

Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended status: Informational
Expires: December 28, 2013

L. Morand, Ed.
Orange Labs
V. Fajardo

H. Tschofenig
Nokia Siemens Networks
June 26, 2013

Diameter Applications Design Guidelines
draft-ietf-dime-app-design-guide-19

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 Diameter. It is meant as a guidelines document and therefore as informative in nature.

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 December 28, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
2.	Terminology	3
3.	Overview	3
4.	Reusing Existing Diameter Applications	5
4.1.	Adding a New Command	5
4.2.	Deleting an Existing Command	6
4.3.	Reusing Existing Commands	6
4.3.1.	Adding AVPs to a Command	6
4.3.2.	Deleting AVPs from a Command	8
4.4.	Reusing Existing AVPs	9
4.4.1.	Setting of the AVP Flags	9
4.4.2.	Reuse of AVP of Type Enumerated	9
5.	Defining New Diameter Applications	9
5.1.	Introduction	9
5.2.	Defining New Commands	10
5.3.	Use of Application-Id in a Message	10
5.4.	Application-Specific Session State Machines	11
5.5.	Session-Id AVP and Session Management	11
5.6.	Use of Enumerated Type AVPs	12
5.7.	Application-Specific Message Routing	12
5.8.	Translation Agents	13
5.9.	End-to-End Application Capabilities Exchange	14
5.10.	Diameter Accounting Support	14
5.11.	Diameter Security Mechanisms	16
6.	Defining Generic Diameter Extensions	16
7.	Guidelines for Registrations of Diameter Values	17
8.	IANA Considerations	19
9.	Security Considerations	19
10.	Contributors	20
11.	Acknowledgments	20
12.	Informative References	20
	Authors' Addresses	22

[1.](#) Introduction

The Diameter base protocol provides facilities to extend Diameter (see [Section 1.3 of \[RFC6733\]](#)) to support new functionality. In the context of this document, extending Diameter means one of the following:

1. Addition of new functionality to an existing Diameter application without defining a new application.
2. Addition of new functionality to an existing Diameter application that requires the definition of a new application.
3. The definition of an entirely new Diameter application to offer functionality not supported 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 be done by any combination of reusing existing or defining new commands, AVPs or AVP values. However, application designers do not have complete freedom when making their design. A number of rules have been defined in [\[RFC6733\]](#) that place constraints on when an extension requires the allocation of a new Diameter application identifier or a new command code value. The objective of this document is the following:

- o Clarify the Diameter extensibility rules as defined in the Diameter base protocol.
- o Discuss design choices and provide guidelines when defining new applications.
- o Present trade-off choices.

2. Terminology

This document reuses the terminology defined in [\[RFC6733\]](#).

3. Overview

As designed, the Diameter base protocol [\[RFC6733\]](#) 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 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 [Section 9 of \[RFC6733\]](#)). Accordingly, a Diameter node is seen in this document as a single instance of a Diameter message delivery layer and one or more Diameter applications using it.

The Diameter base protocol is designed to be extensible and the principles are described in the [Section 1.3 of \[RFC6733\]](#). As a summary, Diameter can be extended by:

1. Defining new AVP values
2. Creating new AVPs
3. Creating new commands
4. Creating new applications

As a main guiding principle, the recommendation is: "try to re-use as much as possible!". It will reduce the time to finalize specification writing, and it will lead to a smaller implementation effort as well as reduce the need for testing. In general, it is clever to avoid duplicate effort when possible.

However, re-use is not appropriate when the existing functionality does not fit the new requirement and/or the re-use leads to ambiguity.

The impact on extending existing applications can be categorized into two groups:

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 for use in an existing command. Diameter implementations supporting the existing application but not the new AVP will simply ignore it, without consequences for the Diameter message handling. The standardization effort will be fairly small.

Major Extension: Enhancing an application that requires the definition of a new Diameter application.

Typical examples would be the creation of a new command for providing functionality not supported by existing applications or the definition of a new AVP with the M-bit set to be carried in an existing command. For such extension, a significant specification effort is required and a careful approach is recommended.

We would also like to remind that the definition of a new Diameter application and the definition of a new command should be something

to avoid as much as possible. In the past, there has been some reluctance to define new commands and new applications. With the modified extensibility rules provided by [[RFC6733](#)], registering new commands and new applications does not lead to additional overhead for the specification author in terms of standardization process. Registering new functionality (new commands, new AVPs, new applications, etc.) with IANA remains important to avoid namespace collisions, which will likely lead to deployment problems.

4. Reusing Existing Diameter Applications

An existing application may need to be enhanced to fulfill 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 impact.

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 Command Code Format (CCF) syntax 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 a 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 example considers the Diameter EAP application [[RFC4072](#)] and the Diameter NASREQ application [[RFC4005](#)]. When network access authentication using EAP is required, the Diameter EAP commands (Diameter-EAP-Request/Diameter-EAP-Answer) are used; otherwise the NASREQ application will be used. When the Diameter EAP application is used, the accounting exchanges defined in Diameter NASREQ may be used.

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:

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

Note: Importing commands too liberally could result in a monolithic and hard to manage application supporting too many different features.

4.2. Deleting an Existing Command

Although this process is not typical, removing a command from 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 (i.e., `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 that 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.

From a historical point of view, it is worth to note that there was a strong recommendation to re-use existing commands in the [[RFC3588](#)] to prevent rapid depletion of code values available for vendor-specific commands. However, [[RFC6733](#)] has relaxed the allocation policy and enlarged the range of available code values for vendor-specific applications. Although reuse of existing commands is still recommended, protocol designers can consider defining a new command when it provides a solution more suitable than the twisting of an existing command's use and applications.

4.3.1. Adding AVPs to a Command

Based on the rules in [[RFC6733](#)], AVPs that are added to an existing command can be categorized into:

- o Mandatory (to understand) AVPs. As defined in [[RFC6733](#)], these are AVPs with the M-bit flag set, which means that a Diameter node receiving them is required to understand not only their values but also their semantics. Failure to do so will cause a message handling error. This is regardless of whether these AVPs are required or optional as specified by the command's Command Code Format (CCF) syntax .
- o Optional (to understand) AVPs. As defined in [[RFC6733](#)], these are AVPs with the M-bit flag cleared. A Diameter node receiving these AVPs can simply ignore them if it does not support them.

The rules are strict in the case where the AVPs to be added are mandatory to understand, i.e., they have the M-bit set. A mandatory AVP cannot be added to an existing command without defining a new Diameter application, as stated in [[RFC6733](#)]. 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. Application designers should consider the following questions when deciding about the M-bit for a new AVP:

- o Would it be required for the receiving side to be able to process and understand the AVP and its content?
- o Would the new AVPs change the state machine of the application?
- o Would the presence of the new AVP lead to a different number of round-trips, effectively changing the state machine of the application?
- o 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?
- o Would the new AVP have duality in meaning, i.e., be used to carry application-related information as well as to indicate that the message is for a new application?

If the answer to at least one of the questions is "yes" then the M-bit has to be set for the new AVP. This list of questions is non-exhaustive and other criteria can be taken into account in the decision process.

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

- o 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.
- o An optional AVPs with dual purpose, i.e., to carry application data as well as to indicate support for one or more features. This has a tendency to introduce interpretation issues.
- o 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 AVP 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

The impacts of deleting an AVP from a command depends on its command code format specification and M-bit setting:

- o Deleting an AVP that is indicated as { AVP } in the command's CCF syntax specification (regardless of the M-bit setting).

In this case, a new command code and subsequently a new Diameter application have to be specified.

- o Deleting an AVP, which has the M-bit set, and 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.

- o Deleting an AVP, which has the M-bit cleared, and 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 Diameter base protocol, the characteristics of an AVP are tied to its role within an application and the commands.

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. Defining New Diameter Applications

5.1. Introduction

This section discusses the case where new applications have requirements that cannot be fulfilled 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, e.g., Cx/Dx ([[TS29.228](#)] and [[TS29.229](#)]), Sh ([[TS29.328](#)] and [[TS29.329](#)]) etc.

Application designers should try 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 5.10](#) should also be considered.

Additional considerations are described in the following sections.

5.2. Defining New Commands

As a general recommendation, commands should not be defined from scratch. It is instead recommended to re-use an existing command offering similar functionality and use it as a starting point.

Moreover, the new command's CCF syntax specification should be carefully defined when considering applicability and extensibility of the application. If most of the AVPs contained in the command are indicated as fixed or required, it might be difficult to reuse the same command and therefore the same application in a slightly changed environment. Defining a command with most of the AVPs indicated as optional must not be seen as a sub-optimal design introducing too much flexibility in the protocol. The protocol designers are only advised to clearly state the condition of presence of these AVPs and properly define the corresponding behaviour of the Diameter nodes when these AVPs are absent from the command.

Note: As a hint for protocol designers, it is not sufficient to just look at the command's CCF syntax specification. It is also necessary to carefully read through the accompanying text in the specification.

In the same way, the CCF syntax specification should be defined such that it will be possible to add any arbitrary optional AVPs with the M-bit cleared (including vendor-specific AVPs) without modifying the application. For this purpose, it is strongly recommended to add "[AVP]" in the command's CCF, which allows the addition of any arbitrary AVP as described in [\[RFC6733\]](#).

5.3. 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 Diameter base protocol, i.e., RAR/RAA, STR/STA, ASR/ASA and possibly ACR/ACA in the coupled accounting model, see [Section 5.10](#). Some existing specifications do not adhere to this rule for historical reasons. However, this guidance should be followed to avoid routing problems.

In general, when a new application has been allocated with a new Application Id and it also reuses existing commands with or without modifications, 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. The Vendor-Id AVP is an informational AVP only and kept for backward compatibility reasons.

5.4. Application-Specific Session State Machines

[Section 8 of \[RFC6733\]](#) provides session state machines for authentication, authorization and accounting (AAA) services and these session state machines are not intended to cover behavior outside of AAA. If a new application cannot clearly be categorized into any of these AAA services, it is recommended that the application defines its own session state machine. 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.[\[Q.3303.3\]](#)).

5.5. Session-Id AVP and Session Management

Diameter applications are usually designed with the aim of managing user sessions (e.g., Diameter network access session (NASREQ) application [\[RFC4005\]](#)) or specific service access session (e.g., Diameter SIP application [\[RFC4740\]](#)). In the Diameter base protocol, session state is referenced using the Session-Id AVP. All Diameter messages that use the same Session-Id will be bound to the same session. Diameter-based session management also implies that both Diameter client and server (and potentially proxy agents along the path) maintain session state information.

However, some applications may not need to rely on the Session-Id to identify and manage sessions because other information can be used instead to correlate Diameter messages. Indeed, the User-Name AVP or any other specific AVP can be present in every Diameter message and used therefore for message correlation. Some applications might not require the notion of Diameter session concept at all. For such applications, the Auth-Session-State AVP is usually set to NO_STATE_MAINTAINED in all Diameter messages and these applications are therefore designed as a set of stand-alone transactions. Even if an explicit access session termination is required, application-specific commands are defined and used instead of the Session-Termination-Request/Answer (STR/STA) or Abort-Session-Request/Answer (ASR/ASA) defined in the Diameter base protocol. In such a case, the Session-Id is not significant.

Based on these considerations, protocol designers should carefully appraise whether the application currently defined relies on it's own session management concept or whether the Session-Id defined in the

Diameter base protocol would be used for correlation of messages related to the same session. If not, the protocol designers could decide to define application commands without the Session-Id AVP. If any session management concept is supported by the application, the application documentation must clearly specify how the session is handled between client and server (as possibly Diameter agents in the path).

5.6. Use of Enumerated Type AVPs

The type Enumerated was initially defined to provide a list of valid values for an AVP with their respective interpretation described in the specification. For instance, AVPs of type Enumerated can be used to provide further information on the reason for the termination of a session or a specific action to perform upon the reception of the request.

However, AVPs of type Enumerated are too often used as a simple Boolean flag, indicating for instance a specific permission or capability, and therefore only two values are defined, e.g., TRUE/FALSE, AUTHORIZED/UNAUTHORIZED or SUPPORTED/UNSUPPORTED. This is a sub-optimal design since it limits the extensibility of the application: any new capability/permission would have to be supported by a new AVP or new Enumerated value of the already defined AVP, causing backwards compatibility issues with existing implementations.

Instead of using an Enumerated AVP for a Boolean flag, protocol designers are encouraged to use Unsigned32 or Unsigned64 AVP type as bit mask whose bit settings are described in the relevant Diameter application specification. Such AVPs can be reused and extended without major impact on the Diameter application. The bit mask should leave room for future additions. Examples of AVPs that use bit masks are the Session-Binding AVP defined in [[RFC6733](#)] and the MIP6-Feature-Vector AVP defined in [[RFC5447](#)].

5.7. Application-Specific Message Routing

Diameter request message routing usually relies on the Destination-Realm AVP and the Application Id present in the request message header. However, some applications may need to rely on the User-Name AVP or any other application-specific AVP present in the request to determine the final destination of a request, e.g., to find the target AAA server hosting the authorization information for a given user when multiple AAA servers are addressable in the realm.

In such a context, basic routing mechanisms described in [[RFC6733](#)] are not fully suitable, and additional application-level routing mechanisms have to be described in the application documentation to

provide such specific AVP-based routing. Such functionality will be basically hosted by an application-specific proxy agent that will be responsible for routing decisions based on the received specific AVPs.

Examples of such application-specific routing functions can be found in the Cx/Dx applications ([[TS29.228](#)] and [[TS29.229](#)]) of the 3GPP IP Multimedia Subsystem, in which the proxy agent (Subscriber Location Function aka SLF) uses specific application-level identities found in the request to determine the final destination of the message.

Whatever the criteria used to establish the routing path of the request, the routing of the answer has to follow the reverse path of the request, as described in [[RFC6733](#)], with the answer being sent to the source of the received request, using transaction states and hop-by-hop identifier matching. In particular, this ensures that the Diameter Relay or Proxy agents in the request routing path will be able to release the transaction state upon receipt of the corresponding answer, avoiding unnecessary failover. Application designers are strongly dissuaded from modifying the answer-routing principles described in [[RFC6733](#)] when defining a new application.

5.8. Translation Agents

As defined in [[RFC6733](#)], a translation agent is a device that provides interworking between Diameter and another protocol (e.g., RADIUS).

In the case of RADIUS, it was initially thought that defining the translation function would be straightforward by adopting few basic principles, e.g., by the use of a shared range of code values for RADIUS attributes and Diameter AVPs. Guidelines for implementing a RADIUS-Diameter translation agent were put into [RFC 4005](#) ([[RFC4005](#)]).

However, it was acknowledged that such translation mechanism was not so obvious and deeper protocol analysis was required to ensure efficient interworking between RADIUS and Diameter. Moreover, the interworking requirements depend on the functionalities provided by the Diameter application under specification, and a case-by-case analysis will be required.

Therefore, protocol designers cannot assume the availability of a "standard" Diameter-to-RADIUS gateways agent when planning to interoperate with the RADIUS infrastructure. They should specify the required translation mechanism along with the Diameter application, if needed. This recommendation applies for any kind of translation.

5.9. End-to-End Application Capabilities Exchange

New Diameter applications can rely on optional AVPs to exchange application-specific capabilities and features. These AVPs can be exchanged on an end-to-end basis at the application layer. Examples of this can be found with the MIP6-Feature-Vector AVP in [[RFC5447](#)] and the QoS-Capability AVP in [[RFC5777](#)].

The end-to-end capabilities AVPs formalize the addition of new optional functionality to existing applications by announcing support for it. Applications that do not understand these AVPs can discard them upon receipt. Receivers of these AVPs can discover the additional functionality supported by the end-point originating the request and behave accordingly when processing the request. Senders of these AVPs can safely assume the receiving end-point does not support any functionality carried by the AVP if it is not present in corresponding response. This is useful in cases where deployment choices are offered, and the generic design can be made available for a number of applications.

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.

It is also important to note that this end-to-end capabilities exchange relies on the use of optional AVPs is not meant as a generic mechanism to support extensibility of Diameter applications with arbitrary functionality. When the added features drastically change the Diameter application or when Diameter agents have to be upgraded to support the new features, a new application should be defined.

5.10. Diameter Accounting Support

Accounting can be treated as an auxiliary application that is used in support of other applications. In most cases, accounting support is required when defining new applications. This document provides two 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 for 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 end up in a central accounting server. A split accounting model is a good design choice when:

- * The application itself does not define its own 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 differentiate received accounting messages. Since the received accounting messages can be for any application and/or service, the accounting server has to have a method to 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 record 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 the 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 does not provide an accounting server that supports Diameter. Consequently, the application server has to be provisioned to use a different protocol to access the accounting server, e.g., via LDAP, SOAP etc. This case includes the support of older accounting systems that are not Diameter aware.
- * The system architecture or deployment requires that the accounting service for the specific application should be handled by the application itself.

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. Although it is not recommended, an application may define a new set of commands to carry application-specific accounting records.

5.11. Diameter Security Mechanisms

As specified in [[RFC6733](#)], the Diameter message exchange should be secured between neighboring Diameter peers using TLS/TCP or DTLS/SCTP. However, IPsec can also be deployed to secure communication between Diameter peers. When IPsec is used instead of TLS or DTLS, the following recommendations apply.

IPsec ESP [[RFC4301](#)] in transport mode with non-null encryption and authentication algorithms is used to provide per-packet authentication, integrity protection and confidentiality, and support the replay protection mechanisms of IPsec. IKEv2 [[RFC5996](#)] is recommended for performing mutual authentication and for establishing and maintaining security associations (SAs).

IKEv1 [[RFC2409](#)] was used with [RFC 3588](#) [[RFC3588](#)] and for easier migration from IKEv1 based implementations both RSA digital signatures and pre-shared keys should be supported in IKEv2. However, if IKEv1 is used, implementers should follow the guidelines given in [Section 13.1 of RFC 3588](#) [[RFC3588](#)].

6. Defining 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 the support for QoS AVPs (see [[RFC5777](#)]).

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

Backward Compatibility:

With the design of generic extensions an protocol designer has to consider with potential concerns about how existing applications deal with the new extension they do not understand. Designers

also have to make sure that new extensions do not break expected message delivery layer behavior.

Forward Compatibility:

Protocol designers need to make sure that their design will not introduce undue restrictions for future applications.

Trade-off 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, this ties the sending of extension data to the application's transmission of 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 trade-off 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 trade-offs based on the requirements of the extension.

In practice, generic extensions often use optional AVPs because they are simple and non-intrusive to the application that would carry them. 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](#)]).

7. Guidelines for Registrations of Diameter Values

As summarized in the [Section 3](#) of this document and further described in the [Section 1.3 of \[RFC6733\]](#), there are four main ways to extend Diameter. The process for defining new functionality slightly varies based on the different extensions. This section provides protocol

designers with some guidance regarding the definition of values for possible Diameter extensions and the necessary interaction with IANA to register the new functionality.

a. Defining new AVP values

The specifications defining AVPs and AVP values provide guidance for defining new values and the corresponding policy for adding these values. For example, the [RFC 5777](#) [[RFC5777](#)] defines the Treatment-Action AVP which contains a list of valid values corresponding to pre-defined actions (drop, shape, mark, permit). This set of values can be extended following the Specification Required policy defined in [[RFC5226](#)]. As a second example, the Diameter base specification [[RFC6733](#)] defines the Result-Code AVP that contains a 32-bit address space used to identity possible errors. According to the [Section 11.3.2 of \[RFC6733\]](#), new values can be assigned by IANA via an IETF Review process [[RFC5226](#)].

b. Creating new AVPs

Two different types of AVP Codes namespaces can be used to create a new AVPs:

- * IETF AVP Codes namespace;
- * Vendor-specific AVP Codes namespace.

In the latter case, a vendor needs to be first assigned by IANA with a private enterprise number, which can be used within the Vendor-Id field of the vendor-specific AVP. This enterprise number delimits a private namespace in which the vendor is responsible for vendor-specific AVP code value assignment. The absence of a Vendor-Id or a Vendor-Id value of zero (0) in the AVP header identifies standard AVPs from the IETF AVP Codes namespace managed by IANA. The allocation of code values from the IANA-managed namespace is conditioned by an Expert Review of the specification defining the AVPs or an IETF review if a block of AVPs needs to be assigned. Moreover, the remaining bits of the AVP Flags field of the AVP header can be also assigned via Standard Action if the creation of new AVP Flags is desired.

c. Creating new commands

Unlike the AVP Code namespace, the Command Code namespace is flat but the range of values is subdivided into three chunks with distinct IANA registration policies:

- * A range of standard Command Code values that can be allocated via IETF review;
- * A range of vendor-specific Command Code values that can be allocated on a First-Come/First-Served basis;
- * A range of values reserved only for experimental and testing purposes.

As for AVP Flags, the remaining bits of the Command Flags field of the Diameter header can also be assigned via a Standards Action to create new Command Flags if required.

d. Creating new applications

Similarly to the Command Code namespace, the Application-Id namespace is flat but divided into two distinct ranges:

- * A range of values reserved for standard Application-Ids allocated after Expert Review of the specification defining the standard application;
- * A range for values for vendor specific applications, allocated by IANA on a First-Come/First-Serve basis.

The IANA AAA parameters page can be found at <http://www.iana.org/assignments/aaa-parameters/aaa-parameters.xml> and the enterprise number IANA page is available at <http://www.iana.org/assignments/enterprise-numbers>. More details on the policies followed by IANA for namespace management (e.g. First-Come/First-Served, Expert Review, IETF Review, etc.) can be found in [RFC5226].

NOTE:

When the same functionality/extension is used by more than one vendor, it is recommended to define a standard extension. Moreover, the registration of vendor-specific extension is encouraged to avoid interoperability issues in the same network. With this aim, the registration policy of vendor-specific extension has been simplified with the publication of [RFC6733] and the namespace reserved for vendor-specific extensions is large enough to avoid exhaustion.

8. IANA Considerations

This document does not require actions by IANA.

9. Security Considerations

This document provides guidelines and considerations for extending Diameter and Diameter applications. Although such an extension may related to a security functionality, the document does not explicitly give guidance on enhancing Diameter with respect to security.

10. 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.

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

11. Acknowledgments

We greatly appreciate the insight provided by Diameter implementers who have highlighted the issues and concerns being addressed by this document. The authors would also like to thank Jean Mahoney and Ben Campbell for their invaluable detailed review and comments on this document.

12. 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.

[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.

[RFC2407] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", [RFC 2407](#), November 1998.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", [RFC 2409](#), November 1998.

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

[RFC4005] Calhoun, P., Zorn, G., Spence, D., and D. Mitton, "Diameter Network Access Server Application", [RFC 4005](#), August 2005.

[RFC4072] Eronen, P., Hiller, T., and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", [RFC 4072](#), August 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", [RFC 4301](#), December 2005.

[RFC4740] Garcia-Martin, M., Belinchon, M., Pallares-Lopez, M., Canales-Valenzuela, C., and K. Tammi, "Diameter Session Initiation Protocol (SIP) Application", [RFC 4740](#), November 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#), May 2008.

- [RFC5447] Korhonen, J., Bournelle, J., Tschofenig, H., Perkins, C., and K. Chowdhury, "Diameter Mobile IPv6: Support for Network Access Server to Diameter Server Interaction", [RFC 5447](#), February 2009.
- [RFC5777] Korhonen, J., Tschofenig, H., Arumaithurai, M., Jones, M., and A. Lior, "Traffic Classification and Quality of Service (QoS) Attributes for Diameter", [RFC 5777](#), February 2010.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", [RFC 5996](#), September 2010.
- [RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", [RFC 6733](#), October 2012.
- [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>>.

Lionel Morand (editor)
Orange Labs
38/40 rue du General Leclerc
Issy-Les-Moulineaux Cedex 9 92794
France

Phone: +33145296257
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>

