Diameter Maintenance and Extensions
(DIME)
Internet-Draft
Intended status: Informational
Expires: October 3, 2012

L. Morand, Ed.
Orange Labs
V. Fajardo

H. Tschofenig
Nokia Siemens Networks

April 1, 2012

Diameter Applications Design Guidelines
draft-ietf-dime-app-design-guide-14

Abstract

The Diameter Base protocol provides facilities for protocol extensibility enabling to define new Diameter applications or modify existing applications. This document is a companion document to the Diameter Base protocol that further explains and clarifies the rules to extend the Diameter Base protocol. It is meant as a guidelines document and therefore it does not add, remove or change existing rules.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on October 3, 2012.

Copyright Notice

Table of Contents

1. Introduction

The Diameter Base protocol provides facilities to extend the Diameter Base protocol (see Section 1.3 of [I-D.ietf-dime-rfc3588bis]) for supporting new functionalities. In the context of this document, extending Diameter means one of the following:

1. Addition of a new functionality to an existing Diameter application without defining a new application.

2. Addition of a new functionality to an existing Diameter application that requires the definition of a new application.

3. The definition of a new Diameter application to provide a set of functionalities not supporting by existing applications.

4. The definition of a new generic functionality that can be reused across different applications.

All of these choices are design decisions that can done by any combination of reusing existing or defining new commands, AVPs or AVP values. Protocol designers do, however, not have total freedom when making their design. A number of rules defined in [I-D.ietf-dime-rfc3588bis] place constraints on when an extension demands a new Diameter application to be defined or a new command code to be registered. The objective of this document is the following:

- Clarify updated Diameter extensibility rules in the Diameter Base Protocol.

- Clarify usage of certain Diameter functionalities that are not explicitly described in the Diameter Base specification.

- Discuss design choices and provide guidelines when defining applications.

- Present tradeoffs of design choices.

2. Terminology

This document reuses the terminology used in [I-D.ietf-dime-rfc3588bis].

3. Overview

As designed, the Diameter Base protocol can be seen as 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 in line 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 [I-D.ietf-dime-rfc3588bis]). 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.

The Diameter protocol is designed to be extensible and the principles are descibed in the section 1.3 of [I-D.ietf-dime-rfc3588bis]. Extending Diameter can mean the definition of a new Diameter application and/or the reuse of commands, AVPs and AVP values in any combination for the purpose of inheriting the features of an existing Diameter application. The reuse recommendation is meaningful as most of the requirements defined for a new application are likely already fulfilled by an existing application.

However, when reusing existing applications, there is a greater likelihood of ambiguity on how much of the existing application can be enhanced without being distorted too much and therefore requiring the definition of a new application.

The impacts of extending existing applications can be categorized as follow:

Minor Extension: Enhancing the functional scope of an existing application by the addition of optional features to support. Such enhancement has no backward compatibility issue with the existing application. A typical example would be the definition of a new optional AVP to use in an existing command. In general, this includes everything that is not covered by the next category. The standardization effort will be fairly small.

Major Extension: Enhancing the functional scope of an existing application in such a way that this implies backward compatible change to the existing application and then requires the definition of a new Diameter application. A typical example would be the creation of a new command for providing functionality not

supported by existing applications. For such extension, a significant specification effort is required and a carefull approach is recommended.

The rules outlined in the section 1.3 of [I-D.ietf-dime-rfc3588bis] indicate when an extension requires a new command code to be registered and when new Diameter applications have to be defined. The subsequent sections further explain and clarify the rules to extend the Diameter Base protocol. It is meant as a guidelines document and therefore it does not add, remove or change existing rules.

4. Reusing existing Diameter applications

When selecting the Diameter Base protocol to support new functionalities, protocol designers are advised to try to re-use as much as possible existing Diameter applications to simplify standardization, implementation and avoid potential interoperability issues. However, existing application needs to be adapted to support new requirements and these modifications can be at the command level and/or at the AVP level. The following sections describe the possible modifications that can be performed on existing applications and their related impacts.

4.1. Adding a new command

Adding a new command is considered as a major extension and requires a new Diameter application to be defined. Adding a new command to an application means either defining a completely new command or importing the command's CCF syntax specification from another application whereby the new application inherits some or all of the functionality of the application where the command came from. In the former case, the decision to create an new application is straightforward since this is typically a result of adding a new functionality that does not exist yet. For the latter, the decision to create a new application will depend on whether importing the command in a new application is more suitable than simply using the existing application as it is in conjunction with any other application. Therefore, a case by case study of each application requirement should be applied.

An illustrative example is the command pair defined in Diameter EAP application [RFC4072] that can be re-used conjointly with any other application (e.g. the Diameter NASREQ application [RFC4005]) as soon as standard EAP-based authentication procedures need to be supported by the implementation. It may therefore not be required to import the command pair in the new defined application.

However, in general, it is difficult to come to a hard guideline, and so a case by case study of each application requirement should be applied. Before adding or importing a command, application designers should consider the following:

- Can the new functionality be fulfilled by creating a new command independent from any existing command? In this case, the resulting new application and the existing application can work independent of, but cooperating with each other.

- Can the existing command be reused without major extensions and therefore without the need for the definition of a new application, e.g. new funtionality introduced by the creation of new optional AVPs.

- Care should be taken to avoid a liberal method of importing existing command's CCF syntax specification. This would result in a monolithic and hard to manage applications supporting too many different functionalities and can cause interoperability issues between the different applications. .

4.2. Deleting a command

Although this process is not typical, removing a command to an application requires a new Diameter application to be defined. this is due to the fact that the reception of the deleted command would systematically result in a protocol error (DIAMETER_COMMAND_UNSUPPORTED).

It is unusual to delete an existing command from an application for the sake of deleting it or the functionality it represents. This normally indicates of a flawed design. An exception might be if the intent of the deletion is to create a newer version of the same application which is somehow simpler than the previous version.

4.3. Reusing existing commands

This section discusses rules in adding and/or deleting AVPs from an existing command of an existing application. The cases described in this section may not necessarily result in the creation of new applications.

It is worth to note that the strong recommendation to re-use existing commands in the [RFC3588] was to prevent rapid scarcity of code values available for vendor-specific commands. [I-D.ietf-dime-rfc3588bis] relaxes the policy with respect to the allocation of command codes for vendor-specific uses and enlarges the range of available code values for vendor-specific applications. Therefore, if it is still recommended to re-use as much as possible existing commands, protocol designers can consider more easily the definition of a new command when it is a solution more suitable than twisting existings command use and applications.

4.3.1. Adding AVPs to a command

Based on the rules in [I-D.ietf-dime-rfc3588bis], AVPs that are added to an existing command can be categorized into:

- Mandatory (to understand) AVPs. As defined in [I-D.ietf-dime-rfc3588bis], these are AVPs with the M-bit flag set, which means that a Diameter node receiving are required to understand not only their values but their semantics. Failure to do so will cause an message handling error. This is regardless of whether these AVPs are required or optional as specified by the command's CCF syntax specification.

- Optional (to understand) AVPs. As defined in [I-D.ietf-dime-rfc3588bis], these are AVPs with the M-bit flag cleared, which mean that a Diameter node receiving these AVP can simply ignore them if not supported in the process of the received command.

The rules are strict in the case where the AVPs to be added are mandatory to understand i.e. with the M-bit set. A mandatory AVP cannot be added to an existing command without defining a new Diameter application, as stated in [I-D.ietf-dime-rfc3588bis]. This falls into the "Major Extensions" category. Despite the clarity of the rule, ambiguity still arises when evaluating whether a new AVP being added should be mandatory to begin with. Here is a list of few common questions that application designers should wonder when trying to decide:

- Would it be required for the receiving side to be able to process and understand the AVP and its content?

- Would the new AVPs change the state machine of the application?

- Would the presence of the new AVP lead to a different number of roundtrips, effectively changing the state machine of the application?

- Would the new AVP be used to differentiate between old and new versions of the same application whereby the two versions are not backward compatible?

- Would the new AVP 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?

When one of the above questions can be answered in the affirmative then the M-bit has to be set for the new AVP.

If application designers are instead contemplating on the use of optional AVPs i.e. with the M-bit cleared, then the following are some of the pitfalls that should be avoided:

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

- An 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.

- Adding one or more optional AVPs and indicating (usually within descriptive text for the command) that at least one of them has to be present in the command. This essentially circumventing the ABNF and is equivalent to adding a mandatory AVPs to the command.

These practices generally result in interoperability issues and should be avoided as much as possible.

4.3.2. Deleting AVPs from a Command

When deleting an AVP from a command, the following cases need to be differentiated:

- Deleting an AVP that is indicated as { AVP } in the command's CCF syntax specification, whatever the setting of the M-bit set. This means the definition of a new command. In this case, a new command code and subsequently a new Diameter application have to be specified.

- Deleting an AVP with M-bit set that is indicated as [AVP] in the command's CCF syntax specification. No new command code has to be specified but the definition of a new Diameter application is required.

- Deleting an AVP with the M-bit cleared that is indicated as [AVP] in the command's CCF syntax specification. In this case, the AVP can be deleted without consequences.

If possible application designers should attempt the reuse the command's CCF syntax specification without modification and simply 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.

4.4. Reusing existing AVPs

This section discusses rules in reusing existing AVP when reusing an existing command or defining a new command in a new application.

4.4.1. Setting of the AVP flags

When reusing AVPs in a new application, the AVP flag setting, such as the mandatory flag ('M'-bit), has to be re-evaluated for a new Diameter application and, if necessary, even for every command within the application. In general, for AVPs defined outside of the base protocol, its mandatory characteristics are tied to its role within an application and command.

All other AVP flags shall remain unchanged

4.4.2. Reuse of AVP of type Enumerated

When modifying the set of values supported by an AVP of type Enumerated, this means defining a new AVP. Modifying the set of Enumerated values includes adding a value or deprecating the use of a value defined initially for the AVP. Defining a new AVP will avoid interoperability issues.

5. Rules for new Applications

The general recommendation for Diameter extensibility is to reuse commands, AVPs and AVP values as much as possible. However, some of the extensibility rules described in the previous section also apply to scenarios where a designer is trying to define a completely new Diameter application.

This section discusses the case where new applications have requirements that cannot be filled by existing applications and would require definition of completely new commands, AVPs and/or 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 ([TS29.228] and [TS29.229]), Sh ([TS29.328] and [TS29.329]) etc.

Application designers should also follow the theme of Diameter extensibility which in this case means to import existing AVPs and AVP values for any newly defined commands. In certain cases where accounting will be used, the models described in Section 7 should also be considered. Though some decisions may be clear, designers should also consider certain aspects of defining a new application. Some of these aspects are described in following sections.

5.1. 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 7. Existing specifications may not adhere to this rule for historical or other reasons. However, this scheme should be followed to avoid possible routing problems for these messages.

In general, when a new application has been allocated with a new application id and it also reuses existing commands with or without modifications (Sec 4.1), it must use the newly allocated application id in the header and in all relevant application id AVPs (Auth-Application-Id or Acct-Application-Id) present in the commands message body.

Additionally, application designs using Vendor-Specific-Application-Id AVP should not use the Vendor-Id AVP to further dissect or differentiate the vendor-specification application id. Diameter routing is not based on the Vendor-Id. As such, the Vendor-ID should not be used as an additional input for

routing or delivery of messages. In general, the Vendor-Id AVP is an informational AVP only and kept for backward compatibility reasons.

5.2. Application Specific Session State Machine

Section 8 of [I-D.ietf-dime-rfc3588bis] provides session state machines for authentication, authorization and accounting (AAA) services. When a new application is being defined that cannot clearly be categorized into any of these services it is recommended that the application itself define its own session state machine. The existing session state machines defined by [I-D.ietf-dime-rfc3588bis] is not intended for general use beyond AAA services, therefore any behavior not covered by that category would not fit well. Support for server initiated request is a clear example where an application specific session state machine would be needed, for example, the Rw interface for ITU-T push model (cf.[Q.3303.3]).

6. End-to-End Applications Capabilities Exchange

It is also possible that applications can use optional AVPs to exchange application specific capabilities and features. These AVPs are exchanged on an end-to-end basis. Examples of this can be found in [I-D.ietf-dime-mip6-integrated] and [I-D.ietf-dime-qos-attributes].

The end-to-end capabilities AVPs can aid in the following cases:

- Formalizing the way new functionality is added to existing applications by announcing support for it.

- Applications that do not understand these AVP can discard it upon receipt. In such case, senders of the AVP can also safely assume the receiving end-point does not support any functionality carried by the AVP if it is not present in subsequent responses.

- Useful in cases where deployment choices are offered and the generic design can be made available for a number of applications.

Note that this list is not meant to be comprehensive.

When used in a new application, protocol designers should clearly specify this end-to-end capabilities exchange and the corresponding behaviour of the Diameter nodes supporting the application.

7. 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. 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 Diameter routing. This means that accounting commands emanating from an application may be routed separately from the rest of the other application messages. This may also imply 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.

Centralizing accounting may have advantages but there are also drawbacks. The model assumes that the accounting server can somehow differentiate received accounting messages. Since the received accounting messages can be for any application and/or service, the accounting server has to be have a method to uniquely match accounting messages with applications and/or services being accounted for. This may mean defining new AVPs, checking the presence, absence or contents of existing AVPs or checking the contents of the accounting records itself. But in general, there is no clean and generic scheme for sorting these messages. Therefore, the use of this model is recommended only when all received accounting messages can be clearly identified and sorted. For most cases, the use of Coupled Accounting Model is recommended.

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 are 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; e.g., via LDAP, SOAP etc. This includes attempting to support 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 might be defining a new set of commands to carry application specific accounting records.

8. 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 [I-D.calhoun-diameter-res-mgmt]), improvements in duplicate detection scheme (see [I-D.asveren-dime-dupcons]), and piggybacking of QoS attributes (see [I-D.ietf-dime-qos-attributes]).

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:

- 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.

- 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.

- 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; 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.

In practice, it is often the case that the generic extensions use optional AVPs because it's simple and not intrusive to the application that would carry it. Peers that do not support the generic extensions need not understand nor recognize these optional AVPs. However, it is recommended that the authors of the extension

specify the context or usage of the optional AVPs. As an example, in the case that the AVP can be used only by a specific set of applications then the specification must enumerate these applications and the scenarios when the optional AVPs will be used. In the case where the optional AVPs can be carried by any application, it is should be sufficient to specify such a use case and perhaps provide specific examples of applications using them.

In most cases, these optional AVPs piggybacked by applications would be defined as a Grouped AVP and it would encapsulate all the functionality of the generic extension. In practice, it is not uncommon that the Grouped AVP will encapsulate an existing AVP that has previously been defined as mandatory ('M'-bit set) e.g., 3GPP IMS Cx / Dx interfaces ([TS29.228] and [TS29.229]).

9. IANA Considerations

This document does not require actions by IANA.

10. 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.

11. Contributors

The content of this document was influenced by a design team created to revisit the Diameter extensibility rules. The team consisting of the members listed below was formed in February 2008 and finished its work in June 2008.

- Avi Lior
- Glen Zorn
- Jari Arkko
- Lionel Morand
- Mark Jones
- Victor Fajardo
- Tolga Asveren
- Jouni Korhonen
- Glenn McGregor
- Hannes Tschofenig
- Dave Frascone

We would like to thank Tolga Asveren, Glenn McGregor, and John Loughney for their contributions as co-authors to earlier versions of this document.

12. Acknowledgments

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

13. References

13.1. Normative References

[I-D.ietf-dime-rfc3588bis]
Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", draft-ietf-dime-rfc3588bis-31 (work in progress), March 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

13.2. Informative References

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

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

[I-D.ietf-dime-mip6-integrated]
Korhonen, J., Bournelle, J., Tschofenig, H., Perkins, C., and K. Chowdhury, "Diameter Mobile IPv6: Support for Network Access Server to Diameter Server Interaction", draft-ietf-dime-mip6-integrated-12 (work in progress), January 2009.

[I-D.ietf-dime-qos-attributes]
Korhonen, J., Tschofenig, H., Arumaithurai, M., Jones, M., and A. Lior, "Traffic Classification and Quality of Service Attributes for Diameter", draft-ietf-dime-qos-attributes-15 (work in progress), December 2009.

[Q.3303.3]
3rd Generation Partnership Project, "ITU-T Recommendation Q.3303.3, "Resource control protocol no. 3 (rcp3): Protocol at the Rw interface between the Policy Decision Physical Entity (PD-PE) and the Policy Enforcement Physical Entity (PE-PE): Diameter"", 2008.

[RFC4005] Calhoun, P., Zorn, G., Spence, D., and D. Mitton, "Diameter Network Access Server Application", August 2005, <http://www.rfc-editor.org/rfc/rfc4005.txt>.

[RFC4072] Eronen, P., Hiller, T., and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", August 2005, <http://www.rfc-editor.org/rfc/rfc4072.txt>.

[TS29.228] 3rd Generation Partnership Project, "3GPP TS 29.228; Technical Specification Group Core Network and Terminals; IP Multimedia (IM) Subsystem Cx and Dx Interfaces; Signalling flows and message contents", <http://www.3gpp.org/ftp/Specs/html-info/29272.htm>.

[TS29.229] 3rd Generation Partnership Project, "3GPP TS 29.229; Technical Specification Group Core Network and Terminals; Cx and Dx interfaces based on the Diameter protocol; Protocol details", <http://www.3gpp.org/ftp/Specs/html-info/29229.htm>.

[TS29.328] 3rd Generation Partnership Project, "3GPP TS 29.328; Technical Specification Group Core Network and Terminals; IP Multimedia (IM) Subsystem Sh interface; signalling flows and message content", <http://www.3gpp.org/ftp/Specs/html-info/29328.htm>.

[TS29.329] 3rd Generation Partnership Project, "3GPP TS 29.329; Technical Specification Group Core Network and Terminals; Sh Interface based on the Diameter protocol; Protocol details", <http://www.3gpp.org/ftp/Specs/html-info/29329.htm>.

Authors' Addresses

Lionel Morand (editor)
Orange Labs

Phone: +33 1 4529 6257
Email: lionel.morand@orange.com

Victor Fajardo

Email: vf0213@gmail.com

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo 02600
Finland

Phone: +358 (50) 4871445
Email: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at