport
   disconnect.

5.4.1. Disconnect-Peer-Request

   The Disconnect-Peer-Request (DPR), indicated by the Command-Code set
   to 282 and the Command Flags' 'R' bit set, is sent to a peer to
   inform its intentions to shutdown the transport connection. Upon
   detection of a transport failure, this message MUST NOT be sent to an
   alternate peer.

   Message Format

      <DPR> ::= < Diameter Header: 282, REQ >
                 { Origin-Host }
                 { Origin-Realm }
                 { Disconnect-Cause }

5.4.2. Disconnect-Peer-Answer

   The Disconnect-Peer-Answer (DPA), indicated by the Command-Code set
   to 282 and the Command Flags' 'R' bit cleared, is sent as a response
   to the Disconnect-Peer-Request message. Upon receipt of this
   message, the transport connection is shutdown.

   Message Format

      <DPA> ::= < Diameter Header: 282 >
                 { Result-Code }
                 { Origin-Host }
                 { Origin-Realm }
                 [Error-Message]
               * [Failed-AVP]

5.4.3. Disconnect-Cause AVP

   The Disconnect-Cause AVP (AVP Code 273) is of type Enumerated. A
   Diameter node MUST include this AVP in the Disconnect-Peer-Request
   message to inform the peer of the reason for its intention to
   shutdown the transport connection. The following values are
   supported:

      REBOOTING   0
         A scheduled reboot is imminent.

      BUSY   1
         The peer's internal resources are constrained, and it has
         determined that the transport connection needs to be closed.

      DO_NOT_WANT_TO_TALK_TO_YOU   2
         The peer has determined that it does not see a need for the
         transport connection to exist, since it does not expect any
         messages to be exchanged in the near future.

5.5. Transport Failure Detection

   Given the nature of the Diameter protocol, it is recommended that
   transport failures be detected as soon as possible. Detecting such
   failures will minimize the occurrence of messages sent to unavailable
   agents, resulting in unnecessary delays, and will provide better
   failover performance. The Device-Watchdog-Request and Device-
   Watchdog-Answer messages, defined in this section, are used to pro-
   actively detect transport failures.

5.5.1. Device-Watchdog-Request

   The Device-Watchdog-Request (DWR), indicated by the Command-Code set
   to 280 and the Command Flags' 'R' bit set, is sent to a peer when no
   traffic has been exchanged between two peers (see Section 5.5.3).
   Upon detection of a transport failure, this message MUST NOT be sent
   to an alternate peer.

   Message Format

      <DWR> ::= < Diameter Header: 280, REQ >
                 { Origin-Host }
                 { Origin-Realm }
                 [Origin-State-Id]

5.5.2. Device-Watchdog-Answer

   The Device-Watchdog-Answer (DWA), indicated by the Command-Code set
   to 280 and the Command Flags' 'R' bit cleared, is sent as a response
   to the Device-Watchdog-Request message.

   Message Format

        <DWA> ::= < Diameter Header: 280 >
                   { Result-Code }
                   { Origin-Host }
                   { Origin-Realm }
                   [Error-Message]
                 * [Failed-AVP]
                   [Original-State-Id]

5.5.3. Transport Failure Algorithm

   The transport failure algorithm is defined in [AAATRANS]. All
   Diameter implementations MUST support the algorithm defined in the
   specification in order to be compliant to the Diameter base protocol.

5.5.4. Failover and Failback Procedures

   In the event that a transport failure is detected with a peer, it is
   necessary for all pending request messages to be forwarded to an
   alternate agent, if possible. This is commonly referred to as
   failover.

   In order for a Diameter node to perform failover procedures, it is
   necessary for the node to maintain a pending message queue for a
   given peer. When an answer message is received, the corresponding
   request is removed from the queue. The Hop-by-Hop Identifier field
   is used to match the answer with the queued request.

   When a transport failure is detected, if possible all messages in the
   queue are sent to an alternate agent with the T flag set. On booting
   a Diameter client or agent, the T flag is also set on any records
   still remaining to be transmitted in non-volatile storage. An
   example of a case where it is not possible to forward the message to
   an alternate server is when the message has a fixed destination, and
   the unavailable peer is the message's final destination (see
   Destination-Host AVP). Such an error requires that the agent return
   an answer message with the 'E' bit set and the Result-Code AVP set to
   DIAMETER_UNABLE_TO_DELIVER.

   It is important to note that multiple identical requests or answers
   MAY be received as a result of a failover. The End-to-End Identifier
   field in the Diameter header along with the Origin-Host AVP MUST be
   used to identify duplicate messages.

As described in Section 2.1, a connection request should be
periodically attempted with the failed peer in order to re-establish
the transport connection. Once a connection has been successfully
established, messages can once again be forwarded to the peer. This
is commonly referred to as failback.

5.6. Peer State Machine

This section contains a finite state machine that MUST be observed by
all Diameter implementations. Each Diameter node MUST follow the
state machine described below when communicating with each peer.
Multiple actions are separated by commas, and may continue on
succeeding lines, as space requires. Similarly, state and next state
may also span multiple lines, as space requires.

This state machine is closely coupled with the state machine
described in [AAATRANS], which is used to open, close, failover,
probe, and reopen transport connections. Note in particular that
[AAATRANS] requires the use of watchdog messages to probe
connections. For Diameter, DWR and DWA messages are to be used.

I- is used to represent the initiator (connecting) connection, while
the R- is used to represent the responder (listening) connection.
The lack of a prefix indicates that the event or action is the same
regardless of the connection on which the event occurred.

The stable states that a state machine may be in are Closed, I-Open
and R-Open; all other states are intermediate. Note that I-Open and
R-Open are equivalent except for whether the initiator or responder
transport connection is used for communication.

A CER message is always sent on the initiating connection immediately
after the connection request is successfully completed. In the case
of an election, one of the two connections will shut down. The
responder connection will survive if the Origin-Host of the local
Diameter entity is higher than that of the peer; the initiator
connection will survive if the peer's Origin-Host is higher. All
subsequent messages are sent on the surviving connection. Note that
the results of an election on one peer are guaranteed to be the
inverse of the results on the other.

For TLS usage, a TLS handshake will begin when both ends are in the
open state. If the TLS handshake is successful, all further messages
will be sent via TLS. If the handshake fails, both ends move to the
closed state.

The state machine constrains only the behavior of a Diameter
implementation as seen by Diameter peers through events on the wire.

Any implementation that produces equivalent results is considered
compliant.

```
     state              event          action         next state
     -----------------------------------------------------------------
     Closed             Start          I-Snd-Conn-Req Wait-Conn-Ack
                        R-Conn-CER     R-Accept,      R-Open
                                       Process-CER,
                                       R-Snd-CEA


     Wait-Conn-Ack      I-Rcv-Conn-Ack I-Snd-CER      Wait-I-CEA
                        I-Rcv-Conn-Nack Cleanup       Closed
                        R-Conn-CER     R-Accept,      Wait-Conn-Ack/
                                       Process-CER    Elect
                        Timeout        Error          Closed


     Wait-I-CEA         I-Rcv-CEA      Process-CEA    I-Open
                        R-Conn-CER     R-Accept,      Wait-Returns
                                       Process-CER,
                                       Elect
                        I-Peer-Disc    I-Disc         Closed
                        I-Rcv-Non-CEA  Error          Closed
                        Timeout        Error          Closed


     Wait-Conn-Ack/     I-Rcv-Conn-Ack I-Snd-CER,Elect Wait-Returns
     Elect              I-Rcv-Conn-Nack R-Snd-CEA      R-Open
                        R-Peer-Disc    R-Disc         Wait-Conn-Ack
                        R-Conn-CER     R-Reject       Wait-Conn-Ack/
                                                      Elect
                        Timeout        Error          Closed


     Wait-Returns       Win-Election   I-Disc,R-Snd-CEA R-Open
                        I-Peer-Disc    I-Disc,        R-Open
                                       R-Snd-CEA
                        I-Rcv-CEA      R-Disc         I-Open
                        R-Peer-Disc    R-Disc         Wait-I-CEA
                        R-Conn-CER     R-Reject       Wait-Returns
                        Timeout        Error          Closed


     R-Open             Send-Message   R-Snd-Message  R-Open
                        R-Rcv-Message  Process        R-Open
                        R-Rcv-DWR      Process-DWR,   R-Open
                                       R-Snd-DWA
                        R-Rcv-DWA      Process-DWA    R-Open
                        R-Conn-CER     R-Reject       R-Open
                        Stop           R-Snd-DPR      Closing
                        R-Rcv-DPR      R-Snd-DPA,     Closed
                                          R-Disc
```

```
                      R-Peer-Disc        R-Disc         Closed
                      R-Rcv-CER          R-Snd-CEA      R-Open
                      R-Rcv-CEA          Process-CEA    R-Open

        I-Open        Send-Message       I-Snd-Message  I-Open
                      I-Rcv-Message      Process        I-Open
                      I-Rcv-DWR          Process-DWR,    I-Open
                                         I-Snd-DWA
                      I-Rcv-DWA          Process-DWA    I-Open
                      R-Conn-CER         R-Reject       I-Open
                      Stop               I-Snd-DPR      Closing
                      I-Rcv-DPR          I-Snd-DPA,     Closed
                                         I-Disc
                      I-Peer-Disc        I-Disc         Closed
                      I-Rcv-CER          I-Snd-CEA      I-Open
                      I-Rcv-CEA          Process-CEA    I-Open

        Closing       I-Rcv-DPA          I-Disc         Closed
                      R-Rcv-DPA          R-Disc         Closed
                      Timeout            Error          Closed
                      I-Peer-Disc        I-Disc         Closed
                      R-Peer-Disc        R-Disc         Closed
```

5.6.1. Incoming connections

   When a connection request is received from a Diameter peer, it is
   not, in the general case, possible to know the identity of that peer
   until a CER is received from it. This is because host and port
   determine the identity of a Diameter peer; and the source port of an
   incoming connection is arbitrary. Upon receipt of CER, the identity
   of the connecting peer can be uniquely determined from Origin-Host.

   For this reason, a Diameter peer must employ logic separate from the
   state machine to receive connection requests, accept them, and await
   CER. Once CER arrives on a new connection, the Origin-Host that
   identifies the peer is used to locate the state machine associated
   with that peer, and the new connection and CER are passed to the
   state machine as an R-Conn-CER event.

   The logic that handles incoming connections SHOULD close and discard
   the connection if any message other than CER arrives, or if an
   implementation-defined timeout occurs prior to receipt of CER.

   Because handling of incoming connections up to and including receipt
   of CER requires logic, separate from that of any individual state
   machine associated with a particular peer, it is described separately
   in this section rather than in the state machine above.

5.6.2. Events

   Transitions and actions in the automaton are caused by events. In
   this section, we will ignore the -I and -R prefix, since the actual
   event would be identical, but would occur on one of two possible
   connections.

   Start   The Diameter application has signaled that a
                  connection should be initiated with the peer.

   R-Conn-CER   An acknowledgement is received stating that the
                  transport connection has been established, and the
                  associated CER has arrived.

   Rcv-Conn-Ack   A positive acknowledgement is received confirming that
                  the transport connection is established.

   Rcv-Conn-Nack A negative acknowledgement was received stating that
                  the transport connection was not established.

   Timeout   An application-defined timer has expired while waiting
                  for some event.

   Rcv-CER   A CER message from the peer was received.

   Rcv-CEA   A CEA message from the peer was received.

   Rcv-Non-CEA   A message other than CEA from the peer was received.

   Peer-Disc   A disconnection indication from the peer was received.

   Rcv-DPR   A DPR message from the peer was received.

   Rcv-DPA   A DPA message from the peer was received.

   Win-Election   An election was held, and the local node was the
                  winner.

   Send-Message   A message is to be sent.

   Rcv-Message   A message other than CER, CEA, DPR, DPA, DWR or DWA
                  was received.

   Stop   The Diameter application has signaled that a
                  connection should be terminated (e.g., on system
                  shutdown).

5.6.3. Actions

   Actions in the automaton are caused by events and typically indicate
   the transmission of packets and/or an action to be taken on the
   connection. In this section we will ignore the I- and R-prefix,
   since the actual action would be identical, but would occur on one of
   two possible connections.

   Snd-Conn-Req   A transport connection is initiated with the peer.

   Accept   The incoming connection associated with the R-Conn-CER
                   is accepted as the responder connection.

   Reject   The incoming connection associated with the R-Conn-CER
                   is disconnected.

   Process-CER   The CER associated with the R-Conn-CER is processed.

   Snd-CER   A CER message is sent to the peer.

   Snd-CEA   A CEA message is sent to the peer.

   Cleanup   If necessary, the connection is shutdown, and any
                   local resources are freed.

   Error   The transport layer connection is disconnected, either
                   politely or abortively, in response to an error
                   condition. Local resources are freed.

   Process-CEA   A received CEA is processed.

   Snd-DPR   A DPR message is sent to the peer.

   Snd-DPA   A DPA message is sent to the peer.

   Disc   The transport layer connection is disconnected, and
                   local resources are freed.

   Elect   An election occurs (see Section 5.6.4 for more
                   information).

   Snd-Message   A message is sent.

   Snd-DWR   A DWR message is sent.

   Snd-DWA   A DWA message is sent.

   Process-DWR   The DWR message is serviced.

   Process-DWA   The DWA message is serviced.

   Process   A message is serviced.

5.6.4. The Election Process

   The election is performed on the responder. The responder compares
   the Origin-Host received in the CER sent by its peer with its own
   Origin-Host. If the local Diameter entity's Origin-Host is higher
   than the peer's, a Win-Election event is issued locally.

   The comparison proceeds by considering the shorter OctetString to be
   padded with zeros so that it length is the same as the length of the
   longer, then performing an octet-by-octet unsigned comparison with
   the first octet being most significant. Any remaining octets are
   assumed to have value 0x80.

6. Diameter message processing

   This section describes how Diameter requests and answers are created
   and processed.

6.1. Diameter Request Routing Overview

   A request is sent towards its final destination using a combination
   of the Destination-Realm and Destination-Host AVPs, in one of these
   three combinations:

   - a request that is not able to be proxied (such as CER) MUST NOT
     contain either Destination-Realm or Destination-Host AVPs.

   - a request that needs to be sent to a home server serving a
     specific realm, but not to a specific server (such as the first
     request of a series of round-trips), MUST contain a Destination-
     Realm AVP, but MUST NOT contain a Destination-Host AVP.

   - otherwise, a request that needs to be sent to a specific home
     server among those serving a given realm, MUST contain both the
     Destination-Realm and Destination-Host AVPs.

   The Destination-Host AVP is used as described above when the
   destination of the request is fixed, which includes:

   - Authentication requests that span multiple round trips

   - A Diameter message that uses a security mechanism that makes use
     of a pre-established session key shared between the source and the
     final destination of the message.

- Server initiated messages that MUST be received by a specific
   Diameter client (e.g., access device), such as the Abort-Session-
   Request message, which is used to request that a particular user's
   session be terminated.

Note that an agent can forward a request to a host described in the
Destination-Host AVP only if the host in question is included in its
peer table (see Section 2.7). Otherwise, the request is routed based
on the Destination-Realm only (see Sections 6.1.6).

The Destination-Realm AVP MUST be present if the message is
proxiable. Request messages that may be forwarded by Diameter agents
(proxies, redirects or relays) MUST also contain an Acct-
Application-Id AVP, an Auth-Application-Id AVP or a Vendor-Specific-
Application-Id AVP. A message that MUST NOT be forwarded by Diameter
agents (proxies, redirects or relays) MUST not include the
Destination-Realm in its ABNF. The value of the Destination-Realm
AVP MAY be extracted from the User-Name AVP, or other application-
specific methods.

When a message is received, the message is processed in the following
order:

1. If the message is destined for the local host, the procedures
   listed in Section 6.1.4 are followed.

2. If the message is intended for a Diameter peer with whom the local
   host is able to directly communicate, the procedures listed in
   Section 6.1.5 are followed. This is known as Request Forwarding.

3. The procedures listed in Section 6.1.6 are followed, which is
   known as Request Routing.

4. If none of the above is successful, an answer is returned with the
   Result-Code set to DIAMETER_UNABLE_TO_DELIVER, with the E-bit set.

For routing of Diameter messages to work within an administrative
domain, all Diameter nodes within the realm MUST be peers.

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 comply with the protocol specification. See Section
7 for more detail on error handling.

6.1.1. Originating a Request

   When creating a request, in addition to any other procedures
   described in the application definition for that specific request,
   the following procedures MUST be followed:

   - the Command-Code is set to the appropriate value

   - the 'R' bit is set

   - the End-to-End Identifier is set to a locally unique value

   - the Origin-Host and Origin-Realm AVPs MUST be set to the
      appropriate values, used to identify the source of the message

   - the Destination-Host and Destination-Realm AVPs MUST be set to the
      appropriate values as described in Section 6.1.

   - an Acct-Application-Id AVP, an Auth-Application-Id or a Vendor-
      Specific-Application-Id AVP must be included if the request is
      proxiable.

6.1.2. Sending a Request

   When sending a request, originated either locally, or as the result
   of a forwarding or routing operation, the following procedures MUST
   be followed:

   - the Hop-by-Hop Identifier should be set to a locally unique value

   - The message should be saved in the list of pending requests.

   Other actions to perform on the message based on the particular role
   the agent is playing are described in the following sections.

6.1.3. Receiving Requests

   A relay or proxy agent MUST check for forwarding loops when receiving
   requests. A loop is detected if the server finds its own identity in
   a Route-Record AVP. When such an event occurs, the agent MUST answer
   with the Result-Code AVP set to DIAMETER_LOOP_DETECTED.

6.1.4. Processing Local Requests

   A request is known to be for local consumption when one of the
   following conditions occur:

   - The Destination-Host AVP contains the local host's identity,

- The Destination-Host AVP is not present, the Destination-Realm AVP
  contains a realm the server is configured to process locally, and
  the Diameter application is locally supported, or

- Both the Destination-Host and the Destination-Realm are not
  present.

When a request is locally processed, the rules in Section 6.2 should
be used to generate the corresponding answer.

6.1.5. Request Forwarding

Request forwarding is done using the Diameter Peer Table. The
Diameter peer table contains all of the peers that the local node is
able to directly communicate with.

When a request is received, and the host encoded in the Destination-
Host AVP is one that is present in the peer table, the message SHOULD
be forwarded to the peer.

6.1.6. Request Routing

Diameter request message routing is done via realms and applications.
A Diameter message that may be forwarded by Diameter agents (proxies,
redirects or relays) MUST include the target realm in the
Destination-Realm AVP and one of the application identification AVPs
Auth-Application-Id, Acct-Application-Id or Vendor-Specific-
Application-Id. The realm MAY be retrieved from the User-Name AVP,
which is in the form of a Network Access Identifier (NAI). The realm
portion of the NAI is inserted in the Destination-Realm AVP.

Diameter agents MAY have a list of locally supported realms and
applications, and MAY have a list of externally supported realms and
applications. When a request is received that includes a realm
and/or application that is not locally supported, the message is
routed to the peer configured in the Realm Routing Table (see Section
2.7).

6.1.7. Redirecting requests

When a redirect agent receives a request whose routing entry is set
to REDIRECT, it MUST reply with an answer message with the 'E' bit
set, while maintaining the Hop-by-Hop Identifier in the header, and
include the Result-Code AVP to DIAMETER_REDIRECT_INDICATION. Each of
the servers associated with the routing entry are added in separate
Redirect-Host AVP.

```
                     +------------------+
                     |     Diameter     |
                     |  Redirect Agent  |
                     +------------------+
                       ^     |    2. command + 'E' bit
    1. Request         |     |    Result-Code =
  joe@example.com |    |     |    DIAMETER_REDIRECT_INDICATION +
                  |    |     |    Redirect-Host AVP(s)
                  |    v
            +-------------+  3. Request  +-------------+
            | example.com |------------->| example.net |
            |    Relay    |              |   Diameter  |
            |    Agent    |<-------------|    Server   |
            +-------------+  4. Answer   +-------------+
```
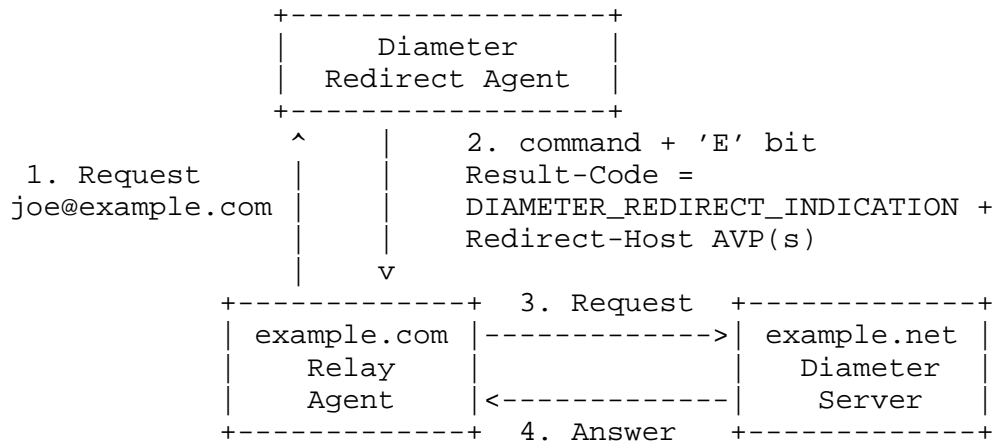
                  Figure 5: Diameter Redirect Agent

   The receiver of the answer message with the 'E' bit set, and the
   Result-Code AVP set to DIAMETER_REDIRECT_INDICATION uses the hop-by-
   hop field in the Diameter header to identify the request in the
   pending message queue (see Section 5.3) that is to be redirected. If
   no transport connection exists with the new agent, one is created,
   and the request is sent directly to it.

   Multiple Redirect-Host AVPs are allowed. The receiver of the answer
   message with the 'E' bit set selects exactly one of these hosts as
   the destination of the redirected message.

6.1.8. Relaying and Proxying Requests

   A relay or proxy agent MUST append a Route-Record AVP to all requests
   forwarded. The AVP contains the identity of the peer the request was
   received from.

   The Hop-by-Hop identifier in the request is saved, and replaced with
   a locally unique value. The source of the request is also saved,
   which includes the IP address, port and protocol.

   A relay or proxy agent MAY include the Proxy-Info AVP in requests if
   it requires access to any local state information when the
   corresponding response is received. Proxy-Info AVP has certain
   security implications and SHOULD contain an embedded HMAC with a
   node-local key. Alternatively, it MAY simply use local storage to
   store state information.

   The message is then forwarded to the next hop, as identified in the
   Realm Routing Table.

Figure 6 provides an example of message routing using the procedures
listed in these sections.

```
     (Origin-Host=nas.mno.net)      (Origin-Host=nas.mno.net)
     (Origin-Realm=mno.net)         (Origin-Realm=mno.net)
     (Destination-Realm=example.com) (Destination-
                                        Realm=example.com)
                                    (Route-Record=nas.example.net)
  +------+     ------>     +------+     ------>     +------+
  |      |    (Request)    |      |    (Request)    |      |
  | NAS  +-----------------+ DRL  +-----------------+ HMS  |
  |      |                 |      |                 |      |
  +------+     <------      +------+     <------      +------+
 example.net   (Answer)   example.net   (Answer)   example.com
    (Origin-Host=hms.example.com)   (Origin-Host=hms.example.com)
    (Origin-Realm=example.com)      (Origin-Realm=example.com)
```

Figure 6: Routing of Diameter messages

6.2. Diameter Answer Processing

   When a request is locally processed, the following procedures MUST be
   applied to create the associated answer, in addition to any
   additional procedures that MAY be discussed in the Diameter
   application defining the command:

   - The same Hop-by-Hop identifier in the request is used in the
      answer.

   - The local host's identity is encoded in the Origin-Host AVP.

   - The Destination-Host and Destination-Realm AVPs MUST NOT be
      present in the answer message.

   - The Result-Code AVP is added with its value indicating success or
      failure.

   - If the Session-Id is present in the request, it MUST be included
      in the answer.

   - Any Proxy-Info AVPs in the request MUST be added to the answer
      message, in the same order they were present in the request.

   - The 'P' bit is set to the same value as the one in the request.

   - The same End-to-End identifier in the request is used in the
      answer.

Note that the error messages (see Section 7.3) are also subjected to
the above processing rules.

6.2.1. Processing received Answers

A Diameter client or proxy MUST match the Hop-by-Hop Identifier in an
answer received against the list of pending requests. The
corresponding message should be removed from the list of pending
requests. It SHOULD ignore answers received that do not match a
known Hop-by-Hop Identifier.

6.2.2. Relaying and Proxying Answers

If the answer is for a request which was proxied or relayed, the
agent MUST restore the original value of the Diameter header's Hop-
by-Hop Identifier field.

If the last Proxy-Info AVP in the message is targeted to the local
Diameter server, the AVP MUST be removed before the answer is
forwarded.

If a relay or proxy agent receives an answer with a Result-Code AVP
indicating a failure, it MUST NOT modify the contents of the AVP.
Any additional local errors detected SHOULD be logged, but not
reflected in the Result-Code AVP. If the agent receives an answer
message with a Result-Code AVP indicating success, and it wishes to
modify the AVP to indicate an error, it MUST modify the Result-Code
AVP to contain the appropriate error in the message destined towards
the access device as well as include the Error-Reporting-Host AVP and
it MUST issue an STR on behalf of the access device.

The agent MUST then send the answer to the host that it received the
original request from.

6.3. Origin-Host AVP

The Origin-Host AVP (AVP Code 264) is of type DiameterIdentity, and
MUST be present in all Diameter messages. This AVP identifies the
endpoint that originated the Diameter message. Relay agents MUST NOT
modify this AVP.

The value of the Origin-Host AVP is guaranteed to be unique within a
single host.

Note that the Origin-Host AVP may resolve to more than one address as
the Diameter peer may support more than one address.

This AVP SHOULD be placed as close to the Diameter header as
possible. 6.10

6.4. Origin-Realm AVP

The Origin-Realm AVP (AVP Code 296) is of type DiameterIdentity.
This AVP contains the Realm of the originator of any Diameter message
and MUST be present in all messages.

This AVP SHOULD be placed as close to the Diameter header as
possible.

6.5. Destination-Host AVP

The Destination-Host AVP (AVP Code 293) is of type DiameterIdentity.
This AVP MUST be present in all unsolicited agent initiated messages,
MAY be present in request messages, and MUST NOT be present in Answer
messages.

The absence of the Destination-Host AVP will cause a message to be
sent to any Diameter server supporting the application within the
realm specified in Destination-Realm AVP.

This AVP SHOULD be placed as close to the Diameter header as
possible.

6.6. Destination-Realm AVP

The Destination-Realm AVP (AVP Code 283) is of type DiameterIdentity,
and contains the realm the message is to be routed to. The
Destination-Realm AVP MUST NOT be present in Answer messages.
Diameter Clients insert the realm portion of the User-Name AVP.
Diameter servers initiating a request message use the value of the
Origin-Realm AVP from a previous message received from the intended
target host (unless it is known a priori). When present, the
Destination-Realm AVP is used to perform message routing decisions.

Request messages whose ABNF does not list the Destination-Realm AVP
as a mandatory AVP are inherently non-routable messages.

This AVP SHOULD be placed as close to the Diameter header as
possible.

6.7. Routing AVPs

The AVPs defined in this section are Diameter AVPs used for routing
purposes. These AVPs change as Diameter messages are processed by
agents, and therefore MUST NOT be protected by end-to-end security.

6.7.1. Route-Record AVP

   The Route-Record AVP (AVP Code 282) is of type DiameterIdentity. The
   identity added in this AVP MUST be the same as the one received in
   the Origin-Host of the Capabilities Exchange message.

6.7.2. Proxy-Info AVP

   The Proxy-Info AVP (AVP Code 284) is of type Grouped. The Grouped
   Data field has the following ABNF grammar:

      Proxy-Info ::= < AVP Header: 284 >
                     { Proxy-Host }
                     { Proxy-State }
                   * [AVP]

6.7.3. Proxy-Host AVP

   The Proxy-Host AVP (AVP Code 280) is of type DiameterIdentity. This
   AVP contains the identity of the host that added the Proxy-Info AVP.

6.7.4. Proxy-State AVP

   The Proxy-State AVP (AVP Code 33) is of type OctetString, and
   contains state local information, and MUST be treated as opaque data.

6.8. Auth-Application-Id AVP

   The Auth-Application-Id AVP (AVP Code 258) is of type Unsigned32 and
   is used in order to advertise support of the Authentication and
   Authorization portion of an application (see Section 2.4). The
   Auth-Application-Id MUST also be present in all Authentication and/or
   Authorization messages that are defined in a separate Diameter
   specification and have an Application ID assigned.

6.9. Acct-Application-Id AVP

   The Acct-Application-Id AVP (AVP Code 259) is of type Unsigned32 and
   is used in order to advertise support of the Accounting portion of an
   application (see Section 2.4). The Acct-Application-Id MUST also be
   present in all Accounting messages. Exactly one of the Auth-
   Application-Id and Acct-Application-Id AVPs MAY be present.

6.10. Inband-Security-Id AVP

   The Inband-Security-Id AVP (AVP Code 299) is of type Unsigned32 and
   is used in order to advertise support of the Security portion of the
   application.

Currently, the following values are supported, but there is ample
room to add new security Ids.

NO_INBAND_SECURITY   0
   This peer does not support TLS. This is the default value, if the
   AVP is omitted.

TLS   1
   This node supports TLS security, as defined by [TLS].

6.11. Vendor-Specific-Application-Id AVP

The Vendor-Specific-Application-Id AVP (AVP Code 260) is of type
Grouped and is used to advertise support of a vendor-specific
Diameter Application. Exactly one of the Auth-Application-Id and
Acct-Application-Id AVPs MAY be present.

This AVP MUST also be present as the first AVP in all experimental
commands defined in the vendor-specific application.

This AVP SHOULD be placed as close to the Diameter header as
possible.

AVP Format

```
<Vendor-Specific-Application-Id> ::= < AVP Header: 260 >
                                  1* [ Vendor-Id ]
                                  0*1{ Auth-Application-Id }
                                  0*1{ Acct-Application-Id }
```

6.12. Redirect-Host AVP

One or more of instances of this AVP MUST be present if the answer
message's 'E' bit is set and the Result-Code AVP is set to
DIAMETER_REDIRECT_INDICATION.

Upon receiving the above, the receiving Diameter node SHOULD forward
the request directly to one of the hosts identified in these AVPs.
The server contained in the selected Redirect-Host AVP SHOULD be used
for all messages pertaining to this session.

6.13. Redirect-Host-Usage AVP

The Redirect-Host-Usage AVP (AVP Code 261) is of type Enumerated.
This AVP MAY be present in answer messages whose 'E' bit is set and
the Result-Code AVP is set to DIAMETER_REDIRECT_INDICATION.

When present, this AVP dictates how the routing entry resulting from
the Redirect-Host is to be used. The following values are supported:

   DONT_CACHE   0
      The host specified in the Redirect-Host AVP should not be cached.
      This is the default value.

   ALL_SESSION   1
      All messages within the same session, as defined by the same value
      of the Session-ID AVP MAY be sent to the host specified in the
      Redirect-Host AVP.

   ALL_REALM   2
      All messages destined for the realm requested MAY be sent to the
      host specified in the Redirect-Host AVP.

   REALM_AND_APPLICATION   3
      All messages for the application requested to the realm specified
      MAY be sent to the host specified in the Redirect-Host AVP.

   ALL_APPLICATION   4
      All messages for the application requested MAY be sent to the host
      specified in the Redirect-Host AVP.

   ALL_HOST   5
      All messages that would be sent to the host that generated the
      Redirect-Host MAY be sent to the host specified in the Redirect-
      Host AVP.

   ALL_USER   6
      All messages for the user requested MAY be sent to the host
      specified in the Redirect-Host AVP.

6.14. Redirect-Max-Cache-Time AVP

   The Redirect-Max-Cache-Time AVP (AVP Code 262) is of type Unsigned32.
   This AVP MUST be present in answer messages whose 'E' bit is set, the
   Result-Code AVP is set to DIAMETER_REDIRECT_INDICATION and the
   Redirect-Host-Usage AVP set to a non-zero value.

   This AVP contains the maximum number of seconds the peer and route
   table entries, created as a result of the Redirect-Host, will be
   cached. Note that once a host created due to a redirect indication
   is no longer reachable, any associated peer and routing table entries
   MUST be deleted.

6.15. E2E-Sequence AVP

   The E2E-Sequence AVP (AVP Code 300) provides anti-replay protection
   for end to end messages and is of type grouped. It contains a random
   value (an OctetString with a nonce) and counter (an Integer). For
   each end-to-end peer with which a node communicates (or remembers
   communicating) a different nonce value MUST be used and the counter
   is initiated at zero and increases by one each time this AVP is
   emitted to that peer. This AVP MUST be included in all messages
   which use end-to-end protection (e.g., CMS signing or encryption).

7. Error Handling

   There are two different types of errors in Diameter; protocol and
   application errors. A protocol error is one that occurs at the base
   protocol level, and MAY require per hop attention (e.g., message
   routing error). Application errors, on the other hand, generally
   occur due to a problem with a function specified in a Diameter
   application (e.g., user authentication, Missing AVP).

   Result-Code AVP values that are used to report protocol errors MUST
   only be present in answer messages whose 'E' bit is set. When a
   request message is received that causes a protocol error, an answer
   message is returned with the 'E' bit set, and the Result-Code AVP is
   set to the appropriate protocol error value. As the answer is sent
   back towards the originator of the request, each proxy or relay agent
   MAY take action on the message.

```
                       1. Request            +---------+ Link Broken
             +--------------------------->|Diameter |----///----+
             |   +---------------------|         |           |
   +------+--+   | 2. answer + 'E' set | Relay 2 |           v
   |Diameter |<-+ (Unable to Forward) +---------+        +--------+
   |         |   |                                       |Diameter|
   |         |   |                                       | Home   |
   | Relay 1 |--+                      +---------+       | Server |
   +---------+   |  3. Request         |Diameter |       +--------+
                +-------------------->|         |           ^
                                       | Relay 3 |-----------+
                                       +---------+
```

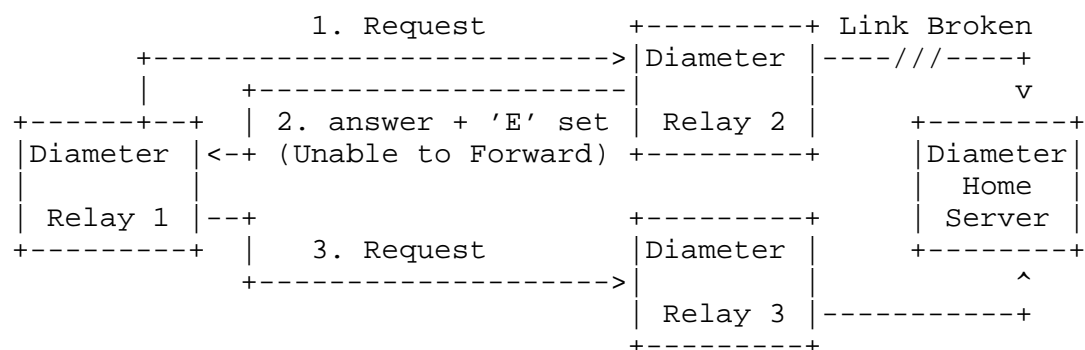   Figure 7: Example of Protocol Error causing answer message

   Figure 7 provides an example of a message forwarded upstream by a
   Diameter relay. When the message is received by Relay 2, and it
   detects that it cannot forward the request to the home server, an
   answer message is returned with the 'E' bit set and the Result-Code
   AVP set to DIAMETER_UNABLE_TO_DELIVER. Given that this error falls

within the protocol error category, Relay 1 would take special
action, and given the error, attempt to route the message through its
alternate Relay 3.

```
      +---------+ 1. Request  +---------+ 2. Request  +---------+
      | Access  |------------>|Diameter |------------>|Diameter |
      |         |             |         |             |  Home   |
      | Device  |<------------|  Relay  |<------------| Server  |
      +---------+ 4. Answer   +---------+ 3. Answer   +---------+
             (Missing AVP)               (Missing AVP)
```
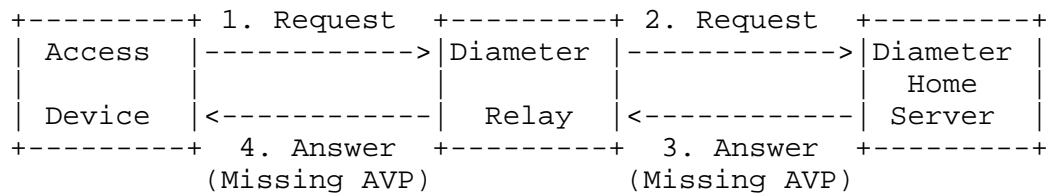
           Figure 8: Example of Application Error Answer message

Figure 8 provides an example of a Diameter message that caused an
application error. When application errors occur, the Diameter
entity reporting the error clears the 'R' bit in the Command Flags,
and adds the Result-Code AVP with the proper value. Application
errors do not require any proxy or relay agent involvement, and
therefore the message would be forwarded back to the originator of
the request.

There are certain Result-Code AVP application errors that require
additional AVPs to be present in the answer. In these cases, the
Diameter node that sets the Result-Code AVP to indicate the error
MUST add the AVPs. Examples are:

- An unrecognized AVP is received with the 'M' bit (Mandatory bit)
   set, causes an answer to be sent with the Result-Code AVP set to
   DIAMETER_AVP_UNSUPPORTED, and the Failed-AVP AVP containing the
   offending AVP.

- An AVP that is received with an unrecognized value causes an
   answer to be returned with the Result-Code AVP set to
   DIAMETER_INVALID_AVP_VALUE, with the Failed-AVP AVP containing the
   AVP causing the error.

- A command is received with an AVP that is omitted, yet is
   mandatory according to the command's ABNF. The receiver issues an
   answer with the Result-Code set to DIAMETER_MISSING_AVP, and
   creates an AVP with the AVP Code and other fields set as expected
   in the missing AVP. The created AVP is then added to the Failed-
   AVP AVP.

The Result-Code AVP describes the error that the Diameter node
encountered in its processing. In case there are multiple errors,
the Diameter node MUST report only the first error it encountered

(detected possibly in some implementation dependent order). The
specific errors that can be described by this AVP are described in
the following section.

7.1. Result-Code AVP

The Result-Code AVP (AVP Code 268) is of type Unsigned32 and
indicates whether a particular request was completed successfully or
whether an error occurred. All Diameter answer messages defined in
IETF applications MUST include one Result-Code AVP. A non-successful
Result-Code AVP (one containing a non 2xxx value other than
DIAMETER_REDIRECT_INDICATION) MUST include the Error-Reporting-Host
AVP if the host setting the Result-Code AVP is different from the
identity encoded in the Origin-Host AVP.

The Result-Code data field contains an IANA-managed 32-bit address
space representing errors (see Section 11.4). Diameter provides the
following classes of errors, all identified by the thousands digit in
the decimal notation:

   - 1xxx (Informational)
   - 2xxx (Success)
   - 3xxx (Protocol Errors)
   - 4xxx (Transient Failures)
   - 5xxx (Permanent Failure)

A non-recognized class (one whose first digit is not defined in this
section) MUST be handled as a permanent failure.

7.1.1. Informational

Errors that fall within this category are used to inform the
requester that a request could not be satisfied, and additional
action is required on its part before access is granted.

DIAMETER_MULTI_ROUND_AUTH   1001
   This informational error is returned by a Diameter server to
   inform the access device that the authentication mechanism being
   used requires multiple round trips, and a subsequent request needs
   to be issued in order for access to be granted.

7.1.2. Success

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

DIAMETER_SUCCESS   2001
   The Request was successfully completed.

   DIAMETER_LIMITED_SUCCESS   2002
      When returned, the request was successfully completed, but
      additional processing is required by the application in order to
      provide service to the user.

7.1.3. Protocol Errors

   Errors that fall within the Protocol Error category SHOULD be treated
   on a per-hop basis, and Diameter proxies MAY attempt to correct the
   error, if it is possible. Note that these and only these errors MUST
   only be used in answer messages whose 'E' bit is set.

   DIAMETER_COMMAND_UNSUPPORTED   3001
      The Request contained a Command-Code that the receiver did not
      recognize or support. This MUST be used when a Diameter node
      receives an experimental command that it does not understand.

   DIAMETER_UNABLE_TO_DELIVER   3002
      This error is given when Diameter can not deliver the message to
      the destination, either because no host within the realm
      supporting the required application was available to process the
      request, or because Destination-Host AVP was given without the
      associated Destination-Realm AVP.

   DIAMETER_REALM_NOT_SERVED   3003
      The intended realm of the request is not recognized.

   DIAMETER_TOO_BUSY   3004
      When returned, a Diameter node SHOULD attempt to send the message
      to an alternate peer. This error MUST only be used when a
      specific server is requested, and it cannot provide the requested
      service.

   DIAMETER_LOOP_DETECTED   3005
      An agent detected a loop while trying to get the message to the
      intended recipient. The message MAY be sent to an alternate peer,
      if one is available, but the peer reporting the error has
      identified a configuration problem.

   DIAMETER_REDIRECT_INDICATION   3006
      A redirect agent 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. When set, the Redirect-Host AVP MUST
      be present.

   DIAMETER_APPLICATION_UNSUPPORTED   3007
      A request was sent for an application that is not supported.

   DIAMETER_INVALID_HDR_BITS   3008
      A request was received whose bits in the Diameter header were
      either set to an invalid combination, or to a value that is
      inconsistent with the command code's definition.

   DIAMETER_INVALID_AVP_BITS   3009
      A request was received that included an AVP whose flag bits are
      set to an unrecognized value, or that is inconsistent with the
      AVP's definition.

   DIAMETER_UNKNOWN_PEER   3010
      A CER was received from an unknown peer.

7.1.4. Transient Failures

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

   DIAMETER_AUTHENTICATION_REJECTED   4001
      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_OUT_OF_SPACE   4002
      A Diameter node received the accounting request but was unable to
      commit it to stable storage due to a temporary lack of space.

   ELECTION_LOST   4003
      The peer has determined that it has lost the election process and
      has therefore disconnected the transport connection.

7.1.5. 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_AVP_UNSUPPORTED   5001
      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_UNKNOWN_SESSION_ID   5002
      The request contained an unknown Session-Id.

DIAMETER_AUTHORIZATION_REJECTED   5003
    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   5004
    The request contained an AVP with an invalid value in its data
    portion. A Diameter message indicating this error MUST include
    the offending AVPs within a Failed-AVP AVP.

DIAMETER_MISSING_AVP   5005
    The request did not contain an AVP that is required 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 Failed-AVP
    AVP MUST contain an example of the missing AVP complete with the
    Vendor-Id if applicable. The value field of the missing AVP
    should be of correct minimum length and contain zeroes.

DIAMETER_RESOURCES_EXCEEDED   5006
    A request was received that cannot be authorized because the user
    has already expended allowed resources. An example of this error
    condition is a user that is restricted to one dial-up PPP port,
    attempts to establish a second PPP connection.

DIAMETER_CONTRADICTING_AVPS   5007
    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 AVPs MUST be present, containing
    the AVPs that contradicted each other.

DIAMETER_AVP_NOT_ALLOWED   5008
    A message was received with an AVP that MUST NOT be present. The
    Failed-AVP AVP MUST be included and contain a copy of the
    offending AVP.

DIAMETER_AVP_OCCURS_TOO_MANY_TIMES 5009
    A message was received that included an AVP that appeared more
    often than permitted in the message definition. The Failed-AVP
    AVP MUST be included and contain a copy of the first instance of
    the offending AVP that exceeded the maximum number of occurrences

DIAMETER_NO_COMMON_APPLICATION   5010
    This error is returned when a CER message is received, and there
    are no common applications supported between the peers.

DIAMETER_UNSUPPORTED_VERSION   5011
    This error is returned when a request was received, whose version
    number is unsupported.

DIAMETER_UNABLE_TO_COMPLY   5012
   This error is returned when a request is rejected for unspecified
   reasons.

DIAMETER_INVALID_BIT_IN_HEADER   5013
   This error is returned when an unrecognized bit in the Diameter
   header is set to one (1).

DIAMETER_INVALID_AVP_LENGTH   5014
   The request contained an AVP with an invalid length. A Diameter
   message indicating this error MUST include the offending AVPs
   within a Failed-AVP AVP.

DIAMETER_INVALID_MESSAGE_LENGTH   5015
   This error is returned when a request is received with an invalid
   message length.

DIAMETER_INVALID_AVP_BIT_COMBO   5016
   The request contained an AVP with which is not allowed to have the
   given value in the AVP Flags field. A Diameter message indicating
   this error MUST include the offending AVPs within a Failed-AVP
   AVP.

DIAMETER_NO_COMMON_SECURITY   5017
   This error is returned when a CER message is received, and there
   are no common security mechanisms supported between the peers. A
   Capabilities-Exchange-Answer (CEA) MUST be returned with the
   Result-Code AVP set to DIAMETER_NO_COMMON_SECURITY.

7.2. Error Bit

   The 'E' (Error Bit) in the Diameter header is set when the request
   caused a protocol-related error (see Section 7.1.3). A message with
   the 'E' bit MUST NOT be sent as a response to an answer message.
   Note that a message with the 'E' bit set is still subjected to the
   processing rules defined in Section 6.2. When set, the answer
   message will not conform to the ABNF specification for the command,
   and will instead conform to the following ABNF:

Message Format

```
<answer-message> ::= < Diameter Header: code, ERR [PXY] >
                    0*1< Session-Id >
                        { Origin-Host }
                        { Origin-Realm }
                        { Result-Code }
                        [ Origin-State-Id ]
                        [ Error-Reporting-Host ]
                        [ Proxy-Info ]
                      * [ AVP ]
```

Note that the code used in the header is the same than the one found
in the request message, but with the 'R' bit cleared and the 'E' bit
set. The 'P' bit in the header is set to the same value as the one
found in the request message.

7.3. Error-Message AVP

The Error-Message AVP (AVP Code 281) is of type UTF8String. It MAY
accompany a Result-Code AVP as 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.

7.4. Error-Reporting-Host AVP

The Error-Reporting-Host AVP (AVP Code 294) is of type
DiameterIdentity. This AVP contains the identity of the Diameter
host that sent the Result-Code AVP to a value other than 2001
(Success), only if the host setting the Result-Code is different from
the one encoded in the Origin-Host AVP. This AVP is intended to be
used for troubleshooting purposes, and MUST be set when the Result-
Code AVP indicates a failure.

7.5. Failed-AVP AVP

The Failed-AVP AVP (AVP Code 279) is of type Grouped and provides
debugging information in cases where a request is rejected or not
fully processed due to erroneous information in a specific AVP. The
value of the Result-Code AVP will provide information on the reason
for the Failed-AVP AVP.

The possible reasons for this AVP are the presence of an improperly
constructed AVP, an unsupported or unrecognized AVP, an invalid AVP
value, the omission of a required AVP, the presence of an explicitly
excluded AVP (see tables in Section 10), or the presence of two or
more occurrences of an AVP which is restricted to 0, 1, or 0-1
occurrences.

   A Diameter message MAY contain one Failed-AVP AVP, containing the
   entire AVP that could not be processed successfully. If the failure
   reason is omission of a required AVP, an AVP with the missing AVP
   code, the missing vendor id, and a zero filled payload of the minimum
   required length for the omitted AVP will be added.

   AVP Format

      <Failed-AVP> ::= < AVP Header: 279 >
                    1* {AVP}

7.6. Experimental-Result AVP

   The Experimental-Result AVP (AVP Code 297) is of type Grouped, and
   indicates whether a particular vendor-specific request was completed
   successfully or whether an error occurred. Its Data field has the
   following ABNF grammar:

   AVP Format

      Experimental-Result ::= < AVP Header: 297 >
                                { Vendor-Id }
                                { Experimental-Result-Code }

   The Vendor-Id AVP (see Section 5.3.3) in this grouped AVP identifies
   the vendor responsible for the assignment of the result code which
   follows. All Diameter answer messages defined in vendor-specific
   applications MUST include either one Result-Code AVP or one
   Experimental-Result AVP.

7.7. Experimental-Result-Code AVP

   The Experimental-Result-Code AVP (AVP Code 298) is of type Unsigned32
   and contains a vendor-assigned value representing the result of
   processing the request.

   It is recommended that vendor-specific result codes follow the same
   conventions given for the Result-Code AVP regarding the different
   types of result codes and the handling of errors (for non 2xxx
   values).

8. Diameter User Sessions

   Diameter can provide two different types of services to applications.
   The first involves authentication and authorization, and can
   optionally make use of accounting. The second only makes use of
   accounting.

When a service makes use of the authentication and/or authorization portion of an application, and a user requests access to the network, the Diameter client issues an auth request to its local server. The auth request is defined in a service specific Diameter application (e.g., NASREQ). The request contains a Session-Id AVP, which is used in subsequent messages (e.g., subsequent authorization, accounting, etc) relating to the user's session. The Session-Id AVP is a means for the client and servers to correlate a Diameter message with a user session.

When a Diameter server authorizes a user to use network resources for a finite amount of time, and it is willing to extend the authorization via a future request, it MUST add the Authorization-Lifetime AVP to the answer message. The Authorization-Lifetime AVP defines the maximum number of seconds a user MAY make use of the resources before another authorization request is expected by the server. The Auth-Grace-Period AVP contains the number of seconds following the expiration of the Authorization-Lifetime, after which the server will release all state information related to the user's session. Note that if payment for services is expected by the serving realm from the user's home realm, the Authorization-Lifetime AVP, combined with the Auth-Grace-Period AVP, implies the maximum length of the session the home realm is willing to be fiscally responsible for. Services provided past the expiration of the Authorization-Lifetime and Auth-Grace-Period AVPs are the responsibility of the access device. Of course, the actual cost of services rendered is clearly outside the scope of the protocol.

An access device that does not expect to send a re-authorization or a session termination request to the server MAY include the Auth-Session-State AVP with the value set to NO_STATE_MAINTAINED as a hint to the server. If the server accepts the hint, it agrees that since no session termination message will be received once service to the user is terminated, it cannot maintain state for the session. If the answer message from the server contains a different value in the Auth-Session-State AVP (or the default value if the AVP is absent), the access device MUST follow the server's directives. Note that the value NO_STATE_MAINTAINED MUST NOT be set in subsequent re-authorization requests and answers.

The base protocol does not include any authorization request messages, since these are largely application-specific and are defined in a Diameter application document. However, the base protocol does define a set of messages that is used to terminate user sessions. These are used to allow servers that maintain state information to free resources.

When a service only makes use of the Accounting portion of the
Diameter protocol, even in combination with an application, the
Session-Id is still used to identify user sessions. However, the
session termination messages are not used, since a session is
signaled as being terminated by issuing an accounting stop message.

8.1. Authorization Session State Machine

   This section contains a set of finite state machines, representing
   the life cycle of Diameter sessions, and which MUST be observed by
   all Diameter implementations that make use of the authentication
   and/or authorization portion of a Diameter application. The term
   Service-Specific below refers to a message defined in a Diameter
   application (e.g., Mobile IPv4, NASREQ).

   There are four different authorization session state machines
   supported in the Diameter base protocol. The first two describe a
   session in which the server is maintaining session state, indicated
   by the value of the Auth-Session-State AVP (or its absence). One
   describes the session from a client perspective, the other from a
   server perspective. The second two state machines are used when the
   server does not maintain session state. Here again, one describes
   the session from a client perspective, the other from a server
   perspective.

   When a session is moved to the Idle state, any resources that were
   allocated for the particular session must be released. Any event not
   listed in the state machines MUST be considered as an error
   condition, and an answer, if applicable, MUST be returned to the
   originator of the message.

   In the state table, the event 'Failure to send X' means that the
   Diameter agent is unable to send command X to the desired
   destination. This could be due to the peer being down, or due to the
   peer sending back a transient failure or temporary protocol error
   notification DIAMETER_TOO_BUSY or DIAMETER_LOOP_DETECTED in the
   Result-Code AVP of the corresponding Answer command. The event 'X
   successfully sent' is the complement of 'Failure to send X'.

The following state machine is observed by a client when state is
maintained on the server:

```
                     CLIENT, STATEFUL
   State      Event                          Action       New State
   --------------------------------------------------------------
   Idle       Client or Device Requests      Send         Pending
              access                         service
                                             specific
                                             auth req

   Idle       ASR Received                   Send ASA     Idle
              for unknown session            with
                                             Result-Code
                                             = UNKNOWN_
                                             SESSION_ID

   Pending    Successful Service-specific    Grant        Open
              authorization answer           Access
              received with default
              Auth-Session-State value

   Pending    Successful Service-specific    Sent STR     Discon
              authorization answer received
              but service not provided

   Pending    Error processing successful    Sent STR     Discon
              Service-specific authorization
              answer

   Pending    Failed Service-specific        Cleanup      Idle
              authorization answer received

   Open       User or client device          Send         Open
              requests access to service     service
                                             specific
                                             auth req

   Open       Successful Service-specific    Provide      Open
              authorization answer received  Service

   Open       Failed Service-specific        Discon.      Idle
              authorization answer           user/device
              received.

   Open       Session-Timeout Expires on     Send STR     Discon
              Access Device
```

```
Open       ASR Received,                     Send ASA    Discon
           client will comply with          with
           request to end the session       Result-Code
                                             = SUCCESS,
                                             Send STR.

Open       ASR Received,                     Send ASA    Open
           client will not comply with      with
           request to end the session       Result-Code
                                             != SUCCESS

Open       Authorization-Lifetime +         Send STR    Discon
           Auth-Grace-Period expires on
           access device

Discon     ASR Received                      Send ASA    Discon

Discon     STA Received                      Discon.     Idle
                                             user/device
```

The following state machine is observed by a server when it is
maintaining state for the session:

```
                        SERVER, STATEFUL
State      Event                            Action      New State
-----------------------------------------------------------
Idle       Service-specific authorization   Send        Open
           request received, and            successful
           user is authorized               serv.
                                            specific answer

Idle       Service-specific authorization   Send        Idle
           request received, and            failed serv.
           user is not authorized           specific answer

Open       Service-specific authorization   Send        Open
           request received, and user       successful
           is authorized                    serv. specific
                                                    answer

Open       Service-specific authorization   Send        Idle
           request received, and user       failed serv.
           is not authorized                specific
                                            answer,
                                            Cleanup

Open       Home server wants to             Send ASR    Discon
           terminate the service
```

```
   Open        Authorization-Lifetime (and   Cleanup    Idle
               Auth-Grace-Period) expires
               on home server.

   Open        Session-Timeout expires on    Cleanup    Idle
               home server

   Discon      Failure to send ASR           Wait,      Discon
                                             resend ASR

   Discon      ASR successfully sent and     Cleanup    Idle
               ASA Received with Result-Code

   Not         ASA Received                  None       No Change.
   Discon

   Any         STR Received                  Send STA,  Idle
                                             Cleanup.
```

The following state machine is observed by a client when state is not
maintained on the server:

```
                         CLIENT, STATELESS
   State      Event                         Action     New State
   -------------------------------------------------------------
   Idle       Client or Device Requests     Send       Pending
              access                        service
                                            specific
                                            auth req

   Pending    Successful Service-specific   Grant      Open
              authorization answer          Access
              received with Auth-Session-
              State set to
              NO_STATE_MAINTAINED

   Pending    Failed Service-specific       Cleanup    Idle
              authorization answer
              received

   Open       Session-Timeout Expires on    Discon.    Idle
              Access Device                 user/device

   Open       Service to user is terminated Discon.    Idle
                                            user/device
```

The following state machine is observed by a server when it is not
maintaining state for the session:

```
                        SERVER, STATELESS
   State      Event                           Action     New State
   ----------------------------------------------------------------
   Idle       Service-specific authorization  Send serv. Idle
              request received, and           specific
              successfully processed          answer
```

8.2. Accounting Session State Machine

   The following state machines MUST be supported for applications that
   have an accounting portion or that require only accounting services.
   The first state machine is to be observed by clients.

   See Section 9.7 for Accounting Command Codes and Section 9.8 for
   Accounting AVPs.

   The server side in the accounting state machine depends in some cases
   on the particular application. The Diameter base protocol defines a
   default state machine that MUST be followed by all applications that
   have not specified other state machines. This is the second state
   machine in this section described below.

   The default server side state machine requires the reception of
   accounting records in any order and at any time, and does not place
   any standards requirement on the processing of these records.
   Implementations of Diameter MAY perform checking, ordering,
   correlation, fraud detection, and other tasks based on these records.
   Both base Diameter AVPs as well as application specific AVPs MAY be
   inspected as a part of these tasks. The tasks can happen either
   immediately after record reception or in a post-processing phase.
   However, as these tasks are typically application or even policy
   dependent, they are not standardized by the Diameter specifications.
   Applications MAY define requirements on when to accept accounting
   records based on the used value of Accounting-Realtime-Required AVP,
   credit limits checks, and so on.

   However, the Diameter base protocol defines one optional server side
   state machine that MAY be followed by applications that require
   keeping track of the session state at the accounting server. Note
   that such tracking is incompatible with the ability to sustain long
   duration connectivity problems. Therefore, the use of this state
   machine is recommended only in applications where the value of the
   Accounting-Realtime-Required AVP is DELIVER_AND_GRANT, and hence
   accounting connectivity problems are required to cause the serviced
   user to be disconnected. Otherwise, records produced by the client

may be lost by the server which no longer accepts them after the
connectivity is re-established. This state machine is the third
state machine in this section. The state machine is supervised by a
supervision session timer Ts, which the value should be reasonably
higher than the Acct_Interim_Interval value. Ts MAY be set to two
times the value of the Acct_Interim_Interval so as to avoid the
accounting session in the Diameter server to change to Idle state in
case of short transient network failure.

Any event not listed in the state machines MUST be considered as an
error condition, and a corresponding answer, if applicable, MUST be
returned to the originator of the message.

In the state table, the event 'Failure to send' means that the
Diameter client is unable to communicate with the desired
destination. This could be due to the peer being down, or due to the
peer sending back a transient failure or temporary protocol error
notification DIAMETER_OUT_OF_SPACE, DIAMETER_TOO_BUSY, or
DIAMETER_LOOP_DETECTED in the Result-Code AVP of the Accounting
Answer command.

The event 'Failed answer' means that the Diameter client received a
non-transient failure notification in the Accounting Answer command.

Note that the action 'Disconnect user/dev' MUST have an effect also
to the authorization session state table, e.g., cause the STR message
to be sent, if the given application has both
authentication/authorization and accounting portions.

The states PendingS, PendingI, PendingL, PendingE and PendingB stand
for pending states to wait for an answer to an accounting request
related to a Start, Interim, Stop, Event or buffered record,
respectively.

```
                    CLIENT, ACCOUNTING
   State       Event                           Action      New State
   ------------------------------------------------------------------
   Idle        Client or device requests       Send        PendingS
               access                          accounting
                                               start req.

   Idle        Client or device requests       Send        PendingE
               a one-time service              accounting
                                               event req

   Idle        Records in storage              Send        PendingB
                                               record
```

PendingS	Successful accounting start answer received		Open
PendingS	Failure to send and buffer space available and realtime not equal to DELIVER_AND_GRANT	Store Start Record	Open
PendingS	Failure to send and no buffer space available and realtime equal to GRANT_AND_LOSE		Open
PendingS	Failure to send and no buffer space available and realtime not equal to GRANT_AND_LOSE	Disconnect user/dev	Idle
PendingS	Failed accounting start answer received and realtime equal to GRANT_AND_LOSE		Open
PendingS	Failed accounting start answer received and realtime not equal to GRANT_AND_LOSE	Disconnect user/dev	Idle
PendingS	User service terminated	Store stop record	PendingS
Open	Interim interval elapses	Send accounting interim record	PendingI
Open	User service terminated	Send accounting stop req.	PendingL
PendingI	Successful accounting interim answer received		Open
PendingI	Failure to send and (buffer space available or old record can be overwritten) and realtime not equal to DELIVER_AND_GRANT	Store interim record	Open
PendingI	Failure to send and no buffer space available and realtime equal to GRANT_AND_LOSE		Open

```
   PendingI  Failure to send and no buffer   Disconnect Idle
             space available and realtime    user/dev
             not equal to GRANT_AND_LOSE

   PendingI  Failed accounting interim                  Open
             answer received and realtime
             equal to GRANT_AND_LOSE

   PendingI  Failed accounting interim       Disconnect Idle
             answer received and realtime    user/dev
             not equal to GRANT_AND_LOSE

   PendingI  User service terminated         Store      PendingI
                                             stop
                                             record
   PendingE  Successful accounting                      Idle
             event answer received

   PendingE  Failure to send and buffer      Store      Idle
             space available                 event
                                             record

   PendingE  Failure to send and no buffer              Idle
             space available

   PendingE  Failed accounting event answer             Idle
             received

   PendingB  Successful accounting answer    Delete     Idle
             received                        record

   PendingB  Failure to send                            Idle

   PendingB  Failed accounting answer        Delete     Idle
             received                        record

   PendingL  Successful accounting                      Idle
             stop answer received

   PendingL  Failure to send and buffer      Store      Idle
             space available                 stop
                                             record

   PendingL  Failure to send and no buffer              Idle
             space available

   PendingL  Failed accounting stop answer              Idle
             received
```

                     SERVER, STATELESS ACCOUNTING
     State   Event   Action   New State

     Idle   Accounting start request   Send   Idle
                 received, and successfully   accounting
                 processed.   start
                                                answer

     Idle   Accounting event request   Send   Idle
                 received, and successfully   accounting
                 processed.   event
                                                answer

     Idle   Interim record received,   Send   Idle
                 and successfully processed.   accounting
                                                interim
                                                answer

     Idle   Accounting stop request   Send   Idle
                 received, and successfully   accounting
                 processed   stop answer

     Idle   Accounting request received,   Send   Idle
                 no space left to store   accounting
                 records   answer,
                                                Result-Code
                                                = OUT_OF_
                                                SPACE

                     SERVER, STATEFUL ACCOUNTING
     State   Event   Action   New State

     Idle   Accounting start request   Send   Open
                 received, and successfully   accounting
                 processed.   start
                                                answer,
                                                Start Ts

     Idle   Accounting event request   Send   Idle
                 received, and successfully   accounting
                 processed.   event
                                                answer

Idle	Accounting request received, no space left to store records	Send accounting answer, Result-Code = OUT_OF_ SPACE	Idle
Open	Interim record received, and successfully processed.	Send accounting interim answer, Restart Ts	Open
Open	Accounting stop request received, and successfully processed	Send accounting stop answer, Stop Ts	Idle
Open	Accounting request received, no space left to store records	Send accounting answer, Result-Code = OUT_OF_ SPACE, Stop Ts	Idle
Open	Session supervision timer Ts expired	Stop Ts	Idle

8.3. Server-Initiated Re-Auth

   A Diameter server may initiate a re-authentication and/or re-
   authorization service for a particular session by issuing a Re-Auth-
   Request (RAR).

   For example, for pre-paid services, the Diameter server that
   originally authorized a session may need some confirmation that the
   user is still using the services.

   An access device that receives a RAR message with Session-Id equal to
   a currently active session MUST initiate a re-auth towards the user,
   if the service supports this particular feature. Each Diameter
   application MUST state whether service-initiated re-auth is
   supported, since some applications do not allow access devices to
   prompt the user for re-auth.

8.3.1. Re-Auth-Request

   The Re-Auth-Request (RAR), indicated by the Command-Code set to 258
   and the message flags' 'R' bit set, may be sent by any server to the
   access device that is providing session service, to request that the
   user be re-authenticated and/or re-authorized.

   Message Format

      <RAR> ::= < Diameter Header: 258, REQ, PXY >
                 < Session-Id >
                 { Origin-Host }
                 { Origin-Realm }
                 { Destination-Realm }
                 { Destination-Host }
                 { Auth-Application-Id }
                 { Re-Auth-Request-Type }
                 [User-Name]
                 [Origin-State-Id]
               * [Proxy-Info]
               * [Route-Record]
               * [AVP]

8.3.2. Re-Auth-Answer

   The Re-Auth-Answer (RAA), indicated by the Command-Code set to 258
   and the message flags' 'R' bit clear, is sent in response to the RAR.
   The Result-Code AVP MUST be present, and indicates the disposition of
   the request.

   A successful RAA message MUST be followed by an application-specific
   authentication and/or authorization message.

Message Format

```
     <RAA>  ::= < Diameter Header: 258, PXY >
                < Session-Id >
                { Result-Code }
                { Origin-Host }
                { Origin-Realm }
                [ User-Name ]
                [ Origin-State-Id ]
                [ Error-Message ]
                [ Error-Reporting-Host ]
              * [ Failed-AVP ]
              * [ Redirect-Host ]
                [ Redirect-Host-Usage ]
                [ Redirect-Host-Cache-Time ]
              * [ Proxy-Info ]
              * [ AVP ]
```

8.4. Session Termination

   It is necessary for a Diameter server that authorized a session, for
   which it is maintaining state, to be notified when that session is no
   longer active, both for tracking purposes as well as to allow
   stateful agents to release any resources that they may have provided
   for the user's session. For sessions whose state is not being
   maintained, this section is not used.

   When a user session that required Diameter authorization terminates,
   the access device that provided the service MUST issue a Session-
   Termination-Request (STR) message to the Diameter server that
   authorized the service, to notify it that the session is no longer
   active. An STR MUST be issued when a user session terminates for any
   reason, including user logoff, expiration of Session-Timeout,
   administrative action, termination upon receipt of an Abort-Session-
   Request (see below), orderly shutdown of the access device, etc.

   The access device also MUST issue an STR for a session that was
   authorized but never actually started. This could occur, for
   example, due to a sudden resource shortage in the access device, or
   because the access device is unwilling to provide the type of service
   requested in the authorization, or because the access device does not
   support a mandatory AVP returned in the authorization, etc.

   It is also possible that a session that was authorized is never
   actually started due to action of a proxy. For example, a proxy may
   modify an authorization answer, converting the result from success to
   failure, prior to forwarding the message to the access device. If
   the answer did not contain an Auth-Session-State AVP with the value

NO_STATE_MAINTAINED, a proxy that causes an authorized session not to
be started MUST issue an STR to the Diameter server that authorized
the session, since the access device has no way of knowing that the
session had been authorized.

A Diameter server that receives an STR message MUST clean up
resources (e.g., session state) associated with the Session-Id
specified in the STR, and return a Session-Termination-Answer.

A Diameter server also MUST clean up resources when the Session-
Timeout expires, or when the Authorization-Lifetime and the Auth-
Grace-Period AVPs expires without receipt of a re-authorization
request, regardless of whether an STR for that session is received.
The access device is not expected to provide service beyond the
expiration of these timers; thus, expiration of either of these
timers implies that the access device may have unexpectedly shut
down.

8.4.1. Session-Termination-Request

The Session-Termination-Request (STR), indicated by the Command-Code
set to 275 and the Command Flags' 'R' bit set, is sent by the access
device to inform the Diameter Server that an authenticated and/or
authorized session is being terminated.

Message Format

```
   <STR> ::= < Diameter Header: 275, REQ, PXY >
             < Session-Id >
             { Origin-Host }
             { Origin-Realm }
             { Destination-Realm }
             { Auth-Application-Id }
             { Termination-Cause }
             [ User-Name ]
             [ Destination-Host ]
          * [ Class ]
             [ Origin-State-Id ]
          * [ Proxy-Info ]
          * [ Route-Record ]
          * [ AVP ]
```

8.4.2. Session-Termination-Answer

   The Session-Termination-Answer (STA), indicated by the Command-Code
   set to 275 and the message flags' 'R' bit clear, is sent by the
   Diameter Server to acknowledge the notification that the session has
   been terminated. The Result-Code AVP MUST be present, and MAY
   contain an indication that an error occurred while servicing the STR.

   Upon sending or receipt of the STA, the Diameter Server MUST release
   all resources for the session indicated by the Session-Id AVP. Any
   intermediate server in the Proxy-Chain MAY also release any
   resources, if necessary.

   Message Format

       <STA> ::= < Diameter Header: 275, PXY >
                  < Session-Id >
                  { Result-Code }
                  { Origin-Host }
                  { Origin-Realm }
                  [User-Name]
                * [Class]
                  [Error-Message]
                  [Error-Reporting-Host]
                * [Failed-AVP]
                  [Origin-State-Id]
                * [Redirect-Host]
                  [Redirect-Host-Usage]
                                    ^
                  [Redirect-Max-Cache-Time]
                * [Proxy-Info]
                * [AVP]

8.5. Aborting a Session

   A Diameter server may request that the access device stop providing
   service for a particular session by issuing an Abort-Session-Request
   (ASR).

   For example, the Diameter server that originally authorized the
   session may be required to cause that session to be stopped for
   credit or other reasons that were not anticipated when the session
   was first authorized. On the other hand, an operator may maintain a
   management server for the purpose of issuing ASRs to administratively
   remove users from the network.

   An access device that receives an ASR with Session-ID equal to a
   currently active session MAY stop the session. Whether the access

device stops the session or not is implementation- and/or
configuration-dependent. For example, an access device may honor
ASRs from certain agents only. In any case, the access device MUST
respond with an Abort-Session-Answer, including a Result-Code AVP to
indicate what action it took.

Note that if the access device does stop the session upon receipt of
an ASR, it issues an STR to the authorizing server (which may or may
not be the agent issuing the ASR) just as it would if the session
were terminated for any other reason.

8.5.1. Abort-Session-Request

The Abort-Session-Request (ASR), indicated by the Command-Code set to
274 and the message flags' 'R' bit set, may be sent by any server to
the access device that is providing session service, to request that
the session identified by the Session-Id be stopped.

Message Format

```
    <ASR>  ::= < Diameter Header: 274, REQ, PXY >
               < Session-Id >
               { Origin-Host }
               { Origin-Realm }
               { Destination-Realm }
               { Destination-Host }
               { Auth-Application-Id }
               [ User-Name ]
               [ Origin-State-Id ]
             * [ Proxy-Info ]
             * [ Route-Record ]
             * [ AVP ]
```

8.5.2. Abort-Session-Answer

The Abort-Session-Answer (ASA), indicated by the Command-Code set to
274 and the message flags' 'R' bit clear, is sent in response to the
ASR. The Result-Code AVP MUST be present, and indicates the
disposition of the request.

If the session identified by Session-Id in the ASR was successfully
terminated, Result-Code is set to DIAMETER_SUCCESS. If the session
is not currently active, Result-Code is set to
DIAMETER_UNKNOWN_SESSION_ID. If the access device does not stop the
session for any other reason, Result-Code is set to
DIAMETER_UNABLE_TO_COMPLY.

Message Format

```
    <ASA>  ::= < Diameter Header: 274, PXY >
               < Session-Id >
             { Result-Code }
             { Origin-Host }
             { Origin-Realm }
             [ User-Name ]
             [ Origin-State-Id ]
             [ Error-Message ]
             [ Error-Reporting-Host ]
          * [ Failed-AVP ]
          * [ Redirect-Host ]
             [ Redirect-Host-Usage ]
             [ Redirect-Max-Cache-Time ]
          * [ Proxy-Info ]
          * [ AVP ]
```

8.6.   Inferring Session Termination from Origin-State-Id

   Origin-State-Id is used to allow rapid detection of terminated
   sessions for which no STR would have been issued, due to
   unanticipated shutdown of an access device.

   By including Origin-State-Id in CER/CEA messages, an access device
   allows a next-hop server to determine immediately upon connection
   whether the device has lost its sessions since the last connection.

   By including Origin-State-Id in request messages, an access device
   also allows a server with which it communicates via proxy to make
   such a determination. However, a server that is not directly
   connected with the access device will not discover that the access
   device has been restarted unless and until it receives a new request
   from the access device. Thus, use of this mechanism across proxies
   is opportunistic rather than reliable, but useful nonetheless.

   When a Diameter server receives an Origin-State-Id that is greater
   than the Origin-State-Id previously received from the same issuer, it
   may assume that the issuer has lost state since the previous message
   and that all sessions that were active under the lower Origin-State-
   Id have been terminated. The Diameter server MAY clean up all
   session state associated with such lost sessions, and MAY also issues
   STRs for all such lost sessions that were authorized on upstream
   servers, to allow session state to be cleaned up globally.

8.7. Auth-Request-Type AVP

   The Auth-Request-Type AVP (AVP Code 274) is of type Enumerated and is
   included in application-specific auth requests to inform the peers
   whether a user is to be authenticated only, authorized only or both.
   Note any value other than both MAY cause RADIUS interoperability
   issues. The following values are defined:

   AUTHENTICATE_ONLY   1
      The request being sent is for authentication only, and MUST
      contain the relevant application specific authentication AVPs that
      are needed by the Diameter server to authenticate the user.

   AUTHORIZE_ONLY   2
      The request being sent is for authorization only, and MUST contain
      the application specific authorization AVPs that are necessary to
      identify the service being requested/offered.

   AUTHORIZE_AUTHENTICATE   3
      The request contains a request for both authentication and
      authorization. The request MUST include both the relevant
      application specific authentication information, and authorization
      information necessary to identify the service being
      requested/offered.

8.8. Session-Id AVP

   The Session-Id AVP (AVP Code 263) is of type UTF8String and is used
   to identify a specific session (see Section 8). All messages
   pertaining to a specific session MUST include only one Session-Id AVP
   and the same value MUST be used throughout the life of a session.
   When present, the Session-Id SHOULD appear immediately following the
   Diameter Header (see Section 3).

   The Session-Id MUST be globally and eternally unique, as it is meant
   to uniquely identify a user session without reference to any other
   information, and may be needed to correlate historical authentication
   information with accounting information. The Session-Id includes a
   mandatory portion and an implementation-defined portion; a
   recommended format for the implementation-defined portion is outlined
   below.

   The Session-Id MUST begin with the sender's identity encoded in the
   DiameterIdentity type (see Section 4.4). The remainder of the
   Session-Id is delimited by a ";" character, and MAY be any sequence
   that the client can guarantee to be eternally unique; however, the
   following format is recommended, (square brackets [] indicate an
   optional element):

    <DiameterIdentity>;<high 32 bits>;<low 32 bits>[;<optional value>]

    <high 32 bits> and <low 32 bits> are decimal representations of the
    high and low 32 bits of a monotonically increasing 64-bit value. The
    64-bit value is rendered in two part to simplify formatting by 32-bit
    processors. At startup, the high 32 bits of the 64-bit value MAY be
    initialized to the time, and the low 32 bits MAY be initialized to
    zero. This will for practical purposes eliminate the possibility of
    overlapping Session-Ids after a reboot, assuming the reboot process
    takes longer than a second. Alternatively, an implementation MAY
    keep track of the increasing value in non-volatile memory.

    <optional value> is implementation specific but may include a modem's
    device Id, a layer 2 address, timestamp, etc.

    Example, in which there is no optional value:
       accesspoint7.acme.com;1876543210;523

    Example, in which there is an optional value:
       accesspoint7.acme.com;1876543210;523;mobile@200.1.1.88

    The Session-Id is created by the Diameter application initiating the
    session, which in most cases is done by the client. Note that a
    Session-Id MAY be used for both the authorization and accounting
    commands of a given application.

8.9. Authorization-Lifetime AVP

    The Authorization-Lifetime AVP (AVP Code 291) is of type Unsigned32
    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, non-zero, value could
    create significant Diameter traffic, which could congest both the
    network and the agents.

    A value of zero (0) means that immediate re-auth is necessary by the
    access device. This is typically used in cases where multiple
    authentication methods are used, and a successful auth response with
    this AVP set to zero is used to signal that the next authentication
    method is to be immediately initiated. The absence of this AVP, or a
    value of all ones (meaning all bits in the 32 bit field are set to
    one) means no re-auth is expected.

    If both this AVP and the Session-Timeout AVP are present in a
    message, the value of the latter MUST NOT be smaller than the
    Authorization-Lifetime AVP.

An Authorization-Lifetime AVP MAY be present in re-authorization
messages, and contains the number of seconds the user is authorized
to receive service from the time the re-auth answer message is
received by the access device.

This AVP MAY be provided by the client as a hint of the maximum
lifetime that it is willing to accept. However, the server MAY
return a value that is equal to, or smaller, than the one provided by
the client.

8.10. Auth-Grace-Period AVP

The Auth-Grace-Period AVP (AVP Code 276) is of type Unsigned32 and
contains the number of seconds the Diameter server will wait
following the expiration of the Authorization-Lifetime AVP before
cleaning up resources for the session.

8.11. Auth-Session-State AVP

The Auth-Session-State AVP (AVP Code 277) is of type Enumerated and
specifies whether state is maintained for a particular session. The
client MAY include this AVP in requests as a hint to the server, but
the value in the server's answer message is binding. The following
values are supported:

   STATE_MAINTAINED   0
      This value is used to specify that session state is being
      maintained, and the access device MUST issue a session termination
      message when service to the user is terminated. This is the
      default value.

   NO_STATE_MAINTAINED   1
      This value is used to specify that no session termination messages
      will be sent by the access device upon expiration of the
      Authorization-Lifetime.

8.12. Re-Auth-Request-Type AVP

The Re-Auth-Request-Type AVP (AVP Code 285) is of type Enumerated and
is included in application-specific auth answers to inform the client
of the action expected upon expiration of the Authorization-Lifetime.
If the answer message contains an Authorization-Lifetime AVP with a
positive value, the Re-Auth-Request-Type AVP MUST be present in an
answer message. The following values are defined:

   AUTHORIZE_ONLY   0
      An authorization only re-auth is expected upon expiration of the
      Authorization-Lifetime. This is the default value if the AVP is
      not present in answer messages that include the Authorization-
      Lifetime.

   AUTHORIZE_AUTHENTICATE   1
      An authentication and authorization re-auth is expected upon
      expiration of the Authorization-Lifetime.

8.13. Session-Timeout AVP

   The Session-Timeout AVP (AVP Code 27) [RADIUS] is of type Unsigned32
   and contains the maximum number of seconds of service to be provided
   to the user before termination of the session. When both the
   Session-Timeout and the Authorization-Lifetime AVPs are present in an
   answer message, the former MUST be equal to or greater than the value
   of the latter.

   A session that terminates on an access device due to the expiration
   of the Session-Timeout MUST cause an STR to be issued, unless both
   the access device and the home server had previously agreed that no
   session termination messages would be sent (see Section 8.9).

   A Session-Timeout AVP MAY be present in a re-authorization answer
   message, and contains the remaining number of seconds from the
   beginning of the re-auth.

   A value of zero, or the absence of this AVP, means that this session
   has an unlimited number of seconds before termination.

   This AVP MAY be provided by the client as a hint of the maximum
   timeout that it is willing to accept. However, the server MAY return
   a value that is equal to, or smaller, than the one provided by the
   client.

8.14. User-Name AVP

   The User-Name AVP (AVP Code 1) [RADIUS] is of type UTF8String, which
   contains the User-Name, in a format consistent with the NAI
   specification [NAI].

8.15. Termination-Cause AVP

   The Termination-Cause AVP (AVP Code 295) is of type Enumerated, and
   is used to indicate the reason why a session was terminated on the
   access device. The following values are defined:

DIAMETER_LOGOUT   1
   The user initiated a disconnect

DIAMETER_SERVICE_NOT_PROVIDED   2
   This value is used when the user disconnected prior to the receipt
   of the authorization answer message.

DIAMETER_BAD_ANSWER   3
   This value indicates that the authorization answer received by the
   access device was not processed successfully.

DIAMETER_ADMINISTRATIVE   4
   The user was not granted access, or was disconnected, due to
   administrative reasons, such as the receipt of a Abort-Session-
   Request message.

DIAMETER_LINK_BROKEN   5
   The communication to the user was abruptly disconnected.

DIAMETER_AUTH_EXPIRED   6
   The user's access was terminated since its authorized session time
   has expired.

DIAMETER_USER_MOVED   7
   The user is receiving services from another access device.

DIAMETER_SESSION_TIMEOUT   8
   The user's session has timed out, and service has been terminated.

8.16. Origin-State-Id AVP

The Origin-State-Id AVP (AVP Code 278), of type Unsigned32, is a
monotonically increasing value that is advanced whenever a Diameter
entity restarts with loss of previous state, for example upon reboot.
Origin-State-Id MAY be included in any Diameter message, including
CER.

A Diameter entity issuing this AVP MUST create a higher value for
this AVP each time its state is reset. A Diameter entity MAY set
Origin-State-Id to the time of startup, or it MAY use an incrementing
counter retained in non-volatile memory across restarts.

The Origin-State-Id, if present, MUST reflect the state of the entity
indicated by Origin-Host. If a proxy modifies Origin-Host, it MUST
either remove Origin-State-Id or modify it appropriately as well.

Typically, Origin-State-Id is used by an access device that always
starts up with no active sessions; that is, any session active prior
to restart will have been lost. By including Origin-State-Id in a
message, it allows other Diameter entities to infer that sessions
associated with a lower Origin-State-Id are no longer active. If an
access device does not intend for such inferences to be made, it MUST
either not include Origin-State-Id in any message, or set its value
to 0.

8.17. Session-Binding AVP

The Session-Binding AVP (AVP Code 270) is of type Unsigned32, and MAY
be present in application-specific authorization answer messages. If
present, this AVP MAY inform the Diameter client that all future
application-specific re-auth messages for this session MUST be sent
to the same authorization server. This AVP MAY also specify that a
Session-Termination-Request message for this session MUST be sent to
the same authorizing server.

This field is a bit mask, and the following bits have been defined:

RE_AUTH   1
   When set, future re-auth messages for this session MUST NOT
   include the Destination-Host AVP. When cleared, the default
   value, the Destination-Host AVP MUST be present in all re-auth
   messages for this session.

STR   2
   When set, the STR message for this session MUST NOT include the
   Destination-Host AVP. When cleared, the default value, the
   Destination-Host AVP MUST be present in the STR message for this
   session.

ACCOUNTING   4
   When set, all accounting messages for this session MUST NOT
   include the Destination-Host AVP. When cleared, the default
   value, the Destination-Host AVP, if known, MUST be present in all
   accounting messages for this session.

8.18. Session-Server-Failover AVP

The Session-Server-Failover AVP (AVP Code 271) is of type Enumerated,
and MAY be present in application-specific authorization answer
messages that either do not include the Session-Binding AVP or
include the Session-Binding AVP with any of the bits set to a zero
value. If present, this AVP MAY inform the Diameter client that if a

re-auth or STR message fails due to a delivery problem, the Diameter
client SHOULD issue a subsequent message without the Destination-Host
AVP. When absent, the default value is REFUSE_SERVICE.

The following values are supported:

REFUSE_SERVICE   0
   If either the re-auth or the STR message delivery fails, terminate
   service with the user, and do not attempt any subsequent attempts.

TRY_AGAIN   1
   If either the re-auth or the STR message delivery fails, resend
   the failed message without the Destination-Host AVP present.

ALLOW_SERVICE   2
   If re-auth message delivery fails, assume that re-authorization
   succeeded. If STR message delivery fails, terminate the session.

TRY_AGAIN_ALLOW_SERVICE   3
   If either the re-auth or the STR message delivery fails, resend
   the failed message without the Destination-Host AVP present. If
   the second delivery fails for re-auth, assume re-authorization
   succeeded. If the second delivery fails for STR, terminate the
   session.

8.19. Multi-Round-Time-Out AVP

The Multi-Round-Time-Out AVP (AVP Code 272) is of type Unsigned32,
and SHOULD be present in application-specific authorization answer
messages whose Result-Code AVP is set to DIAMETER_MULTI_ROUND_AUTH.
This AVP contains the maximum number of seconds that the access
device MUST provide the user in responding to an authentication
request.

8.20. Class AVP

The Class AVP (AVP Code 25) is of type OctetString and is used to by
Diameter servers to return state information to the access device.
When one or more Class AVPs are present in application-specific
authorization answer messages, they MUST be present in subsequent
re-authorization, session termination and accounting messages. Class
AVPs found in a re-authorization answer message override the ones
found in any previous authorization answer message. Diameter server
implementations SHOULD NOT return Class AVPs that require more than
4096 bytes of storage on the Diameter client. A Diameter client that
receives Class AVPs whose size exceeds local available storage MUST
terminate the session.

8.21. Event-Timestamp AVP

   The Event-Timestamp (AVP Code 55) is of type Time, and MAY be
   included in an Accounting-Request and Accounting-Answer messages to
   record the time that the reported event occurred, in seconds since
   January 1, 1900 00:00 UTC.

9. Accounting

   This accounting protocol is based on a server directed model with
   capabilities for real-time delivery of accounting information.
   Several fault resilience methods [ACCMGMT] have been built in to the
   protocol in order minimize loss of accounting data in various fault
   situations and under different assumptions about the capabilities of
   the used devices.

9.1. Server Directed Model

   The server directed model means that the device generating the
   accounting data gets information from either the authorization server
   (if contacted) or the accounting server regarding the way accounting
   data shall be forwarded. This information includes accounting record
   timeliness requirements.

   As discussed in [ACCMGMT], real-time transfer of accounting records
   is a requirement, such as the need to perform credit limit checks and
   fraud detection. Note that batch accounting is not a requirement,
   and is therefore not supported by Diameter. Should batched
   accounting be required in the future, a new Diameter application will
   need to be created, or it could be handled using another protocol.
   Note, however, that even if at the Diameter layer accounting requests
   are processed one by one, transport protocols used under Diameter
   typically batch several requests in the same packet under heavy
   traffic conditions. This may be sufficient for many applications.

   The authorization server (chain) directs the selection of proper
   transfer strategy, based on its knowledge of the user and
   relationships of roaming partnerships. The server (or agents) uses
   the Acct-Interim-Interval and Accounting-Realtime-Required AVPs to
   control the operation of the Diameter peer operating as a client.
   The Acct-Interim-Interval AVP, when present, instructs the Diameter
   node acting as a client to produce accounting records continuously
   even during a session. Accounting-Realtime-Required AVP is used to
   control the behavior of the client when the transfer of accounting
   records from the Diameter client is delayed or unsuccessful.

The Diameter accounting server MAY override the interim interval or
the realtime requirements by including the Acct-Interim-Interval or
Accounting-Realtime-Required AVP in the Accounting-Answer message.
When one of these AVPs is present, the latest value received SHOULD
be used in further accounting activities for the same session.

9.2. Protocol Messages

A Diameter node that receives a successful authentication and/or
authorization messages from the Home AAA server MUST collect
accounting information for the session. The Accounting-Request
message is used to transmit the accounting information to the Home
AAA server, which MUST reply with the Accounting-Answer message to
confirm reception. The Accounting-Answer message includes the
Result-Code AVP, which MAY indicate that an error was present in the
accounting message. A rejected Accounting-Request message MAY cause
the user's session to be terminated, depending on the value of the
Accounting-Realtime-Required AVP received earlier for the session in
question.

Each Diameter Accounting protocol message MAY be compressed, in order
to reduce network bandwidth usage. If IPsec and IKE are used to
secure the Diameter session, then IP compression [IPComp] MAY be used
and IKE [IKE] MAY be used to negotiate the compression parameters.
If TLS is used to secure the Diameter session, then TLS compression
[TLS] MAY be used.

9.3. Application document requirements

Each Diameter application (e.g., NASREQ, MobileIP), MUST define their
Service-Specific AVPs that MUST be present in the Accounting-Request
message in a section entitled "Accounting AVPs". The application
MUST assume that the AVPs described in this document will be present
in all Accounting messages, so only their respective service-specific
AVPs need to be defined in this section.

9.4. Fault Resilience

Diameter Base protocol mechanisms are used to overcome small message
loss and network faults of temporary nature.

Diameter peers acting as clients MUST implement the use of failover
to guard against server failures and certain network failures.
Diameter peers acting as agents or related off-line processing
systems MUST detect duplicate accounting records caused by the
sending of same record to several servers and duplication of messages

in transit. This detection MUST be based on the inspection of the
Session-Id and Accounting-Record-Number AVP pairs. Appendix C
discusses duplicate detection needs and implementation issues.

Diameter clients MAY have non-volatile memory for the safe storage of
accounting records over reboots or extended network failures, network
partitions, and server failures. If such memory is available, the
client SHOULD store new accounting records there as soon as the
records are created and until a positive acknowledgement of their
reception from the Diameter Server has been received. Upon a reboot,
the client MUST starting sending the records in the non-volatile
memory to the accounting server with appropriate modifications in
termination cause, session length, and other relevant information in
the records.

A further application of this protocol may include AVPs to control
how many accounting records may at most be stored in the Diameter
client without committing them to the non-volatile memory or
transferring them to the Diameter server.

The client SHOULD NOT remove the accounting data from any of its
memory areas before the correct Accounting-Answer has been received.
The client MAY remove oldest, undelivered or yet unacknowledged
accounting data if it runs out of resources such as memory. It is an
implementation dependent matter for the client to accept new sessions
under this condition.

9.5. Accounting Records

In all accounting records, the Session-Id AVP MUST be present; the
User-Name AVP MUST be present if it is available to the Diameter
client. If strong authentication across agents is required, end-to-
end security may be used for authentication purposes.

Different types of accounting records are sent depending on the
actual type of accounted service and the authorization server's
directions for interim accounting. If the accounted service is a
one-time event, meaning that the start and stop of the event are
simultaneous, then the Accounting-Record-Type AVP MUST be present and
set to the value EVENT_RECORD.

If the accounted service is of a measurable length, then the AVP MUST
use the values START_RECORD, STOP_RECORD, and possibly,
INTERIM_RECORD. If the authorization server has not directed interim
accounting to be enabled for the session, two accounting records MUST
be generated for each service of type session. When the initial

Accounting-Request for a given session is sent, the Accounting-
Record-Type AVP MUST be set to the value START_RECORD. When the last
Accounting-Request is sent, the value MUST be STOP_RECORD.

If the authorization server has directed interim accounting to be
enabled, the Diameter client MUST produce additional records between
the START_RECORD and STOP_RECORD, marked INTERIM_RECORD. The
production of these records is directed by Acct-Interim-Interval as
well as any re-authentication or re-authorization of the session. The
Diameter client MUST overwrite any previous interim accounting
records that are locally stored for delivery, if a new record is
being generated for the same session. This ensures that only one
pending interim record can exist on an access device for any given
session.

A particular value of Accounting-Sub-Session-Id MUST appear only in
one sequence of accounting records from a DIAMETER client, except for
the purposes of retransmission. The one sequence that is sent MUST
be either one record with Accounting-Record-Type AVP set to the value
EVENT_RECORD, or several records starting with one having the value
START_RECORD, followed by zero or more INTERIM_RECORD and a single
STOP_RECORD. A particular Diameter application specification MUST
define the type of sequences that MUST be used.

9.6. Correlation of Accounting Records

The Diameter protocol's Session-Id AVP, which is globally unique (see
Section 8.8), is used during the authorization phase to identify a
particular session. Services that do not require any authorization
still use the Session-Id AVP to identify sessions. Accounting
messages MAY use a different Session-Id from that sent in
authorization messages. Specific applications MAY require different
a Session-ID for accounting messages.

However, there are certain applications that require multiple
accounting sub-sessions. Such applications would send messages with
a constant Session-Id AVP, but a different Accounting-Sub-Session-Id
AVP. In these cases, correlation is performed using the Session-Id.
It is important to note that receiving a STOP_RECORD with no
Accounting-Sub-Session-Id AVP when sub-sessions were originally used
in the START_RECORD messages implies that all sub-sessions are
terminated.

Furthermore, there are certain applications where a user receives
service from different access devices (e.g., Mobile IPv4), each with
their own unique Session-Id. In such cases, the Acct-Multi-Session-
Id AVP is used for correlation. During authorization, a server that

determines that a request is for an existing session SHOULD include
the Acct-Multi-Session-Id AVP, which the access device MUST include
in all subsequent accounting messages.

The Acct-Multi-Session-Id AVP MAY include the value of the original
Session-Id. It's contents are implementation specific, but MUST be
globally unique across other Acct-Multi-Session-Id, and MUST NOT
change during the life of a session.

A Diameter application document MUST define the exact concept of a
session that is being accounted, and MAY define the concept of a
multi-session. For instance, the NASREQ DIAMETER application treats
a single PPP connection to a Network Access Server as one session,
and a set of Multilink PPP sessions as one multi-session.

9.7. Accounting Command-Codes

This section defines Command-Code values that MUST be supported by
all Diameter implementations that provide Accounting services.

9.7.1. Accounting-Request

The Accounting-Request (ACR) command, indicated by the Command-Code
field set to 271 and the Command Flags' 'R' bit set, is sent by a
Diameter node, acting as a client, in order to exchange accounting
information with a peer.

One of Acct-Application-Id and Vendor-Specific-Application-Id AVPs
MUST be present. If the Vendor-Specific-Application-Id grouped AVP
is present, it must have an Acct-Application-Id inside.

The AVP listed below SHOULD include service specific accounting AVPs,
as described in Section 9.3.

   Message Format

       <ACR> ::= < Diameter Header: 271, REQ, PXY >
                 < Session-Id >
                 { Origin-Host }
                 { Origin-Realm }
                 { Destination-Realm }
                 { Accounting-Record-Type }
                 { Accounting-Record-Number }
                 [Acct-Application-Id]
                 [Vendor-Specific-Application-Id]
                 [User-Name]
                 [Accounting-Sub-Session-Id]
                 [Acct-Session-Id]
                 [Acct-Multi-Session-Id]
                 [Acct-Interim-Interval]
                 [Accounting-Realtime-Required]
                 [Origin-State-Id]
                 [Event-Timestamp]
               * [Proxy-Info]
               * [Route-Record]
               * [AVP]

9.7.2. Accounting-Answer

   The Accounting-Answer (ACA) command, indicated by the Command-Code
   field set to 271 and the Command Flags' 'R' bit cleared, is used to
   acknowledge an Accounting-Request command. The Accounting-Answer
   command contains the same Session-Id and includes the usage AVPs only
   if CMS is in use when sending this command. Note that the inclusion
   of the usage AVPs when CMS is not being used leads to unnecessarily
   large answer messages, and can not be used as a server's proof of the
   receipt of these AVPs in an end-to-end fashion. If the Accounting-
   Request was protected by end-to-end security, then the corresponding
   ACA message MUST be protected by end-to-end security.

   Only the target Diameter Server, known as the home Diameter Server,
   SHOULD respond with the Accounting-Answer command.

   One of Acct-Application-Id and Vendor-Specific-Application-Id AVPs
   MUST be present. If the Vendor-Specific-Application-Id grouped AVP
   is present, it must have an Acct-Application-Id inside.

   The AVP listed below SHOULD include service specific accounting AVPs,
   as described in Section 9.3.

Message Format

```
    <ACA> ::= < Diameter Header: 271, PXY >
              < Session-Id >
              { Result-Code }
              { Origin-Host }
              { Origin-Realm }
              { Accounting-Record-Type }
              { Accounting-Record-Number }
              [ Acct-Application-Id ]
              [ Vendor-Specific-Application-Id ]
              [ User-Name ]
              [ Accounting-Sub-Session-Id ]
              [ Acct-Session-Id ]
              [ Acct-Multi-Session-Id ]
              [ Error-Reporting-Host ]
              [ Acct-Interim-Interval ]
              [ Accounting-Realtime-Required ]
              [ Origin-State-Id ]
              [ Event-Timestamp ]
            * [ Proxy-Info ]
            * [ AVP ]
```

9.8. Accounting AVPs

   This section contains AVPs that describe accounting usage information
   related to a specific session.

9.8.1. Accounting-Record-Type AVP

   The Accounting-Record-Type AVP (AVP Code 480) is of type Enumerated
   and contains the type of accounting record being sent. The following
   values are currently defined for the Accounting-Record-Type AVP:

   EVENT_RECORD   1
      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.

   START_RECORD   2
      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.

INTERIM_RECORD   3
   An Interim Accounting Record contains cumulative accounting
   information for an existing accounting session. Interim
   Accounting Records SHOULD be sent every time a re-authentication
   or re-authorization occurs. Further, additional interim record
   triggers MAY be defined by application-specific Diameter
   applications. The selection of whether to use INTERIM_RECORD
   records is done by the Acct-Interim-Interval AVP.

STOP_RECORD   4
   An Accounting Stop Record is sent to terminate an accounting
   session and contains cumulative accounting information relevant to
   the existing session.

9.8.2. Acct-Interim-Interval

   The Acct-Interim-Interval AVP (AVP Code 85) is of type Unsigned32 and
   is sent from the Diameter home authorization server to the Diameter
   client. The client uses information in this AVP to decide how and
   when to produce accounting records. With different values in this
   AVP, service sessions can result in one, two, or two+N accounting
   records, based on the needs of the home-organization. The following
   accounting record production behavior is directed by the inclusion of
   this AVP:

   1. The omission of the Acct-Interim-Interval AVP or its inclusion
      with Value field set to 0 means that EVENT_RECORD, START_RECORD,
      and STOP_RECORD are produced, as appropriate for the service.

   2. The inclusion of the AVP with Value field set to a non-zero value
      means that INTERIM_RECORD records MUST be produced between the
      START_RECORD and STOP_RECORD records. The Value field of this AVP
      is the nominal interval between these records in seconds. The
      Diameter node that originates the accounting information, known as
      the client, MUST produce the first INTERIM_RECORD record roughly
      at the time when this nominal interval has elapsed from the
      START_RECORD, the next one again as the interval has elapsed once
      more, and so on until the session ends and a STOP_RECORD record is
      produced.

      The client MUST ensure that the interim record production times
      are randomized so that large accounting message storms are not
      created either among records or around a common service start
      time.

9.8.3. Accounting-Record-Number AVP

   The Accounting-Record-Number AVP (AVP Code 485) is of type Unsigned32
   and identifies this record within one session. As Session-Id AVPs
   are globally unique, the combination of Session-Id and Accounting-
   Record-Number AVPs is also globally unique, and can be used in
   matching accounting records with confirmations. An easy way to
   produce unique numbers is to set the value to 0 for records of type
   EVENT_RECORD and START_RECORD, and set the value to 1 for the first
   INTERIM_RECORD, 2 for the second, and so on until the value for
   STOP_RECORD is one more than for the last INTERIM_RECORD.

9.8.4. Acct-Session-Id AVP

   The Acct-Session-Id AVP (AVP Code 44) is of type OctetString is only
   used when RADIUS/Diameter translation occurs. This AVP contains the
   contents of the RADIUS Acct-Session-Id attribute.

9.8.5. Acct-Multi-Session-Id AVP

   The Acct-Multi-Session-Id AVP (AVP Code 50) is of type UTF8String,
   following the format specified in Section 8.8. The Acct-Multi-
   Session-Id AVP is used to link together multiple related accounting
   sessions, where each session would have a unique Session-Id, but the
   same Acct-Multi-Session-Id AVP. This AVP MAY be returned by the
   Diameter server in an authorization answer, and MUST be used in all
   accounting messages for the given session.

9.8.6. Accounting-Sub-Session-Id AVP

   The Accounting-Sub-Session-Id AVP (AVP Code 287) is of type
   Unsigned64 and contains the accounting sub-session identifier. The
   combination of the Session-Id and this AVP MUST be unique per sub-
   session, and the value of this AVP MUST be monotonically increased by
   one for all new sub-sessions. The absence of this AVP implies no
   sub-sessions are in use, with the exception of an Accounting-Request
   whose Accounting-Record-Type is set to STOP_RECORD. A STOP_RECORD
   message with no Accounting-Sub-Session-Id AVP present will signal the
   termination of all sub-sessions for a given Session-Id.

9.8.7. Accounting-Realtime-Required AVP

   The Accounting-Realtime-Required AVP (AVP Code 483) is of type
   Enumerated and is sent from the Diameter home authorization server to
   the Diameter client or in the Accounting-Answer from the accounting
   server. The client uses information in this AVP to decide what to do
   if the sending of accounting records to the accounting server has
   been temporarily prevented due to, for instance, a network problem.

DELIVER_AND_GRANT   1
   The AVP with Value field set to DELIVER_AND_GRANT means that the
   service MUST only be granted as long as there is a connection to
   an accounting server. Note that the set of alternative accounting
   servers are treated as one server in this sense. Having to move
   the accounting record stream to a backup server is not a reason to
   discontinue the service to the user.

GRANT_AND_STORE   2
   The AVP with Value field set to GRANT_AND_STORE means that service
   SHOULD be granted if there is a connection, or as long as records
   can still be stored as described in Section 9.4.

   This is the default behavior if the AVP isn't included in the
   reply from the authorization server.

GRANT_AND_LOSE   3
   The AVP with Value field set to GRANT_AND_LOSE means that service
   SHOULD be granted even if the records can not be delivered or
   stored.

10. AVP Occurrence Table

The following tables presents the AVPs defined in this document, and
specifies in which Diameter messages they MAY, or MAY NOT be present.
Note that AVPs that can only be present within a Grouped AVP are not
represented in this table.

The table uses the following symbols:

0   The AVP MUST NOT be present in the message.
0+   Zero or more instances of the AVP MAY be present in the
       message.
0-1   Zero or one instance of the AVP MAY be present in the
       message. It is considered an error if there are more than
       one instance of the AVP.
1   One instance of the AVP MUST be present in the message.
1+   At least one instance of the AVP MUST be present in the
       message.

10.1. Base Protocol Command AVP Table

The table in this section is limited to the non-accounting Command
Codes defined in this specification.

Attribute Name	CER	CEA	DPR	DPA	DWR	DWA	RAR	RAA	ASR	ASA	STR	STA
Acct-Interim- Interval	0	0	0	0	0	0	0-1	0	0	0	0	0
Accounting-Realtime- Required	0	0	0	0	0	0	0-1	0	0	0	0	0
Acct-Application-Id	0+	0+	0	0	0	0	0	0	0	0	0	0
Auth-Application-Id	0+	0+	0	0	0	0	1	0	1	0	1	0
Auth-Grace-Period	0	0	0	0	0	0	0	0	0	0	0	0
Auth-Request-Type	0	0	0	0	0	0	0	0	0	0	0	0
Auth-Session-State	0	0	0	0	0	0	0	0	0	0	0	0
Authorization- Lifetime	0	0	0	0	0	0	0	0	0	0	0	0
Class	0	0	0	0	0	0	0	0	0	0	0+	0+
Destination-Host	0	0	0	0	0	0	1	0	1	0	0-1	0
Destination-Realm	0	0	0	0	0	0	1	0	1	0	1	0
Disconnect-Cause	0	0	1	0	0	0	0	0	0	0	0	0
Error-Message	0	0-1	0	0-1	0	0-1	0	0-1	0	0-1	0	0-1
Error-Reporting-Host	0	0	0	0	0	0	0	0-1	0	0-1	0	0-1
Failed-AVP	0	0+	0	0+	0	0+	0	0+	0	0+	0	0+
Firmware-Revision	0-1	0-1	0	0	0	0	0	0	0	0	0	0
Host-IP-Address	1+	1+	0	0	0	0	0	0	0	0	0	0
Inband-Security-Id	0+	0+	0	0	0	0	0	0	0	0	0	0
Multi-Round-Time-Out	0	0	0	0	0	0	0	0	0	0	0	0
Origin-Host	1	1	1	1	1	1	1	1	1	1	1	1
Origin-Realm	1	1	1	1	1	1	1	1	1	1	1	1
Origin-State-Id	0-1	0-1	0	0	0-1	0-1	0-1	0-1	0-1	0-1	0-1	0-1
Product-Name	1	1	0	0	0	0	0	0	0	0	0	0
Proxy-Info	0	0	0	0	0	0	0+	0+	0+	0+	0+	0+
Redirect-Host	0	0	0	0	0	0	0	0+	0	0+	0	0+
Redirect-Host-Usage	0	0	0	0	0	0	0	0-1	0	0-1	0	0-1
Redirect-Max-Cache- Time	0	0	0	0	0	0	0	0-1	0	0-1	0	0-1
Result-Code	0	1	0	1	0	1	0	1	0	0	0	1
Re-Auth-Request-Type	0	0	0	0	0	0	1	0	0	0	0	0
Route-Record	0	0	0	0	0	0	0+	0	0+	0	0+	0
Session-Binding	0	0	0	0	0	0	0	0	0	0	0	0
Session-Id	0	0	0	0	0	0	1	1	1	1	1	1
Session-Server- Failover	0	0	0	0	0	0	0	0	0	0	0	0
Session-Timeout	0	0	0	0	0	0	0	0	0	0	0	0
Supported-Vendor-Id	0+	0+	0	0	0	0	0	0	0	0	0	0
Termination-Cause	0	0	0	0	0	0	0	0	0	0	1	0
User-Name	0	0	0	0	0	0	0-1	0-1	0-1	0-1	0-1	0-1
Vendor-Id	1	1	0	0	0	0	0	0	0	0	0	0

```
Vendor-Specific-         |0+ |0+ |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |
   Application-Id        |   |   |   |   |   |   |   |   |   |   |   |   |
--------------------+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

10.2. Accounting AVP Table

   The table in this section is used to represent which AVPs defined in
   this document are to be present in the Accounting messages. These
   AVP occurrence requirements are guidelines, which may be expanded,
   and/or overridden by application-specific requirements in the
   Diameter applications documents.

```
                                  +-----------+
                                  |  Command  |
                                  |   Code    |
                                  +-----+-----+
   Attribute Name                 | ACR | ACA |
   ---------------------------+-----+-----+
   Acct-Interim-Interval          | 0-1 | 0-1 |
   Acct-Multi-Session-Id          | 0-1 | 0-1 |
   Accounting-Record-Number       | 1   | 1   |
   Accounting-Record-Type         | 1   | 1   |
   Acct-Session-Id                | 0-1 | 0-1 |
   Accounting-Sub-Session-Id      | 0-1 | 0-1 |
   Accounting-Realtime-Required   | 0-1 | 0-1 |
   Acct-Application-Id            | 0-1 | 0-1 |
   Auth-Application-Id            | 0   | 0   |
   Class                          | 0+  | 0+  |
   Destination-Host               | 0-1 | 0   |
   Destination-Realm              | 1   | 0   |
   Error-Reporting-Host           | 0   | 0+  |
   Event-Timestamp                | 0-1 | 0-1 |
   Origin-Host                    | 1   | 1   |
   Origin-Realm                   | 1   | 1   |
   Proxy-Info                     | 0+  | 0+  |
   Route-Record                   | 0+  | 0+  |
   Result-Code                    | 0   | 1   |
   Session-Id                     | 1   | 1   |
   Termination-Cause              | 0-1 | 0-1 |
   User-Name                      | 0-1 | 0-1 |
   Vendor-Specific-Application-Id | 0-1 | 0-1 |
   ---------------------------+-----+-----+
```

11. IANA Considerations

   This section provides guidance to the Internet Assigned Numbers
   Authority (IANA) regarding registration of values related to the
   Diameter protocol, in accordance with BCP 26 [IANA]. The following
   policies are used here with the meanings defined in BCP 26: "Private
   Use", "First Come First Served", "Expert Review", "Specification
   Required", "IETF Consensus", "Standards Action".

   This section explains the criteria to be used by the IANA for
   assignment of numbers within namespaces defined within this document.

   Diameter is not intended as a general purpose protocol, and
   allocations SHOULD NOT be made for purposes unrelated to
   authentication, authorization or accounting.

   For registration requests where a Designated Expert should be
   consulted, the responsible IESG area director should appoint the
   Designated Expert. For Designated Expert with Specification
   Required, the request is posted to the AAA WG mailing list (or, if it
   has been disbanded, a successor designated by the Area Director) for
   comment and review, and MUST include a pointer to a public
   specification. Before a period of 30 days has passed, the Designated
   Expert will either approve or deny the registration request and
   publish a notice of the decision to the AAA WG mailing list or its
   successor. A denial notice must be justified by an explanation and,
   in the cases where it is possible, concrete suggestions on how the
   request can be modified so as to become acceptable.

11.1. AVP Header

   As defined in Section 4, the AVP header contains three fields that
   requires IANA namespace management; the AVP Code, Vendor-ID and Flags
   field.

11.1.1. AVP Codes

   The AVP Code namespace is used to identify attributes. There are
   multiple namespaces. Vendors can have their own AVP Codes namespace
   which will be identified by their Vendor-ID (also known as
   Enterprise-Number) and they control the assignments of their vendor-
   specific AVP codes within their own namespace. The absence of a
   Vendor-ID or a Vendor-ID value of zero (0) identifies the IETF IANA
   controlled AVP Codes namespace. The AVP Codes and sometimes also
   possible values in an AVP are controlled and maintained by IANA.

AVP Code 0 is not used. AVP Codes 1-255 are managed separately as
RADIUS Attribute Types [RADTYPE]. This document defines the AVP
Codes 257-274, 276-285, 287, 291-300, 480, 483 and 485-486. See
Section 4.5 for the assignment of the namespace in this
specification.

AVPs may be allocated following Designated Expert with Specification
Required [IANA]. Release of blocks of AVPs (more than 3 at a time
for a given purpose) should require IETF Consensus.

Note that Diameter defines a mechanism for Vendor-Specific AVPs,
where the Vendor-Id field in the AVP header is set to a non-zero
value. Vendor-Specific AVPs codes are for Private Use and should be
encouraged instead of allocation of global attribute types, for
functions specific only to one vendor's implementation of Diameter,
where no interoperability is deemed useful. Where a Vendor-Specific
AVP is implemented by more than one vendor, allocation of global AVPs
should be encouraged instead.

11.1.2. AVP Flags

There are 8 bits in the AVP Flags field of the AVP header, defined in
Section 4. This document assigns bit 0 ('V'endor Specific), bit 1
('M'andatory) and bit 2 ('P'rotected). The remaining bits should
only be assigned via a Standards Action [IANA].

11.2. Diameter Header

As defined in Section 3, the Diameter header contains two fields that
require IANA namespace management; Command Code and Command Flags.

11.2.1. Command Codes

The Command Code namespace is used to identify Diameter commands.
The values 0-255 are reserved for RADIUS backward compatibility, and
are defined as "RADIUS Packet Type Codes" in [RADTYPE]. Values 256-
16,777,213 are for permanent, standard commands, allocated by IETF
Consensus [IANA]. This document defines the Command Codes 257, 258,
271, 274-275, 280 and 282. See Section 3.1 for the assignment of the
namespace in this specification.

The values 16,777,214 and 16,777,215 (hexadecimal values 0xfffffe -
0xffffff) are reserved for experimental commands. As these codes are
only for experimental and testing purposes, no guarantee is made for
interoperability between Diameter peers using experimental commands,
as outlined in [IANA-EXP].

11.2.2. Command Flags

   There are eight bits in the Command Flags field of the Diameter
   header. This document assigns bit 0 ('R'equest), bit 1 ('P'roxy),
   bit 2 ('E'rror) and bit 3 ('T'). Bits 4 through 7 MUST only be
   assigned via a Standards Action [IANA].

11.3. Application Identifiers

   As defined in Section 2.4, the Application Identifier is used to
   identify a specific Diameter Application. There are standards-track
   application ids and vendor specific application ids.

   IANA [IANA] has assigned the range 0x00000001 to 0x00ffffff for
   standards-track applications; and 0x01000000 - 0xfffffffe for vendor
   specific applications, on a first-come, first-served basis. The
   following values are allocated.

         Diameter Common Messages   0
         NASREQ   1 [NASREQ]
         Mobile-IP   2 [DIAMMIP]
         Diameter Base Accounting   3
         Relay   0xffffffff

   Assignment of standards-track application IDs are by Designated
   Expert with Specification Required [IANA].

   Both Application-Id and Acct-Application-Id AVPs use the same
   Application Identifier space.

   Vendor-Specific Application Identifiers, are for Private Use.
   Vendor-Specific Application Identifiers are assigned on a First Come,
   First Served basis by IANA.

11.4. AVP Values

   Certain AVPs in Diameter define a list of values with various
   meanings. For attributes other than those specified in this section,
   adding additional values to the list can be done on a First Come,
   First Served basis by IANA.

11.4.1. Result-Code AVP Values

   As defined in Section 7.1, the Result-Code AVP (AVP Code 268) defines
   the values 1001, 2001-2002, 3001-3010, 4001-4002 and 5001-5017.

   All remaining values are available for assignment via IETF Consensus
   [IANA].

11.4.2. Accounting-Record-Type AVP Values

   As defined in Section 9.8.1, the Accounting-Record-Type AVP (AVP Code
   480) defines the values 1-4. All remaining values are available for
   assignment via IETF Consensus [IANA].

11.4.3. Termination-Cause AVP Values

   As defined in Section 8.15, the Termination-Cause AVP (AVP Code 295)
   defines the values 1-8. All remaining values are available for
   assignment via IETF Consensus [IANA].

11.4.4. Redirect-Host-Usage AVP Values

   As defined in Section 6.13, the Redirect-Host-Usage AVP (AVP Code
   261) defines the values 0-5. All remaining values are available for
   assignment via IETF Consensus [IANA].

11.4.5. Session-Server-Failover AVP Values

   As defined in Section 8.18, the Session-Server-Failover AVP (AVP Code
   271) defines the values 0-3. All remaining values are available for
   assignment via IETF Consensus [IANA].

11.4.6. Session-Binding AVP Values

   As defined in Section 8.17, the Session-Binding AVP (AVP Code 270)
   defines the bits 1-4. All remaining bits are available for
   assignment via IETF Consensus [IANA].

11.4.7. Disconnect-Cause AVP Values

   As defined in Section 5.4.3, the Disconnect-Cause AVP (AVP Code 273)
   defines the values 0-2. All remaining values are available for
   assignment via IETF Consensus [IANA].

11.4.8. Auth-Request-Type AVP Values

   As defined in Section 8.7, the Auth-Request-Type AVP (AVP Code 274)
   defines the values 1-3. All remaining values are available for
   assignment via IETF Consensus [IANA].

11.4.9. Auth-Session-State AVP Values

   As defined in Section 8.11, the Auth-Session-State AVP (AVP Code 277)
   defines the values 0-1. All remaining values are available for
   assignment via IETF Consensus [IANA].

11.4.10. Re-Auth-Request-Type AVP Values

   As defined in Section 8.12, the Re-Auth-Request-Type AVP (AVP Code
   285) defines the values 0-1. All remaining values are available for
   assignment via IETF Consensus [IANA].

11.4.11. Accounting-Realtime-Required AVP Values

   As defined in Section 9.8.7, the Accounting-Realtime-Required AVP
   (AVP Code 483) defines the values 1-3. All remaining values are
   available for assignment via IETF Consensus [IANA].

11.4.12. Inband-Security-Id AVP (code 299)

   As defined in Section 6.10, the Inband-Security-Id AVP (AVP Code 299)
   defines the values 0-1. All remaining values are available for
   assignment via IETF Consensus [IANA].

11.5. Diameter TCP/SCTP Port Numbers

   The IANA has assigned TCP and SCTP port number 3868 to Diameter.

11.6. NAPTR Service Fields

   The registration in the RFC MUST include the following information:

   Service Field: The service field being registered. An example for a
   new fictitious transport protocol called NCTP might be "AAA+D2N".

   Protocol: The specific transport protocol associated with that
   service field. This MUST include the name and acronym for the
   protocol, along with reference to a document that describes the
   transport protocol. For example - "New Connectionless Transport
   Protocol (NCTP), RFC 5766".

   Name and Contact Information: The name, address, email address and
   telephone number for the person performing the registration.

   The following values have been placed into the registry:

        Services Field   Protocol
        AAA+D2T   TCP
        AAA+D2S   SCTP

12. Diameter protocol related configurable parameters

   This section contains the configurable parameters that are found
   throughout this document:

Diameter Peer
   A Diameter entity MAY communicate with peers that are statically
   configured. A statically configured Diameter peer would require
   that either the IP address or the fully qualified domain name
   (FQDN) be supplied, which would then be used to resolve through
   DNS.

Realm Routing Table
   A Diameter proxy server routes messages based on the realm portion
   of a Network Access Identifier (NAI). The server MUST have a
   table of Realm Names, and the address of the peer to which the
   message must be forwarded to. The routing table MAY also include
   a "default route", which is typically used for all messages that
   cannot be locally processed.

Tc timer
   The Tc timer controls the frequency that transport connection
   attempts are done to a peer with whom no active transport
   connection exists. The recommended value is 30 seconds.

13. Security Considerations

   The Diameter base protocol assumes that messages are secured by using
   either IPSec or TLS. This security mechanism is acceptable in
   environments where there is no untrusted third party agent. In other
   situations, end-to-end security is needed.

   Diameter clients, such as Network Access Servers (NASes) and Mobility
   Agents MUST support IP Security [SECARCH] and MAY support TLS [TLS].
   Diameter servers MUST support TLS and IPsec. Diameter
   implementations MUST use transmission-level security of some kind
   (IPsec or TLS) on each connection.

   If a Diameter connection is not protected by IPsec, then the CER/CEA
   exchange MUST include an Inband-Security-ID AVP with a value of TLS.
   For TLS usage, a TLS handshake will begin when both ends are in the
   open state, after completion of the CER/CEA exchange. If the TLS
   handshake is successful, all further messages will be sent via TLS.
   If the handshake fails, both ends move to the closed state.

   It is suggested that IPsec be used primarily at the edges for intra-
   domain exchanges. For NAS devices without certificate support, pre-
   shared keys can be used between the NAS and a local AAA proxy.

   For protection of inter-domain exchanges, TLS is recommended. See
   Sections 13.1 and 13.2 for more details on IPsec and TLS usage.

13.1. IPsec Usage

   All Diameter implementations MUST support IPsec ESP [IPsec] in
   transport mode with non-null encryption and authentication algorithms
   to provide per-packet authentication, integrity protection and
   confidentiality, and MUST support the replay protection mechanisms of
   IPsec.

   Diameter implementations MUST support IKE for peer authentication,
   negotiation of security associations, and key management, using the
   IPsec DOI [IPSECDOI]. Diameter implementations MUST support peer
   authentication using a pre-shared key, and MAY support certificate-
   based peer authentication using digital signatures. Peer
   authentication using the public key encryption methods outlined in
   IKE's Sections 5.2 and 5.3 [IKE] SHOULD NOT be used.

   Conformant implementations MUST support both IKE Main Mode and
   Aggressive Mode. When pre-shared keys are used for authentication,
   IKE Aggressive Mode SHOULD be used, and IKE Main Mode SHOULD NOT be
   used. When digital signatures are used for authentication, either
   IKE Main Mode or IKE Aggressive Mode MAY be used.

   When digital signatures are used to achieve authentication, an IKE
   negotiator SHOULD use IKE Certificate Request Payload(s) to specify
   the certificate authority (or authorities) that are trusted in
   accordance with its local policy. IKE negotiators SHOULD use
   pertinent certificate revocation checks before accepting a PKI
   certificate for use in IKE's authentication procedures.

   The Phase 2 Quick Mode exchanges used to negotiate protection for
   Diameter connections MUST explicitly carry the Identity Payload
   fields (IDci and IDcr). The DOI provides for several types of
   identification data. However, when used in conformant
   implementations, each ID Payload MUST carry a single IP address and a
   single non-zero port number, and MUST NOT use the IP Subnet or IP
   Address Range formats. This allows the Phase 2 security association
   to correspond to specific TCP and SCTP connections.

   Since IPsec acceleration hardware may only be able to handle a
   limited number of active IKE Phase 2 SAs, Phase 2 delete messages may
   be sent for idle SAs, as a means of keeping the number of active
   Phase 2 SAs to a minimum. The receipt of an IKE Phase 2 delete
   message SHOULD NOT be interpreted as a reason for tearing down a
   Diameter connection. Rather, it is preferable to leave the
   connection up, and if additional traffic is sent on it, to bring up
   another IKE Phase 2 SA to protect it. This avoids the potential for
   continually bringing connections up and down.

13.2. TLS Usage

   A Diameter node that initiates a connection to another Diameter node
   acts as a TLS client according to [TLS], and a Diameter node that
   accepts a connection acts as a TLS server. Diameter nodes
   implementing TLS for security MUST mutually authenticate as part of
   TLS session establishment. In order to ensure mutual authentication,
   the Diameter node acting as TLS server must request a certificate
   from the Diameter node acting as TLS client, and the Diameter node
   acting as TLS client MUST be prepared to supply a certificate on
   request.

   Diameter nodes MUST be able to negotiate the following TLS cipher
   suites:

      TLS_RSA_WITH_RC4_128_MD5
      TLS_RSA_WITH_RC4_128_SHA
      TLS_RSA_WITH_3DES_EDE_CBC_SHA

   Diameter nodes SHOULD be able to negotiate the following TLS cipher
   suite:

      TLS_RSA_WITH_AES_128_CBC_SHA

   Diameter nodes MAY negotiate other TLS cipher suites.

13.3. Peer-to-Peer Considerations

   As with any peer-to-peer protocol, proper configuration of the trust
   model within a Diameter peer is essential to security. When
   certificates are used, it is necessary to configure the root
   certificate authorities trusted by the Diameter peer. These root CAs
   are likely to be unique to Diameter usage and distinct from the root
   CAs that might be trusted for other purposes such as Web browsing.
   In general, it is expected that those root CAs will be configured so
   as to reflect the business relationships between the organization
   hosting the Diameter peer and other organizations. As a result, a
   Diameter peer will typically not be configured to allow connectivity
   with any arbitrary peer. When certificate authentication Diameter
   peers may not be known beforehand, and therefore peer discovery may
   be required.

   Note that IPsec is considerably less flexible than TLS when it comes
   to configuring root CAs. Since use of Port identifiers is prohibited
   within IKE Phase 1, within IPsec it is not possible to uniquely
   configure trusted root CAs for each application individually; the
   same policy must be used for all applications. This implies, for
   example, that a root CA trusted for use with Diameter must also be

trusted to protect SNMP. These restrictions can be awkward at best.
Since TLS supports application-level granularity in certificate
policy, TLS SHOULD be used to protect Diameter connections between
administrative domains. IPsec is most appropriate for intra-domain
usage when pre-shared keys are used as a security mechanism.

When pre-shared key authentication is used with IPsec to protect
Diameter, unique pre-shared keys are configured with Diameter peers,
who are identified by their IP address (Main Mode), or possibly their
FQDN (Aggressive Mode). As a result, it is necessary for the set of
Diameter peers to be known beforehand. Therefore, peer discovery is
typically not necessary.

The following is intended to provide some guidance on the issue.

It is recommended that a Diameter peer implement the same security
mechanism (IPsec or TLS) across all its peer-to-peer connections.
Inconsistent use of security mechanisms can result in redundant
security mechanisms being used (e.g., TLS over IPsec) or worse,
potential security vulnerabilities. When IPsec is used with
Diameter, a typical security policy for outbound traffic is "Initiate
IPsec, from me to any, destination port Diameter"; for inbound
traffic, the policy would be "Require IPsec, from any to me,
destination port Diameter".

This policy causes IPsec to be used whenever a Diameter peer
initiates a connection to another Diameter peer, and to be required
whenever an inbound Diameter connection occurs. This policy is
attractive, since it does not require policy to be set for each peer
or dynamically modified each time a new Diameter connection is
created; an IPsec SA is automatically created based on a simple
static policy. Since IPsec extensions are typically not available to
the sockets API on most platforms, and IPsec policy functionality is
implementation dependent, use of a simple static policy is the often
the simplest route to IPsec-enabling a Diameter implementation.

One implication of the recommended policy is that if a node is using
both TLS and IPsec, there is not a convenient way in which to use
either TLS or IPsec, but not both, without reserving an additional
port for TLS usage. Since Diameter uses the same port for TLS and
non-TLS usage, where the recommended IPsec policy is put in place, a
TLS-protected connection will match the IPsec policy, and both IPsec
and TLS will be used to protect the Diameter connection. To avoid
this, it would be necessary to plumb peer-specific policies either
statically or dynamically.

If IPsec is used to secure Diameter peer-to-peer connections, IPsec policy SHOULD be set so as to require IPsec protection for inbound connections, and to initiate IPsec protection for outbound connections. This can be accomplished via use of inbound and outbound filter policy.

14. References

14.1. Normative References

    [AAATRANS]   Aboba, B. and J. Wood, "Authentication, Authorization
                     and Accounting (AAA) Transport Profile", RFC 3539,
                     June 2003.

    [ABNF]   Crocker, D. and P. Overell, "Augmented BNF for Syntax
                     Specifications: ABNF", RFC 2234, November 1997.

    [ASSIGNNO]   Reynolds, J., "Assigned Numbers: RFC 1700 is Replaced
                     by an On-line Database", RFC 3232, January 2002.

    [DIFFSERV]   Nichols, K., Blake, S., Baker, F. and D. Black,
                     "Definition of the Differentiated Services Field (DS
                     Field) in the IPv4 and IPv6 Headers", RFC 2474,
                     December 1998.

    [DIFFSERVAF]   Heinanen, J., Baker, F., Weiss, W. and J. Wroclawski,
                     "Assured Forwarding PHB Group", RFC 2597, June 1999.

    [DIFFSERVEF]   Davie, B., Charny, A., Bennet, J., Benson, K., Le
                     Boudec, J., Courtney, W., Davari, S., Firoiu, V. and
                     D. Stiliadis, "An Expedited Forwarding PHB", RFC 3246,
                     March 2002.

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

    [EAP]   Blunk, L. and J. Vollbrecht, "PPP Extensible
                     Authentication Protocol (EAP)", RFC 2284, March 1998.

    [FLOATPOINT]   Institute of Electrical and Electronics Engineers,
                     "IEEE Standard for Binary Floating-Point Arithmetic",
                     ANSI/IEEE Standard 754-1985, August 1985.

    [IANA]   Narten, T. and H. Alvestrand, "Guidelines for Writing
                     an IANA Considerations Section in RFCs", BCP 26, RFC
                     2434, October 1998.

   [IANAADFAM]   IANA; "Address Family Numbers",
                    http://www.iana.org/assignments/address-family-numbers

   [IANAWEB]   IANA, "Number assignment", http://www.iana.org

   [IKE]   Harkins, D. and D. Carrel, "The Internet Key Exchange
                    (IKE)", RFC 2409, November 1998.

   [IPComp]   Shacham, A., Monsour, R., Pereira, R. and M. Thomas,
                    "IP Payload Compression Protocol (IPComp)", RFC 3173,
                    September 2001.

   [IPSECDOI]   Piper, D., "The Internet IP Security Domain of
                    Interpretation for ISAKMP", RFC 2407, November 1998.

   [IPV4]   Postel, J., "Internet Protocol", STD 5, RFC 791,
                    September 1981.

   [IPV6]   Hinden, R. and S. Deering, "IP Version 6 Addressing
                    Architecture", RFC 2373, July 1998.

   [KEYWORDS]   Bradner, S., "Key words for use in RFCs to Indicate
                    Requirement Levels", BCP 14, RFC 2119, March 1997.

   [NAI]   Aboba, B. and M. Beadles, "The Network Access
                    Identifier", RFC 2486, January 1999.

   [NAPTR]   Mealling, M. and R. Daniel, "The naming authority
                    pointer (NAPTR) DNS resource record," RFC 2915,
                    September 2000.

   [RADTYPE]   IANA, "RADIUS Types",
                    http://www.iana.org/assignments/radius-types

   [SCTP]   Stewart, R., Xie, Q., Morneault, K., Sharp, C.,
                    Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M.,
                    Zhang, L. and V. Paxson, "Stream Control Transmission
                    Protocol", RFC 2960, October 2000.

   [SLP]   Veizades, J., Guttman, E., Perkins, C. and M. Day,
                    "Service Location Protocol, Version 2", RFC 2165, June
                    1999.

   [SNTP]   Mills, D., "Simple Network Time Protocol (SNTP)
                    Version 4 for IPv4, IPv6 and OSI", RFC 2030, October
                    1996.

    [TCP]   Postel, J. "Transmission Control Protocol", STD 7, RFC
                       793, January 1981.

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

    [TLS]   Dierks, T. and C. Allen, "The TLS Protocol Version
                       1.0", RFC 2246, January 1999.

    [TLSSCTP]   Jungmaier, A., Rescorla, E. and M. Tuexen, "Transport
                       Layer Security over Stream Control Transmission
                       Protocol", RFC 3436, December 2002.

    [URI]   Berners-Lee, T., Fielding, R. and L. Masinter,
                       "Uniform Resource Identifiers (URI): Generic Syntax",
                       RFC 2396, August 1998.

    [UTF8]   Yergeau, F., "UTF-8, a transformation format of ISO
                       10646", RFC 2279, January 1998.

14.2. Informative References

    [AAACMS]   P. Calhoun, W. Bulley, S. Farrell, "Diameter CMS
                       Security Application", Work in Progress.

    [AAAREQ]   Aboba, B., Calhoun, P., Glass, S., Hiller, T., McCann,
                       P., Shiino, H., Zorn, G., Dommety, G., Perkins, C.,
                       Patil, B., Mitton, D., Manning, S., Beadles, M.,
                       Walsh, P., Chen, X., Sivalingham, S., Hameed, A.,
                       Munson, M., Jacobs, S., Lim, B., Hirschman, B., Hsu
                       R., Xu, Y., Campbell, E., Baba, S. and E. Jaques,
                       "Criteria for Evaluating AAA Protocols for Network
                       Access", RFC 2989, November 2000.

    [ACCMGMT]   Aboba, B., Arkko, J. and D. Harrington. "Introduction
                       to Accounting Management", RFC 2975, October 2000.

    [CDMA2000]   Hiller, T., Walsh, P., Chen, X., Munson, M., Dommety,
                       G., Sivalingham, S., Lim, B., McCann, P., Shiino, H.,
                       Hirschman, B., Manning, S., Hsu, R., Koo, H., Lipford,
                       M., Calhoun, P., Lo, C., Jaques, E., Campbell, E., Xu,
                       Y., Baba, S., Ayaki, T., Seki, T. and A. Hameed,
                       "CDMA2000 Wireless Data Requirements for AAA", RFC
                       3141, June 2001.

    [DIAMMIP]   P. Calhoun, C. Perkins, "Diameter Mobile IP
                       Application", Work in Progress.

   [DYNAUTH]   Chiba, M., Dommety, G., Eklund, M., Mitton, D. and B.
                 Aboba, "Dynamic Authorization Extensions to Remote
                 Authentication Dial In User Service (RADIUS)", RFC
                 3576, July 2003.

   [IANA-EXP]   T. Narten, "Assigning Experimental and Testing Numbers
                 Considered Useful", Work in Progress.

   [MIPV4]   Perkins, C., "IP Mobility Support for IPv4", RFC 3344,
                 August 2002.

   [MIPREQ]   Glass, S., Hiller, T., Jacobs, S. and C. Perkins,
                 "Mobile IP Authentication, Authorization, and
                 Accounting Requirements", RFC 2977, October 2000.

   [NASNG]   Mitton, D. and M. Beadles, "Network Access Server
                 Requirements Next Generation (NASREQNG) NAS Model",
                 RFC 2881, July 2000.

   [NASREQ]   P. Calhoun, W. Bulley, A. Rubens, J. Haag, "Diameter
                 NASREQ Application", Work in Progress.

   [NASCRIT]   Beadles, M. and D. Mitton, "Criteria for Evaluating
                 Network Access Server Protocols", RFC 3169, September
                 2001.

   [PPP]   Simpson, W., "The Point-to-Point Protocol (PPP)", STD
                 51, RFC 1661, July 1994.

   [PROXYCHAIN]   Aboba, B. and J. Vollbrecht, "Proxy Chaining and
                 Policy Implementation in Roaming", RFC 2607, June
                 1999.

   [RADACCT]   Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.

   [RADEXT]   Rigney, C., Willats, W. and P. Calhoun, "RADIUS
                 Extensions", RFC 2869, June 2000.

   [RADIUS]   Rigney, C., Willens, S., Rubens, A. and W. Simpson,
                 "Remote Authentication Dial In User Service (RADIUS)",
                 RFC 2865, June 2000.

   [ROAMREV]   Aboba, B., Lu, J., Alsop, J., Ding, J. and W. Wang,
                 "Review of Roaming Implementations", RFC 2194,
                 September 1997.

   [ROAMCRIT]   Aboba, B. and G. Zorn, "Criteria for Evaluating
                 Roaming Protocols", RFC 2477, January 1999.

   [SECARCH]   Kent, S. and R. Atkinson, "Security Architecture for
                      the Internet Protocol", RFC 2401, November 1998.

   [TACACS]   Finseth, C., "An Access Control Protocol, Sometimes
                      Called TACACS", RFC 1492, July 1993.

15. Acknowledgements

   The authors would like to thank Nenad Trifunovic, Tony Johansson and
   Pankaj Patel for their participation in the pre-IETF Document Reading
   Party. Allison Mankin, Jonathan Wood and Bernard Aboba provided
   invaluable assistance in working out transport issues, and similarly
   with Steven Bellovin in the security area.

   Paul Funk and David Mitton were instrumental in getting the Peer
   State Machine correct, and our deep thanks go to them for their time.

   Text in this document was also provided by Paul Funk, Mark Eklund,
   Mark Jones and Dave Spence. Jacques Caron provided many great
   comments as a result of a thorough review of the spec.

   The authors would also like to acknowledge the following people for
   their contribution in the development of the Diameter protocol:

   Allan C. Rubens, Haseeb Akhtar, William Bulley, Stephen Farrell,
   David Frascone, Daniel C. Fox, Lol Grant, Ignacio Goyret, Nancy
   Greene, Peter Heitman, Fredrik Johansson, Mark Jones, Martin Julien,
   Bob Kopacz, Paul Krumviede, Fergal Ladley, Ryan Moats, Victor Muslin,
   Kenneth Peirce, John Schnizlein, Sumit Vakil, John R. Vollbrecht and
   Jeff Weisberg.

   Finally, Pat Calhoun would like to thank Sun Microsystems since most
   of the effort put into this document was done while he was in their
   employ.

Appendix A. Diameter Service Template

   The following service template describes the attributes used by
   Diameter servers to advertise themselves. This simplifies the
   process of selecting an appropriate server to communicate with. A
   Diameter client can request specific Diameter servers based on
   characteristics of the Diameter 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:
      Diameter 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 Diameter peer without proper preconfigured
      secrets or cryptographic keys. Still, as Diameter 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 Diameter
      servers.

   Template text:
   -------------------------template begins here----------------------
   template-type=service:diameter

   template-version=0.0

   template-description=
     The Diameter protocol is defined by RFC 3588.

   template-url-syntax=
     url-path= ; The Diameter URL format is described in Section 2.9.
             ; Example: 'aaa://aaa.example.com:1812;transport=tcp
     supported-auth-applications= string L M
     # This attribute lists the Diameter applications supported by the
     # AAA implementation. The applications currently defined are:
     # Application Name   Defined by
     # ----------------   -----------------------------------
     # NASREQ   Diameter Network Access Server Application
     # MobileIP   Diameter Mobile IP Application
     #
     # Notes:
     #   . Diameter implementations support one or more applications.
     #   . Additional applications may be defined in the future.
     #   An updated service template will be created at that time.

```
      #
      NASREQ,MobileIP

      supported-acct-applications= string L M
      # This attribute lists the Diameter applications supported by the
      # AAA implementation.  The applications currently defined are:
      #  Application Name     Defined by
      #  ----------------     -----------------------------------
      #  NASREQ               Diameter Network Access Server Application
      #  MobileIP             Diameter Mobile IP Application
      #
      # Notes:
      #   . Diameter implementations support one or more applications.
      #   . Additional applications may be defined in the future.
      #     An updated service template will be created at that time.
      #
      NASREQ,MobileIP

      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----------------------
```

Appendix B. NAPTR Example

   As an example, consider a client that wishes to resolve aaa:ex.com.
   The client performs a NAPTR query for that domain, and the following
   NAPTR records are returned:

```
   ;;          order pref flags service            regexp  replacement
      IN NAPTR 50   50  "s"  "AAA+D2S"          ""
      _diameter._sctp.example.com IN NAPTR 100  50  "s"  "AAA+D2T"
      ""  _aaa._tcp.example.com
```

   This indicates that the server supports SCTP, and TCP, in that order.
   If the client supports over SCTP, SCTP will be used, targeted to a
   host determined by an SRV lookup of _diameter._sctp.ex.com. That
   lookup would return:

```
   ;;          Priority Weight Port   Target
      IN SRV  0        1      5060   server1.example.com IN SRV  0
      2      5060   server2.example.com
```

Appendix C. Duplicate Detection

   As described in Section 9.4, accounting record duplicate detection is
   based on session identifiers. Duplicates can appear for various
   reasons:

   - Failover to an alternate server. Where close to real-time
      performance is required, failover thresholds need to be kept low
      and this may lead to an increased likelihood of duplicates.
      Failover can occur at the client or within Diameter agents.

   - Failure of a client or agent after sending of a record from non-
      volatile memory, but prior to receipt of an application layer ACK
      and deletion of the record. record to be sent. This will result
      in retransmission of the record soon after the client or agent has
      rebooted.

   - Duplicates received from RADIUS gateways. Since the
      retransmission behavior of RADIUS is not defined within [RFC2865],
      the likelihood of duplication will vary according to the
      implementation.

   - Implementation problems and misconfiguration.

   The T flag is used as an indication of an application layer
   retransmission event, e.g., due to failover to an alternate server.
   It is defined only for request messages sent by Diameter clients or
   agents. For instance, after a reboot, a client may not know whether
   it has already tried to send the accounting records in its non-
   volatile memory before the reboot occurred. Diameter servers MAY use
   the T flag as an aid when processing requests and detecting duplicate
   messages. However, servers that do this MUST ensure that duplicates
   are found even when the first transmitted request arrives at the
   server after the retransmitted request. It can be used only in cases
   where no answer has been received from the Server for a request and
   the request is sent again, (e.g., due to a failover to an alternate
   peer, due to a recovered primary peer or due to a client re-sending a
   stored record from non-volatile memory such as after reboot of a
   client or agent).

   In some cases the Diameter accounting server can delay the duplicate
   detection and accounting record processing until a post-processing
   phase takes place. At that time records are likely to be sorted
   according to the included User-Name and duplicate elimination is easy
   in this case. In other situations it may be necessary to perform
   real-time duplicate detection, such as when credit limits are imposed
   or real-time fraud detection is desired.

In general, only generation of duplicates due to failover or re-
sending of records in non-volatile storage can be reliably detected
by Diameter clients or agents. In such cases the Diameter client or
agents can mark the message as possible duplicate by setting the T
flag. Since the Diameter server is responsible for duplicate
detection, it can choose to make use of the T flag or not, in order
to optimize duplicate detection. Since the T flag does not affect
interoperability, and may not be needed by some servers, generation
of the T flag is REQUIRED for Diameter clients and agents, but MAY be
implemented by Diameter servers.

As an example, it can be usually be assumed that duplicates appear
within a time window of longest recorded network partition or device
fault, perhaps a day. So only records within this time window need
to be looked at in the backward direction. Secondly, hashing
techniques or other schemes, such as the use of the T flag in the
received messages, may be used to eliminate the need to do a full
search even in this set except for rare cases.

The following is an example of how the T flag may be used by the
server to detect duplicate requests.

   A Diameter server MAY check the T flag of the received message to
   determine if the record is a possible duplicate. If the T flag is
   set in the request message, the server searches for a duplicate
   within a configurable duplication time window backward and
   forward. This limits database searching to those records where
   the T flag is set. In a well run network, network partitions and
   device faults will presumably be rare events, so this approach
   represents a substantial optimization of the duplicate detection
   process. During failover, it is possible for the original record
   to be received after the T flag marked record, due to differences
   in network delays experienced along the path by the original and
   duplicate transmissions. The likelihood of this occurring
   increases as the failover interval is decreased. In order to be
   able to detect out of order duplicates, the Diameter server should
   use backward and forward time windows when performing duplicate
   checking for the T flag marked request. For example, in order to
   allow time for the original record to exit the network and be
   recorded by the accounting server, the Diameter server can delay
   processing records with the T flag set until a time period
   TIME_WAIT + RECORD_PROCESSING_TIME has elapsed after the closing
   of the original transport connection. After this time period has
   expired, then it may check the T flag marked records against the
   database with relative assurance that the original records, if
   sent, have been received and recorded.

Appendix D.   Intellectual Property Statement

   The IETF takes no position regarding the validity or scope of any
   intellectual property or other rights that might be claimed to
   pertain to the implementation or use of the technology described in
   this document or the extent to which any license under such rights
   might or might not be available; neither does it represent that it
   has made any effort to identify any such rights. Information on the
   IETF's procedures with respect to rights in standards-track and
   standards-related documentation can be found in BCP-11. Copies of
   claims of rights made available for publication and any assurances of
   licenses to be made available, or the result of an attempt made to
   obtain a general license or permission for the use of such
   proprietary rights by implementers or users of this specification can
   be obtained from the IETF Secretariat.

   The IETF invites any interested party to bring to its attention any
   copyrights, patents or patent applications, or other proprietary
   rights which may cover technology that may be required to practice
   this standard. Please address the information to the IETF Executive
   Director.

Authors' Addresses

    Pat R. Calhoun
    Airespace, Inc.
    110 Nortech Parkway
    San Jose, California, 95134
    USA

    Phone: +1 408-635-2023
    Fax: +1 408-635-2020
    EMail: pcalhoun@airespace.com

    John Loughney
    Nokia Research Center
    Itamerenkatu 11-13
    00180 Helsinki
    Finland

    Phone: +358 50 483 6242
    EMail: john.Loughney@nokia.com

    Jari Arkko
    Ericsson
    02420 Jorvas
    Finland

    Phone: +358 40 5079256
    EMail: Jari.Arkko@ericsson.com

    Erik Guttman
    Sun Microsystems, Inc.
    Eichhoelzelstr. 7
    74915 Waibstadt
    Germany

    Phone: +49 7263 911 701
    EMail: erik.guttman@sun.com

    Glen Zorn
    Cisco Systems, Inc.
    500 108th Avenue N.E., Suite 500
    Bellevue, WA 98004
    USA

    Phone: +1 425 438 8218

Full Copyright Statement

Acknowledgement