Diameter Maintanence and
Extensions (DIME)
Internet-Draft
Intended status: Informational
Expires: October 20, 2007

V. Fajardo, Ed.
Toshiba America Research Inc.
T. Asveren
Sonus Network
H. Tschofenig
Siemens Networks GmbH & Co KG
G. McGregor
Alcatel-Lucent
J. Loughney
Nokia Research Center
April 18, 2007

Diameter Applications Design Guidelines
draft-fajardo-dime-app-design-guide-02.txt

Status of this Memo

Copyright Notice

Abstract

The Diameter Base protocol provides rules on how to extend Diameter

and to create new Diameter applications. This is a companion document to clarify these rules. This document does not intended to add, remove or change these rules, rather it helps protocol designers to extend Diameter.

Table of Contents

1. Introduction

The Diameter Base protocol document defines rules on how one would extend Diameter (see Section 1.2 of [1]). In the context of this document, extending Diameter means that a new Diameter application is being defined which may or may not be based on an existing Diameter application. A decision to define a new application would mean allocation of a new application ID.

By themselves, the rules defined in the Diameter Base protocol are not necessarily comprehensive enough that one can easily derive good design decisions from them. The effect of this can be seen in various attempts to extend Diameter where protocol designers have no clear answer on whether to even define a new application or not. At worst, some existing Diameter applications that had purposely been derived from another existing application resulted in some in-appropriate design decision in which both applications are no longer interoperable in certain conditions.

The intent of this document is to influence ongoing and future Diameter application design by providing the following content:

- o Clarify existing Diameter extensibility rules present in the Diameter Base Protocol.
- o Clarify usage of certain Diameter functionality which are not explicitly described in the Diameter Base specification.
- o Discuss design choices when defining new applications.
- o Present tradeoffs of design choices.

Note that it is not always possible to offer a complete and concise answer to certain design choices. There is, however, the belief that at a minimum, this document can be used as a guide to Diameter extensibility.

2. Terminology

This document reuses the terminology used in [1].

3. Diameter Application Model

As it is currently interpreted and practiced, the Diameter Base protocol is a two-layer protocol. The lower layer is mainly responsible for managing connections between neighboring peers and

for message routing. The upper layer is where the Diameter applications reside. This model is inline with a Diameter node having an application layer and a peer-to-peer delivery layer. The Diameter Base protocol document completely defines the architecture and behavior of the message delivery layer and then provides the framework for designing Diameter applications on the application layer. This framework includes definitions of application sessions and accounting support (see Section 8 and 9 of [1]). The remainder of this document also treats a Diameter node as a single instance of a Diameter message delivery layer and one or more Diameter applications using it.

4. Rules on Diameter Extensibility

The general theme of Diameter extensibility is to reuse AVPs, AVP values, commands and applications as much as possible. However, there are also rules for extending Diameter as specified in Section 1.2 of [1]. As is, the rules apply to the scenario where one is trying to define a new Diameter application. Defining a new Diameter application can be done by:

Defining a completely new application

This case applies to applications which have requirements that cannot be filled by existing applications and would require definition of new command(s), AVPs and AVP values. Typically, there is little ambiguity about the decision to create these types of applications. Some examples are the interfaces defined for the IP Multimedia Subsystem of 3GPP, i.e.; Cx/Dx ([2] and [3]), Sh ([4] and [5]) etc . Though some decisions may be clear, designers should also consider certain aspects of the application itself. Some of these are described in Section 5. Applications design should also follow the theme of Diameter extensibility which advocates reuse of AVPs and AVP values as much as possible even in newly defined commands. In certain cases where accounting will be used, the models described in Section 5.1 should be considered.

Extending an existing application

In this case, the requirements of the new applications are not completely unique and there are existing application's that can be reused to solve some or all of the application requirements. Thus, there is a greater likelihood of ambiguity on how much of the existing application can be reused, to what extent and what the implications for both the new and existing application. Section 4.1 discusses some of the issues in this case.

4.1. Rules on Extending Existing Applications

The Diameter base protocol provides a clear set of rules on when one should define a new Diameter application. In the context of this document, the rules are:

Adding an AVP to a command ABNF of an existing application

The rules are strict in the case where the AVP(s) to be added is mandatory to be understood and interpreted. This means that the M-bit is set and the AVP(s) is required to exist in command ABNF. Note that this mandatory AVP rule applies to AVP(s) that either already exist in the same or in another application or the AVP(s) are yet to be defined. In the latter case, the ambiguity arises when trying to decide whether the AVP(s) should be mandatory or not. There are several questions that application designers should contemplate when trying to decide:

* Does the AVP(s) change the state machine of the application ?

* Would the presence of the AVP(s) cause additional message round-trips; effectively changing the state machine of the application ?

* Will the AVP be used to fulfill new required functionality ?

* Would the AVP be used to differentiate between old and new versions of the same application ?

* Will it have duality in meaning; i.e., be used to carry application related information as well as be used to indicate that the message is for a new application ?

These questions are not comprehensive in any way but in all cases the semantics of the application must change to justify the use of mandatory AVPs.

However, care should also be taken when opting for optional AVPs instead of mandatory AVPs simply to avoid allocating new applications. Optional AVPs that fall into any of the categorical questions above would have consequences. See Section 5.4 for details.

Add a new AVP value to an to an existing AVP

In this case, the rule applies to existing mandatory AVPs already present in a command ABNF where the semantics of the AVP changes. This means that the meaning or usage of the AVP has changed and

significantly affects the behavior of the application. Although this case may be less common or seem more subtle, the exact same considerations given in the first scenario above apply here as well.

Add a command to an existing application

In this case, the rule applies to defining a new command for an existing application or importing an existing command from another application so as to inherit some or all of the functionality of that application. In the first case, the decision is straight forward since this is typically a result of adding new functionality that does not yet exist. The latter case would result in a new application but it has a more subtle issue such as deciding whether importing of commands and functionality is really better than simply using the existing application as it is in conjunction with any new application.

A typical example would be the Diameter MIPv6 split scenario (see [6]) in which several application models would have been possible during the design phase; one model would reuse existing Diameter EAP application combined with a new Diameter MIPv6 application to form a complete authentication and authorization scheme and another would be to reuse Diameter EAP like commands within the new Diameter MIPv6 application to accomplish the same result. In this case, the latter model was chosen which would permit the reuse of commands and/or AVPs from one application to another. Other applications such as Diameter QoS (see [7]) would likely face similar decisions.

In general, it is difficult to come to a hard and fast guideline for this scenario so a case by case study of each application requirement should be applied. Before importing a command, application designers should consider whether:

* The existing application can be reused as is without fundamental changes; i.e. an optional AVP is sufficient to indicate support for new optional functionality if any. There are pitfalls to this as well. See Section 5.4

* Reuse of existing applications would result in a distributed environment which may not be conducive to certain requirements of the applications; i.e. security and or deployment difficulties - because of Diameter routing, messages for different applications providing service to the same user may end up in different servers would then need to be co-related. This could mean extra signaling between application servers. A typical example would be the initial proposal for Diameter

MIPv6 split scenario (see [6]) where authorization and authentication is separated.

5. Design Considerations

The following are some of the design considerations that apply to a Diameter application.

5.1. Diameter Accounting Support

Accounting can be treated as an auxiliary application which is used in support of other applications. In most cases, accounting support is required when defining new applications. However, the lack of clarity in the base protocol document has prevented easy use the base accounting messages (ACR/ACA). This document provides two(2) possible models for using accounting:

Split Accounting Model

In this model, the accounting messages will use the Diameter base accounting application ID (value of 3). The design implication for this is that the accounting is treated as an independent application, especially during routing. This means that accounting commands emanating from an application may be routed separately from the rest of the other application messages. This also implies that the messages generally end up in a central accounting server. A split accounting model is a good design choice when:

* The application itself will not define its own unique accounting commands.

* The overall system architecture permits the use of centralized accounting for one or more Diameter applications.

From a Diameter architecture perspective, this model should be the typical design choice. Note that when using this model, the accounting server must use the Acct-Application-Id AVP to determine which application is being accounted for. Therefore, the application designer should specify the proper values used in Acct-Application-Id AVP when sending ACR messages.

Coupled Accounting Model

In this model, the accounting messages will use the application ID of the application using the accounting service. The design implication for this is that the accounting messages is tightly coupled with the application itself; meaning that accounting messages will be routed like any other application messages. It would then be the responsibility of the application server (application entity receiving the ACR message) to send the accounting records carried by the accounting messages to the proper accounting server. The application server is also responsible for formulating a proper response (ACA). A coupled accounting model is a good design choice when:

* The system architecture or deployment will not provide an accounting server that supports Diameter.

* The system architecture or deployment requires that the accounting service for the specific application should be handled by the application itself.

* The application server is provisioned to use a different protocol to access the accounting server; i.e., via LDAP, XML etc. This includes attempting to supporting older accounting systems that are not Diameter aware.

In all cases above, there will generally be no direct Diameter access to the accounting server.

These models provide a basis for using accounting messages. Application designers may obviously deviate from these models provided that the factors being addressed here have also been taken into account. Though it is not recommended, examples of other methods would be defining a new set of commands to carry application specific accounting records.

Additionally, the application ID in the message header and Accounting-Application-Id AVP are populated depending on the accounting model used for a specific application, as described in [1]. Therefore, application designers have to specify the accounting model used to guarantee proper routing of accounting requests.

5.2. Generic Diameter Extensions

Generic Diameter extensions are AVPs, commands or applications that are designed to support other Diameter applications. They are auxiliary applications meant to improve or enhance the Diameter protocol itself or Diameter applications/functionality. Some

examples include the extensions to support auditing and redundancy (see [8]), improvements in duplicate detection scheme (see [9]).

Since generic extensions can cover many aspects of Diameter and Diameter applications, it is not possible to enumerate all the probable scenarios in this document. However, some of the most common considerations are as follows:

- o Backward compatibility: Dealing with existing applications that do not understand the new extension. Designers also have to make sure that new extensions do not break expected message delivery layer behavior.

- o Forward compatibility: Making sure that the design will not introduce undue restrictions for future applications. Future applications attempting to support this feature should not have to go through great lengths to implement any new extensions.

- o Tradeoffs in signaling: Designers may have to choose between the use of optional AVPs piggybacked onto existing commands versus defining new commands and applications. Optional AVPs are simpler to implement and may not need changes to existing applications; i.e., use of proxy agents. However, the drawback is that the timing of sending extension data will be tied to when the application would be sending a message. This has consequences if the application and the extensions have different timing requirements. The use of commands and applications solves this issue but the tradeoff is the additional complexity of defining and deploying a new application. It is left up to the designer to find a good balance among these tradeoffs based on the requirements of the extension.

5.3. Updating an existing Application

An application that is being upgraded must follow the same rules mentioned Section 4. Even if the new version is fundamentally the same application, allocation of a new application ID is possible if it meets those criteria.

Optional AVPs can also be used to indicate version differences. If this approach is chosen, it is recommended that the optional AVP is used specifically to indicate version information only and nothing else. Additionally, the use of too many optional AVPs to carry application enhancements should be avoided since such approach has a tendency to become unmanageable and introduce interoperability issues. These pitfalls are discussed in Section 5.4

For the same reason, care should be taken in attempting to justify allocation of new application ID for every change. The pitfalls of this approach is discussed in Section 5.6.

5.4. Use of optional AVPs

Problems arise when there is a tendency by applications designers to keep adding optional AVPs to an existing command so they can circumvent the extension rules in Section 4. Some of the pitfalls that application designers should avoid are:

- o Use of optional AVPs with intersecting meaning; one AVP has partially the same usage and/or meaning as another AVP. The presence of both can lead to confusion.

- o Optional AVPs with dual purpose; i.e.; to carry applications data as well as to indicate support for one or more features. This has a tendency to introduce interpretation issues.

- o Use of optional AVPs with a minimum occurrence of one(1) in the command ABNF. This is generally contradictory. Application designers should not use this scheme to circumvent definition of mandatory AVPs.

All of these practices generally result in interoperability problems so they should be avoided as much as possible.

5.5. Deleting AVPs from a Command ABNF

Although this scenario is not as common, the deletion of AVPs from a command ABNF is significant when trying to extend an existing application. Deletion can be categorized between deletion of mandatory and optional AVPs.

In the unlikely event that an application designer would require that mandatory AVPs must be deleted then it constitutes a fundamental change to an existing application. Though not specified in [1], deletion of mandatory would require the allocation of a new application since it dictates changes in the behavior and semantics of an application.

The deletion of an optional AVP may not necessarily indicate allocation of a new application. An optional AVP with a minimum occurrence of at least one(1) in the command ABNF would mean that the AVP is required and that if deleted, there would effectively be changes to the behavior of the application as well. Such cases are highly dubious to begin with since those AVPs already exhibits properties of mandatory AVPs. It should therefore fall into the

category of deleting mandatory AVPs.

In other cases, it is recommended that application designers reuse the command ABNF as is and safely ignore (but not delete) any optional AVP that will not be used. This is to maintain compatibility with existing applications that will not know about the new functionality as well as maintain the integrity of existing dictionaries.

5.6. Justifying the Allocation of Application-Id

Application designers should avoid justifying the allocation of application IDs for every change that is made to an existing application. Proliferation of application ID can lead to confusion and an in-efficient use of the application ID namespaces. Application designers should always use Section 4 as a basis for justifying allocation of a new application ID.

5.7. Use of Application-Id in a Message

When designing new applications, designers should specify that the application ID carried in all session level messages must be the application ID of the application using those messages. This includes the session level messages defined in base protocol, i.e., RAR/RAA, STR/STA, ASR/ASA and possibly ACR/ACA in the coupled accounting model, see Section 5.1. Existing specifications may not adhere to this rule for historical or other reasons. However, this scheme is followed to avoid possible routing problems for these messages.

Additionally, application designers using the Vendor-Specific-Application-Id AVP should note that the Vendor-Id AVP will not be used in any way by the Diameter message delivery layer. Therefore its meaning and usage should be segregated only within the application.

5.8. Support for Server Initiated Requests

Section 8 of [1] implies that only client sessions can initiate a request messages. Assuming that the Diameter client and server sessions can be de-coupled from application client and server entities, an application design requiring server initiated request can simply create a Diameter client session and the client entity can then initiate a corresponding Diameter server session.

5.9. System Architecture and Deployment

The following are some of the architecture considerations that applications designers should contemplate when defining new applications:

- For general AAA applications, Diameter requires more message exchanges for the same set of services compared to RADIUS. Therefore, application designers should consider scalability issues during the design process.

- Application design should be agnostic to any Diameter topology. Application designers should not always assume a particular Diameter topology; i.e., assume that there will always be application proxies in the path or assume that only intra-domain routing is applicable.

- Security Considerations. Application designers should take into account that there is no end-to-end authentication built into Diameter.

- Application design should consider some form of redundancy. Session state information is the primary data necessary for backup/recovering endpoints to continue processing for an previously existing session. Carrying enough information in the messages to reconstruct session state facilitates redundant implementations and is highly recommended.

- Application design should segregate message delivery layer processing from application level processing. An example is the use of timers to detect lack of a response for a previously sent requests. Although the Diameter base protocol defines a watchdog timer Tw, its use on application level is discouraged since Tw is a hop-by-hop timer and it would not be relevant for end-to-end message delivery error detection. In such a case, it is recommended that applications should define their own set of timers for such purpose.

6. IANA Considerations

This document does not require actions by IANA.

7. Security Considerations

This document does provides guidelines and considerations for extending Diameter and Diameter applications. It does not define nor

address security related protocols or schemes.

8. Acknowledgments

We greatly appreciate the insight provided by Diameter implementers who have highlighted the issues and concerns being addressed by this document.

9. References

9.1. Normative References

[1] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

[2] 3GPP, "IMS Cx and Dx interfaces : signalling flows and message contents", 3GPP TS 29.228 Version 7.0.0 2006.

[3] 3GPP, "IMS Cx and Dx interfaces based on the Diameter protocol; Protocol details", 3GPP TS 29.229 Version 7.0.0 2006.

[4] 3GPP, "IMS Sh interface : signalling flows and message content", 3GPP TS 29.328 Version 6.8.0 2005.

[5] 3GPP, "IMS Sh interface based on the Diameter protocol; Protocol details", 3GPP TS 29.329 Version 6.6.0 2005.

[6] Bournelle, J., "Diameter Mobile IPv6: HA-to-AAAH support", draft-ietf-dime-mip6-split-01 (work in progress), October 2006.

[7] Zorn, G., "Diameter Quality of Service Application", draft-ietf-dime-diameter-qos-00 (work in progress), February 2007.

9.2. Informative References

[8] Calhoun, P., "Diameter Resource Management Extensions", draft-calhoun-diameter-res-mgmt-08.txt (work in progress), March 2001.

[9] Asveren, T., "Diameter Duplicate Detection Cons.", draft-asveren-dime-dupcons-00 (work in progress), August 2006.

Authors' Addresses

Victor Fajardo (editor)
Toshiba America Research Inc.
One Telcordia Drive
Piscataway, NJ 08854
USA

Email: vfajardo@tari.toshiba.com

Tolga Asveren
Sonus Network
4400 Route 9 South
Freehold, NJ, 07728
USA

Email: tasveren@sonusnet.com

Hannes Tschofenig
Siemens Networks GmbH & Co KG
Otto-Hahn-Ring 6
Munich, Bavaria 81739
Germany

Phone: +49 89 636 40390
Email: Hannes.Tschofenig@siemens.com
URI: http://www.tschofenig.com

Glenn McGregor
Alcatel-Lucent
USA

Email: glenn@aaa.lucent.com

John Loughney
Nokia Research Center
USA

Email: john.loughney@nokia.com

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).