Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 5, 2020

M. Pala
CableLabs

June 3, 2020

Credentials Provisioning and Management via EAP (EAP-CREDS)
draft-pala-eap-creds-06

Abstract

With the increase number of devices, protocols, and applications that rely on strong credentials (e.g., digital certificates, keys, or tokens) for network access, the need for a standardized credentials provisioning and management framework is paramount. The 802.1x architecture allows for entities (e.g., devices, applications, etc.) to authenticate to the network by providing a communication channel where different methods can be used to exchange different types of credentials. However, the need for managing these credentials (i.e., provisioning and renewal) is still a hard problem to solve.

EAP-CREDS, if implemented in Managed Networks (e.g., Cable Modems), could enable our operators to offer a registration and credentials management service integrated in the home WiFi thus enabling visibility about registered devices. During initialization, EAP-CREDS also allows for MUD files or URLs to be transferred between the EAP Peer and the EAP Server, thus giving detailed visibility about devices when they are provisioned with credentials for accessing the networks. The possibility provided by EAP-CREDS can help to secure home or business networks by leveraging the synergies of the security teams from the network operators thanks to the extended knowledge of what and how is registered/authenticated.

This specifications define how to support the provisioning and management of authentication credentials that can be exploited in different environments (e.g., Wired, WiFi, cellular, etc.) to users and/or devices by using EAP together with standard provisioning protocols.

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 https://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Requirements notation

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].

2. Introduction

Many environments are, today, moving towards requiring strong authentication when it comes to gain access to networks. The 802.1x architecture provides network administrators with the possibility to check credentials presented by a device even before providing any connectivity or IP services to it.

However, the provisioning and management of these credentials is a hard problem to solve and many vendors opt for long-lived credentials that can not be easily revoked, replaced, or simply renewed.

This specification addresses the problem of providing a simple-to-use and simple-to-deploy conduit for credentials management by extending the EAP protocol to support credentials provisioning and management functionality. In particular, the EAP-CREDS method defined here provides a generic framework that can carry the messages for provisioning different types of credentials. EAP-CREDS cannot be used as a stand-alone method, it is required that EAP-CREDS is used as an inner method of EAP-TLS, EAP-TEAP, or any other tunnelling method that can provide the required secrecy and (at minimum) server-side authentication to make sure that the communication is protected and with the right server.

2.1. Overview of existing solutions

Currently there are many protocols that address credentials lifecycle management. In particular, when it comes to digital certificates, some of the most deployed management protocols are: Certificate Management Protocol (CMP) [RFC4210], Certificate Management over CMS (CMC) [RFC5272][RFC6402], Enrollment over Secure Transport (EST) [RFC7030], and Automated Certificate Management Environment (ACME) . However, none of these protocols provide native support for client

that do not have IP connectivity yet (e.g., because they do not have network-access credentials, yet). EAP-CREDS provides the possibility to use such protocols (i.e., message-based) by defining a series of messages that can be used to encapsulate the provisioning messages for the selected provisioning protocol.

In addition to these protocols, EAP-CREDS also defines a series of simple messages that provide a generic enrollment protocol that allows not only certificates but also other types of credentials (e.g., username/password pairs, tokens, or symmetric secrets) to be delivered to the client as part of the provisioning and/or renewal process. The set of messages that make up the generic provisioning protocol is referred to as the Simple Provisioning Protocol protocol or SPP.

2.2. Scope Statement

This document focuses on the definition of the EAP-CREDS method to convey credentials provisioning and managing messages between the client and the AAA server. Moreover, the document defines how to encode messages for the main IETF provisioning protocols.

This document, however, does not provide specifications for how and where the credentials are generated. In particular, the credentials could be generated directly within the AAA server or at a different location (i.e., the Certificate Service Provider or CSP) site. Different authentication mechanisms (e.g., TLS, etc.) can be used to secure the communication between the server's endpoint and the CSP.

2.3. EAP-CREDS as tunneled mechanism only

EAP-CREDS requires that an outer mechanism is in place between the Peer and the Server in order to provide authentication and confidentiality of the messages exchanged via EAP-CREDS. In other words, EAP-CREDS assumes that an appropriatly encrypted and authenticated channel has been established to prevent the possibility to leak information or to allow man-in-the-middle attacks.

This choice was taken to simplify the message flow between Peer and Server, and to abstract EAP-CREDS from the secure-channel establishment mechanism. EAP-TLS, or EAP-TEAP are examples of such mechanisms.s

2.4. Fragmentation Support

EAP does not directly support handling fragmented packets and it requires the outer method to provide fragmentation support.

Because of the outer method requirements in particular, removing any support for fragmented messages in EAP-CREDS removes the duplication of packets (e.g., Acknowledgment Packets) sent across the Peer and the Server, thus resulting in a smaller number of exchanged messages

2.5. Encapsulating Provisioning Protocols in EAP-CREDS

In order to use EAP-CREDS together with your favorite provisioning protocol, the messages from the provisioning protcol need to be sent to the other party. In EAP-CREDS, this is done by encoding the provisioning protocol messages inside the ('Provisioning-Data') TLV. In case the provisioning protocol uses additional data for its operations (e.g., uses HTTP Headers), this data can be encoded in a separate ('Provisioning-Headers') TLV.

Since the implementation of the provisioning endpoint could happen in a (logically or physically) different component, a method is needed to identify when a provisioning protocol has actually ended. In EAP-CREDS, the 'D' bit in the message headers is used for this purpose.

In the first message of Phase Two, the Server provides the client with all the selected parameters for one specific credential that needs attention (or for a new credential) to be managed by the network. In particular, the server provides, at minimum, the ('Protocol') TLV, the ('Action') TLV, and the ('Provisioning-Params') or the ('Credentials-Info') TLV.

After checking the parameters sent by the Server, if the Peer does not support any of the proposed ones, it MUST send a message with one single ('Error') TLV with the appropriate error code(s). The server, can then decide if to manage a different set of credentials (if more where reported by the Peer in its Phase One message) or if to terminate the EAP session with an error.

The Peer and the Server exchange Provisioning messages until an error is detected (and the appropriate error message is sent to the other party) or until Phase Two is successfully completed.

2.6. Algorithm Requirements

EAP-CREDS uses the SHA-256 hashing algorithm to verify credentials in phase three of the protocol. Peers and Servers MUST support SHA-256 for this purpose.

2.7. Notation

In this document we use the following notation in the diagrams to provide information about the cardinality of the data structures (TLVs) within EAP-CREDS messages:

Symbol	Example	Usage
{ }	{TLV1}	Curly Brackets are used to indicate a set
[ ]	{[TLV2]}	Square Brackets are used to indicate that a field is optional
( )	{TLV1(=V)}	Round Squares are used to specify a value
+	{TLV_2+}	The Plus character indicates that one or more instances are allowed

Table 1: EAP-CREDS Notation

3. EAP-CREDS Protocol

In a nutshell, EAP-CREDS provides the abstraction layer on top of which credentials provisioning/managing protocols can be deployed thus enabling their use even before provisioning IP services.

This section outlines the operation of the protocol and message flows. The format of the CREDS messages is given in Section 4.

3.1. Message Flow

EAP-CREDS message flow is logically subdivided into three different phases: Initialization, Provisioning, and Validation. EAP-CREDS enforces the order of phases, i.e. it is not possible to move to an earlier phase.

Phase transitioning is controlled by the Server. In particular, the server, after the last message of a phase, it can decide to either (a) start the next phase by sending the first message of the next phase, or (b) continue the same phase by sending another "first" message of the phase (e.g., managing a second set of credentials) - this is allowed only in Phase Two and Phase Three but NOT in Phase One, or (c) terminate the EAP session.

Phase One (Required). Initialization. During this phase the Peer and the Server exchange the information needed to select the appropriate credentials management protocol. Phase One flow is composed by only messages. In particular, the Sever sends its initial message of type ('EAP-CREDS-Init'). The Peer replies with

the details about which provisioning protocols are supported, and additional information such as the list of installed credentials and, optionally, authorization data (for new credentials registration).

Phase Two (Optional). Provisioning Protocol Flow. In this phase, the Peer and the Server exchange the provisioning protocol's messages encapsulated in a EAP-CREDS message of type Provisioning. The messages use two main TLVs. The first one is the ('Provisioning-Headers') TLV which is optional and carries information that might be normally coveyed via the transport protocol (e.g., HTTP headers). The second one is the ('Provisioning-Data'), which is required and carries the provisioning protocol's messages. The server can decide to repeat phase two again to register new credentials or to renew a separate set of credentials by issuing a new ('Provisioning') message for the new target. When no more credentials have to be managed, the Server can start phase three or simply terminate the EAP session.

Phase Three (Optional). Credentials Validation. This optional phase can be initiated by the server and it is used to validate that the Peer has properly installed the credentials and can use them to authenticate itself. Depending on the credentials' type, the messages can carry a challenge/nonce, the value of the secret/token, or other information. The format of the credentials is supposed to be known by the provider and the device.

3.2. Phase Transitioning Rules

In order to keep track of starting and ending a phase, EAP-CREDS defines several bits and fields in the EAP-CREDS message headers. In particular, as described in Section 4.1, the 'S' bit is used to indicate the beginning (or Start) of a phase, while the 'Phase' field (4 bits) is used to indicate the phase for this message.

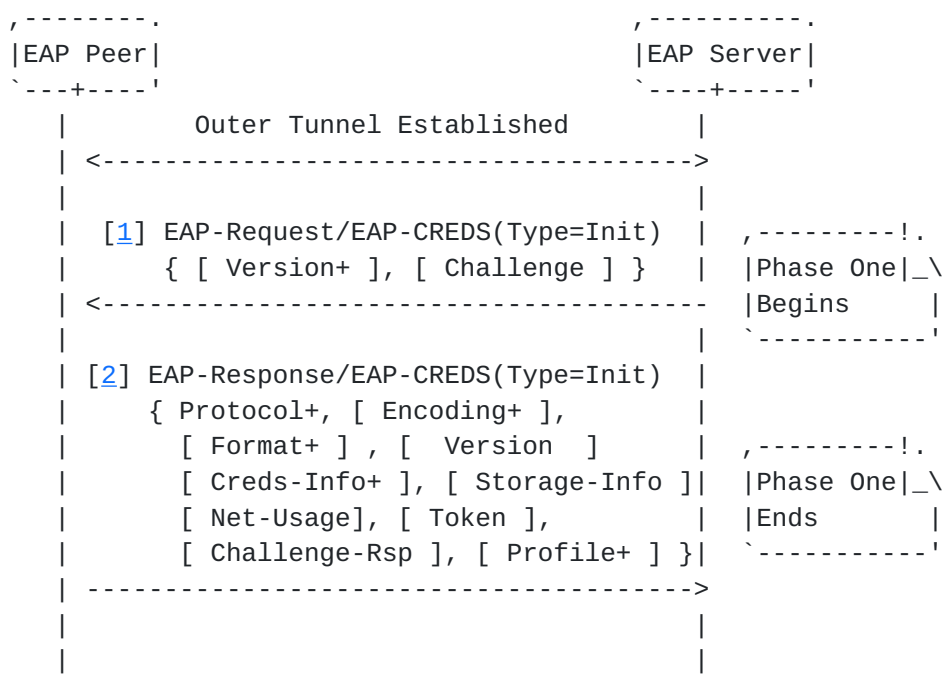
In EAP-CREDS, phase transitioning is under the sole control of the Server, therefore the value of the 'S' bit is meaningful only in messages sent by the Server. The value of the 'S' bit in Peer's messages SHALL be set to '0x0' and SHALL be ignored by the server.

When starting a new phase, the Server MUST set the 'S' bit to '1' and the 'Phase' field to the current phase number (e.g., one, two, or three).

In case the first message of a phase is to be repeated (e.g., because of processing multiple credentials), the 'S' bit SHALL be set to '0' (i.e., it should be set to '1' only on the first occurrency and set to '0' in subsequent messages).

3.3. Phase One: Initialization

The following figure provides the message flow for Phase One:



EAP-CREDS Phase One Message Flow

[1] Server sends EAP-Request/EAP-CREDS(Type=Init):

After the establishment of the outer mechanism (e.g., EAP-TLS, EAP-TEAP, EAP-TTLS, etc.), the server MAY decide to start a credentials management session. In order to do that, the Server sends an EAP-Request/EAP-CREDS(Type=Init) message to the Peer with one ('Phase-Control') TLV with the 'S' bit set to '1' and the value set to '1' (thus indicating the beginning of Phase One). Also, the Server MAY use one or more ('Version') TLVs to indicate the supported versions.

The Server MAY also specify which versions of EAP-CREDS are supported by adding one or more ('Version') TLVs. If no ('Version') TLV is added to the message, the Peer SHOULD assume the supported version is 1 ('0x1').

[2] The Peer sends EAP-Response/EAP-CREDS(Type=Init)

The Peer, sends back a message that carries one ('Version') TLV to indicate the selected version of EAP-CREDS (i.e. from the list

provided by the server) (optional). If the client does not include the ('Version') TLV, the Server MUST use the most recent supported version of EAP-CREDS. Moreover, the Server includes one or more ('Protocol') TLVs to indicate the list of supported provisioning protocols, followed by one ('Credentials-Info') TLVs for each installed credentials to provide their status to the server (i.e., if multiple credentials are configured on the Peer for this Network, then the Peer MUST include one ('Credentials-Info') TLV for each of them).

The Peer also provides the list of supported Encodings and Formats by adding one or more ('Supported-Encodings') and ('Supported-Formats') TLVs. The Peer MAY also provide the Server with information about the Peer's credentials storage by using the 'Storage-Status' TLV.

When there are no abailable credentials, the Peer MAY include an authorization token that can be consumed by the Server for registering new credentials. In particular, the Peer can include the ('Token-Data') TLV to convey the value of the token. The ('Challenge-Data') and ('Challenge-Response') TLVs, instead, can be used to convey a challenge and its response based on the authorization information (e.g., maybe a public key hash is present in the Token, then the peer can generate some random data - or use the one from the Server - and generate a signature on that value: the signature SHALL be encoded in the ('Challenge-Response') TLV and it should be calculated over the concatenation of values inside the ('Challenge-Data') TLV and the ('Token-Data') TLV.

Also, the Peer MAY add one or more ('Profile') TLVs to indicate to the Server which profiles are requested/supported (e.g., a pre-configuration MAY exist on the Peer with these ecosystem-specific identifiers).

Ultimately, the Peer MAY include additional metadata regarding the status of the Peer. To this end, the Peer can use a ('Storage-Info') TLV to provide the server with additional data about the Peer's capabilities and resources. Also, the ('Network-Usage') TLV can be used to provide the Server with the indication of which network resources are needed by the Peer and what is its intended utilization pattern(s).

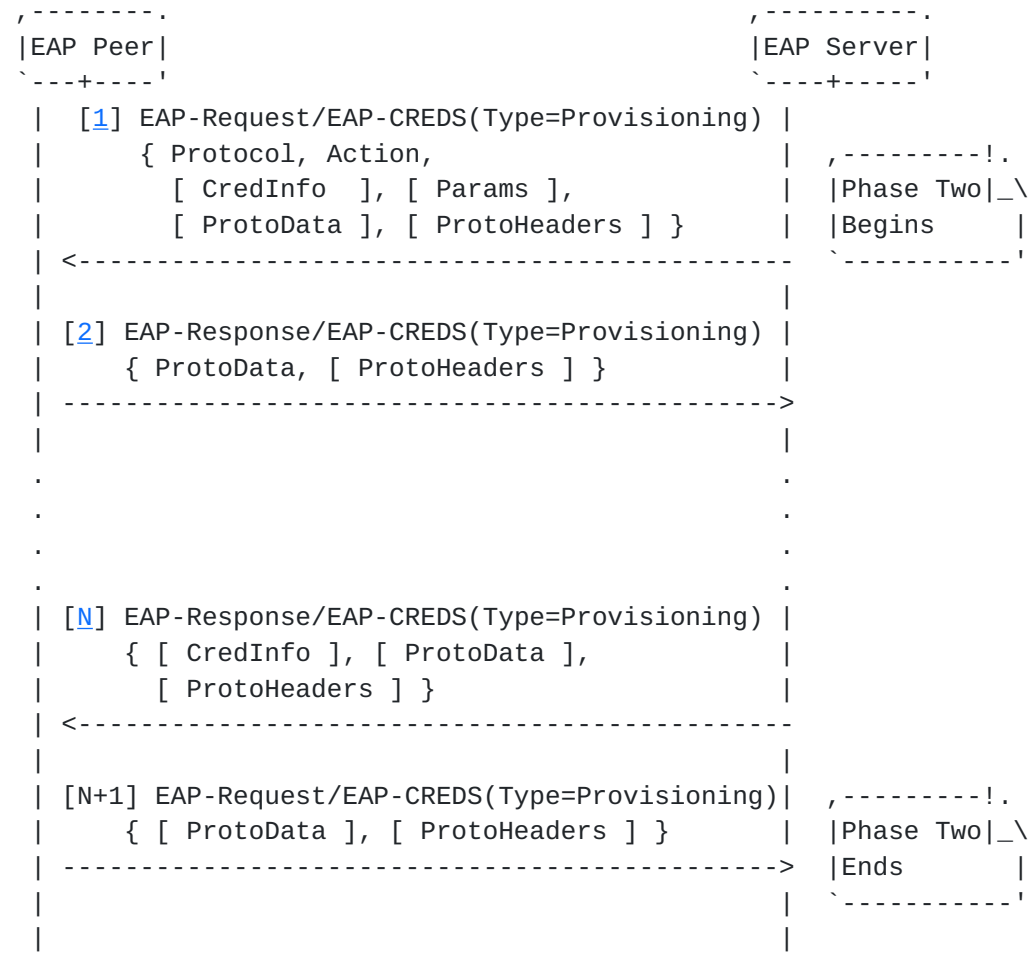
The server checks that the Peer's selected protocol, version, and parameters are supported and, if not (or if the server detects an error), it can (a) send a non-recoverable error message to the peer, notify the outer (tunneling) layer, and terminate the EAP-CREDS session, or (b) start phase one again by sending a new

('EAP-CREDS-Init') message that will also carry an ERROR TLV that provides the Peer with the reason the initial response was not acceptable. In this case, the ('Phase-Control') TLV MUST be omitted since it is not the first message of phase one. The server and the peer can repeat phase one until they reach an agreement or the session is terminated by the Server.

NOTE WELL: The determination of the need to start phase two or not is based on the contents of the ('Credentials-Info') TLV sent by the Peer (e.g., a credential is about to expire or a credential is simply missing).

3.4. Phase Two: Provisioning

The following figure provides the message flow for Phase 2:



EAP-CREDS Phase Two Message Flow

[1] The Server sends EAP-Request/EAP-CREDS(Type=Init)

The first message of Phase Two indicates that the Server is ready to initiate the selected provisioning protocol.

[2] The Peer sends EAP-Response/EAP-CREDS(Type=Init)

After that, the Peer sends its first message to the Server by sending the EAP-Response/EAP-CREDS(Type=Provisioning) message. This message contains the selected provisioning protocol's message data and some extra fields (e.g., transport-protocol headers) in the ('Provisioning-Data') and ('Protocol-Headers') TLVs respectively.

[3] The Server sends EAP-Request/EAP-CREDS(Type=Init)

The Server replies to the Peer's message with EAP-Request/EAP-CREDS(Type=Provisioning) messages until the provisioning protocol reaches an end or an error condition arise (non-recoverable).

[N] The Server sends EAP-Request/EAP-CREDS(Type=Provisioning)

When the provisioning protocol has been executed for the specific set of credentials, the server sends a last message that MUST include the description of the provisioned credentials in a ('Credentials-Info') TLV and MUST set the 'D' bit in the EAP-CREDS message header to '1' to indicates that the server does not have any more ('Provisioning') messages for this credenital. The final message does not need to be an empty one, i.e. other TLVs are still allowed in the same message (e.g., the 'Provisioning-Data' and the 'Provisioning-Headers' ones).

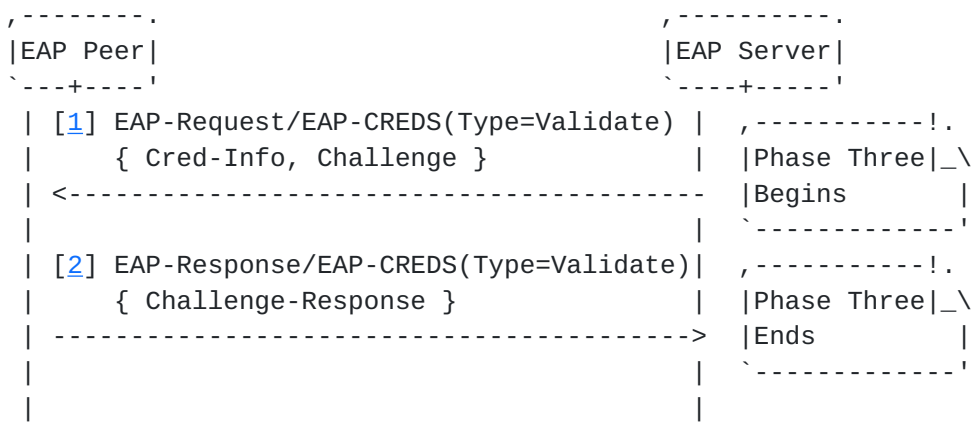
[N+1] The Peer sends EAP-Request/EAP-CREDS(Type=Provisioning)

The Peer MUST reply to the server with a ('Provisioning') message that MUST have the 'D' bit in the EAP-CREDS message header set to '1', thus indicating that the credentials have been installed correctly. In case of errors, the Peer MUST include the appropriate ('Error') TLV. Also in this case, the final message does not need to be an empty one, i.e. other TLVs are still allowed in the same message (e.g., the 'Provisioning-Data' and the 'Provisioning-Headers' ones).

At this point, the Server can decide to provision (or manage) another set of credentials by issuing a new ('Provisioning') message, or it can decide to start Phase Three by sending its first ('Validate') message, or it can terminate the EAP session.

3.5. Phase Three: Validation

The following figure provides the message flow for Phase 3:



EAP-CREDS Phase Three Message Flow

Phase three is optional and it is used by the server to request the client to validate (proof) that the new credentials have been installed correctly before issuing the final Success message.

NOTE WELL: Phase Three introduces a dependency on the selected hashing algorithm to provide common and easy way to check the integrity and functionality of a newly installed set of credentials.

[1] The Server sends EAP-Request/EAP-CREDS(Type=Validate)

In order to start Phase Three, the Server sends an EAP-Request/ EAP-CREDS(Type=Validate) message to the Peer. The Server MUST include the ('Credentials-Info') TLV to provide the indication about which set of credentials the Server intends to validate. The Server MUST also include a randomly generated challenge in the message to the client. The type of challenge determines how the ('Challenge-Response') is calculated. EAP-CREDS defines the asymmetric and symmetric challenges in Section 8.6 and others can be defined according to the specified rules.

As usual, the Server MUST set, in the headers, the 'S' bit to '1' in its first message of Phase Three and the 'Phase' value shall be set to '3' (beginning of Phase Three).

[2] The Peer sends EAP-Response/EAP-CREDS(Type=Validate)

When the client receives the Validate message from the server, it calculates the response to the challenge and sends the response back to the server in a EAP-Response/EAP-CREDS(Type=Validate) message. When the EAP-CREDS-ASYMMETRIC-CHALLENGE and EAP-CREDS-SYMMETRIC-CHALLENGE values are used in the Challenge type, the Peer MUST calculate the response as follows:

Public-Key

For any public-key based credentials (e.g., certificates or raw key pairs), the response to the challenge is calculated by generating a signature over the hashed value of the challenge. The hashing algorithm to be used for this purpose is specified in Section 2.6. The format of the signature in the ('Challenge-Response') TLV is the concatenation of:

- The signatureAlgorithm (DER encoded) which contains the identifier for the cryptographic algorithm used by the Peer to generate the signature. [RFC3279], [RFC4055], and [RFC4491] list supported signature algorithms, but other signature algorithms MAY also be supported. The definition of the signatureAlgorithm is provided in Section 4.1.1.2 of [RFC5280].

- The signatureValue (DER encoded) which contains the digital signature itself. The signature value is encoded as a BIT STRING and the details of how to generate the signatures' structures can be found in Section 4.1.1.3 of [RFC5280] and referenced material.

Symmetric Secret

For any symmetric based credentials (e.g., password or Key), the response to the challenge is calculated by using the selected hash function (see Section 2.6) on the concatenation of (a) the value carried in the server-provided ('Challenge-Data') TLV, and (b) the secret value itself (salted hash).

The initial values for the type of challenges are described in the Section 8.6. Other types of challenges MAY be defined according to the specified procedures.

In case of issues with the validation of newly deployed credentials, both the Server and the Peer should consider those credentials invalid (or unusable) and should issue the required failure message(s).

4. EAP-CREDS Message Format

The EAP-CREDS defines the following message types:

1. EAP-CREDS/Init

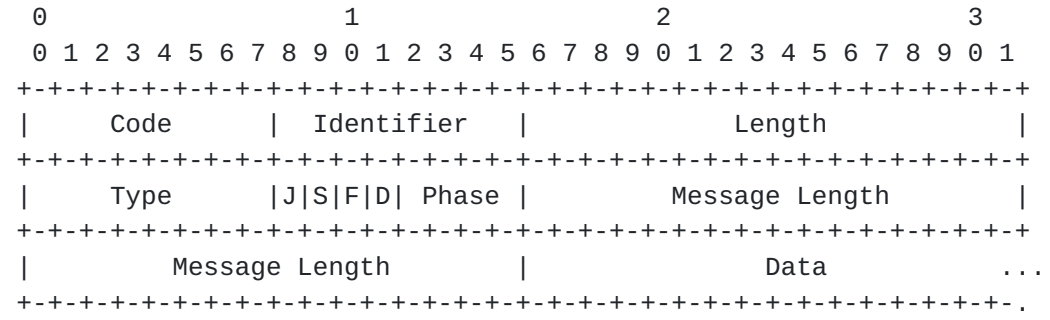
2. EAP-CREDS/Provisioning

3. EAP-CREDS/Validate

Each of these message types have the basic structure as identified in Section 4.1. EAP-CREDS messages contain zero, one, or more TLVs. The internal structure of the different types of TLVs is described in Section 4.2, while a detailed description of the EAP-CREDS message types is provided in Section 5.

4.1. Message Header

The EAP-CREDS messages consist of the standard EAP header (see Section 4 of [RFC3748]), followed by the version of the EAP-CREDS (4 bits) and a field (4 bits) reserved for future use. The header has the following structure:



Where the Code, Identifier, Length, and Type fields are all part of the EAP header as defined in [RFC3748]. Since EAP-CREDS can only be used as a tunneled mechanism, the presence of these fields is only for backward compatibility with existing parsers. In particular, the 'Length' field is not used (can be ignored): the message length is carried in the 'Message Length' field instead.

The Type field in the EAP header is <TBD> for EAP-CREDS.

The Flags bitfield is used to convey status information (e.g., extra long message, phase number, phase transitioning state). The transition-control bit (i.e., the 'S' bit) are set in Server's messages and are ignored in Peer's messages (the Server is the entity

that unilaterally controls the phase transition process). The meanins of the bits in the 'Flags' field are as follows:

Bit 'J' (Jumbo Message) - If set, it idicates the presence of the 'Message Length' field. This bit SHALL be used only when the size of the message exceeds the maximum value allowed in the 'Length' field. In this case, the 'Message Length' field is added to the message and set to the whole message size and the 'Length' field is used for the current fragment length. If not set, the 'Message Length' field is not present in the Message and the 'Length' field is used for the message size (and the 'F' bit MUST be set to '0').

Bit 'S' (Start) - If set, this message is the first one of a new EAP-CREDS phase. The value of the new phase is encoded in the 'Phase' field.

Bit 'F' - If set, this message is a fragment of a message. In this case, the 'Data' field is to be concatenated with all messages with the 'F' bit set to '1' until the message with the 'F' bit set to '0' that indicates the end of the message. If the message is not fragmented, the 'F' bit MUST be set to '0'. The use of this bit is required when the tunneling method does not provide support for messages up to 2^32 bits in size.

Bit 'D' - This bit is used in Phase Two and Phase Three to indicate that the specific operation for the identified credential is over. For example, when multiple credentials exist on the Peer and the Server needs to manage and validate one of them. In its last message, when the provisioning protocol is done, the server sets the 'D' (Done) bit to indicate that it is done. The Peer, in its reply, sets the bit to indicate the end of provisioning for this credentials is also over. After that, the Server can continue Phase Two, transition to Phase Three, or terminate the EAP session.

The Phase field is a 4-bits value and identifies the EAP-CREDS phase for the current message. The version of EAP-CREDS described in this document supports three values for this field:

0x01 - Phase One

0x02 - Phase Two

0x03 - Phase Three

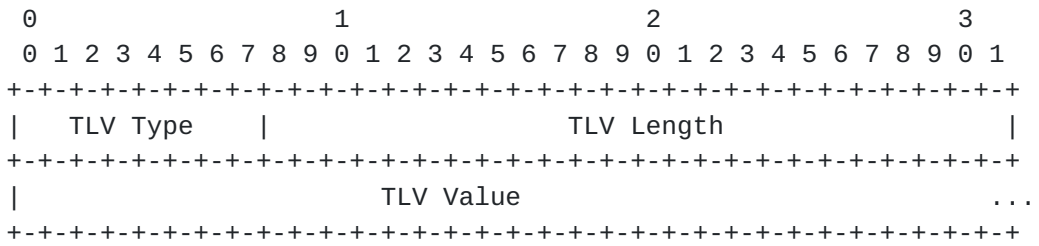
A detailed explanation of the 'Phase' and 'Flags' fields of the message headers is provided in Section 3.2.

The Data field is the message payload. The full description of this field is provided in the next section.

4.2. Message Payload

The Data part of the message is organized as zero, one, or more TLV objects whose structure is defined in this section.

Each TLV object has the same basic structure that is defined as follows:



Where:

TLV-Type (uint8)

This field is used to indicate the type of data that the TLV carries. The type of TLV determines its internal structure. The supported values for this fields are provided in the following table:

Length (uint24)

This field carries the size of the value of the TLV. In particular, the overall size of a TLV (i.e., the header plus the value) can be calculated by adding the size of the header (6 octects) to the value of the Length field (i.e., the size of the TLV's value).

TLV Name	TLV Type	Scope/Usage
<TBD>	Action TLV	Phase Two
<TBD>	Certificate-Data TLV	Phase Two/SPP
<TBD>	Challenge-Data TLV	Phase Two, Phase Three
<TBD>	Challenge-Response TLV	Phase Two, Phase Three
<TBD>	Credentials-Data TLV	Phase Two/SPP
<TBD>	Credentials-Info TLV	Phase Two, Phase Three
<TBD>	Error TLV	All Phases
<TBD>	Network-Usage TLV	Phase One
<TBD>	Profile TLV	Phase Two
<TBD>	Protocol TLV	Phase One, Phase Two
<TBD>	Provisioning-Data TLV	Phase Two
<TBD>	Provisioning-Headers TLV	Phase Two
<TBD>	Provisioning-Params TLV	Phase Two
<TBD>	Certificate-Request TLV	SPP
<TBD>	Storage-Info TLV	SPP
<TBD>	Supported-Format TLV	SPP
<TBD>	Supported-Encoding TLV	SPP
<TBD>	Token-Data TLV	Phase One
<TBD>	Version TLV	Phase One

Table 2: EAP-CREDS Supported TLVs Types

TLV Value (> 1 octet)

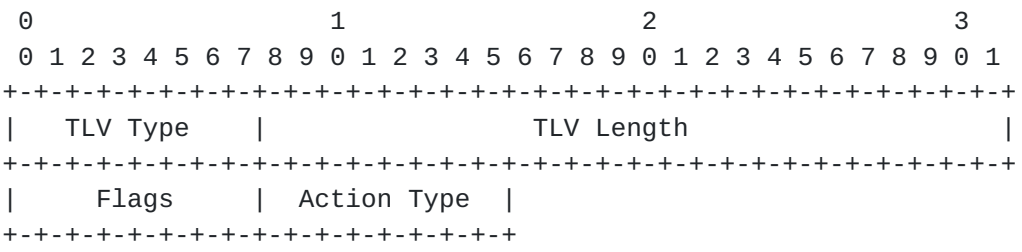
This field carries data for the identified TLV. The internal structure is determined by the TLV Type field.

The rest of this section describes the structure of the different supported TLVs and their usage in the different messages.

4.3. EAP-CREDS defined TLVs

EAP-CREDS messages's payload comprieses zero, one, or more TLVs that are encoded in a single EAP-CREDS message. The values for the TLV Type that are supported by this specifications are listed in Table 2.

4.3.1. The Action TLV



TLV Type (uint8)

<TBD> - Action TLV

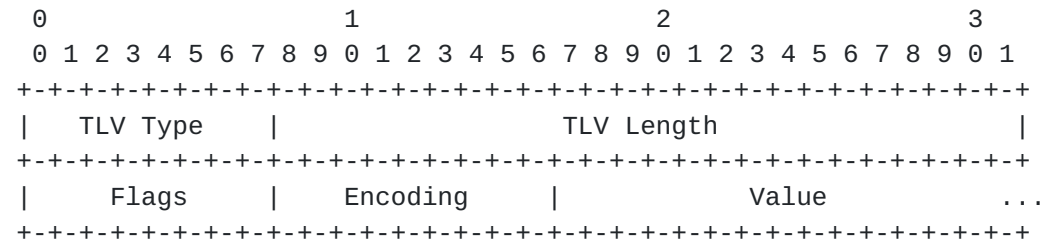
TLV Length (uint24)

Fixed value (=2)

Flags (uint8)

Reserved

4.3.2. The Certificate-Data TLV



TLV Type (uint8)

<TBD> - Certificate-Data TLV

Length (uint24)

Provides the length of the TLV (> 3 octets)

Flags (uint8)

Provides a BITMASK that can be used to provide additional information related to the encapsulated certificate. The bits have the following meaning:

Bit 0 - If set, the certificate is trusted (Trust Anchor)

Bit 1 - If set, the certificate is a CA certificate

Bit 2 - If set, the certificate is self-signed

Bit 3 - If set, the certificate is a proxy certificate

Bit 4 - If set, the certificate is an attribute certificate

Bit 5 - Reserved

Bit 6 - Reserved

Bit 7 - Reserved

For a Trusted Root CA, the value of the flags shall be 0x7 (0000 0111). For an intermediate CA certificate that is not implicitly trusted, the value of the flags field should be set to 0x02 (0000 0010). For an End-Entity certificate, the value of the Flags will be 0x0 (0000 0000).

Format (uint8)

Provides the indication of the Format the certificate is in. The allowed values for this field are listed in Section 8.5.

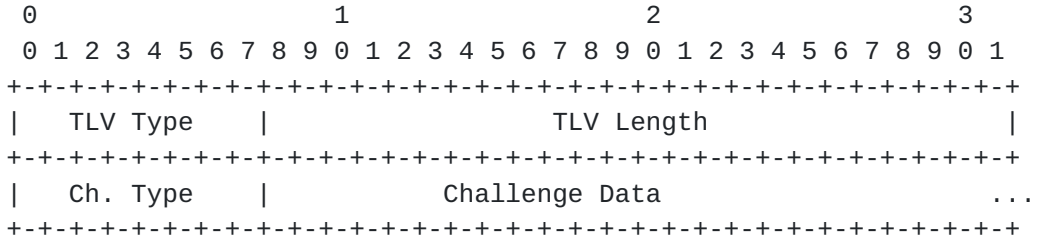
Encoding (uint8)

Provides the indication of the Encoding the certificate is in. The allowed values for this field are listed in Section 8.8.

Value (octet string)

This field carries the data for the certificate.

4.3.3. The Challenge-Data TLV



TLV Type (uint8)

<TBD> - Challenge-Data TLV

Length (uint24)

3 octets

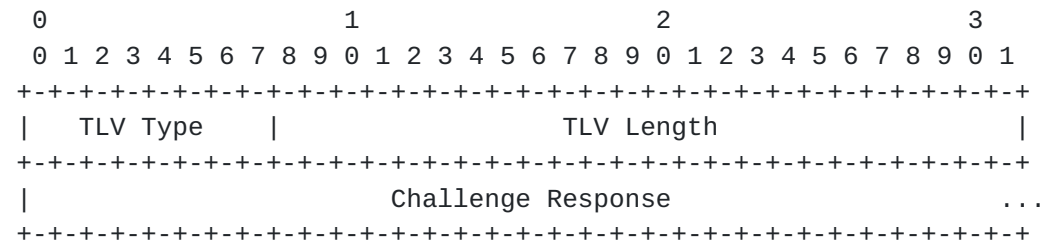
Challenge Type (uint8)

This field carries the type of Challenge. In particular, the challenge type determines how the Peer MUST calculate the ('Challenge-Response'). The initial values for this fiel are listed in Section 8.6. Please refer to Section 3.5 for a detailed explanation of how to calculate the response to the challenge for the challenge types defined in this document.

Challenge Data (> 1 octet)

This field carries the data to be used as a challenge when validating newly deployed credentials.

4.3.4. The Challenge-Response TLV



TLV Type (uint8)

<TBD> - Challenge-Response TLV

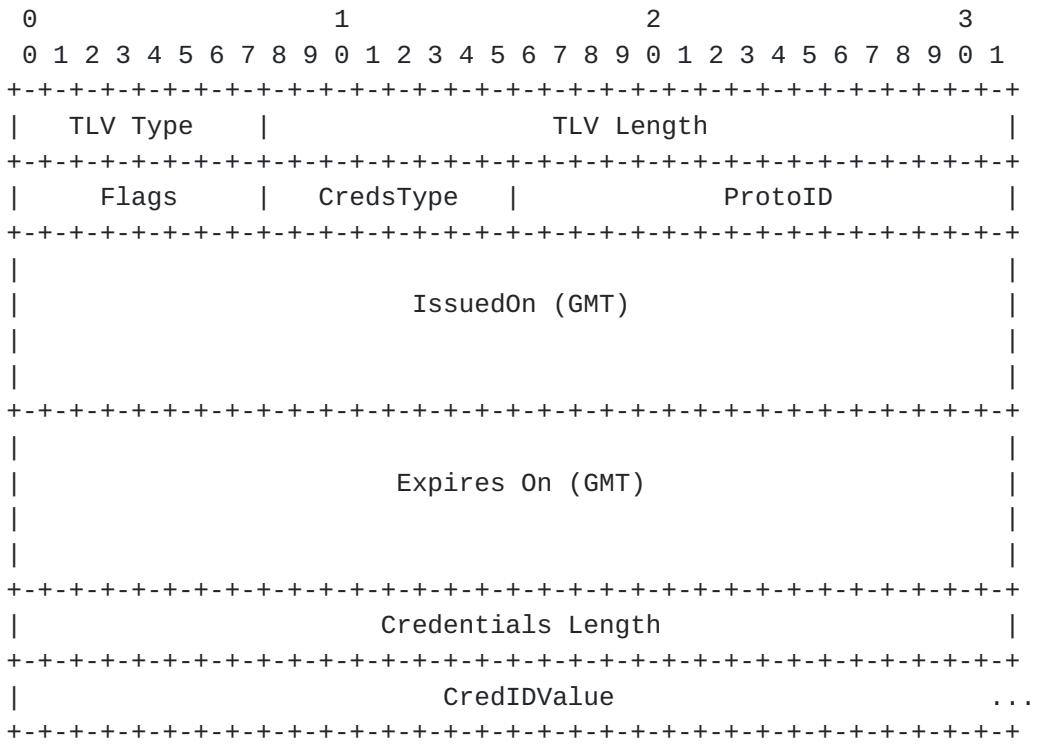
Length (uint24)

3 octets

Challenge Response (> 1 octet)

This field carries the data that resulted from the use of the credentials to be validated.

4.3.5. The Credentials-Information TLV



The Credential-Information TLV is used by the Peer to provide a description of the installed credentials that are relevant for the network that is being accessed.

For example, when a set of credentials need to be renewed, the server checks the ('Credentials-Info') from the Peer and eventually selects the right one for renewal. The TLV structure is as follows:

TLV Type (uint8)

<TBD> - Credentials-Information TLV

Length (uint24)

Provides the total length of the body of the Credential-Information TLV.

Flags (uint8)

Provides a BITMASK that can be used to provide information about the status of the credentials (e.g., if the use marks the

credentials to be compromised). The bits have the following meaning:

Bit 0 - If set, the credential is marked as compromised

Bit 1 - If set, the credential is immutable and cannot be updated

Bit 2 - Private Key or Secret Immutable, the public part of the credential (e.g., a certificate) can still be updated

Bit 3 - If set, the credential cannot be updated (both public and private parts)

Bit 4 - If set, the credential is ready to be used

Bit 5 - If set, the credential was generated on the server

Bit 6 - If set, the Peer would like to update the credential even if they are not expired

Bit 7 - Reserved

CredType (uint8)

This field provides the description of the type of credential. The type of credentials are listed in Section 8.3

ProtoID (uint16)

This field indicates the protocol that was used to retrieve the target credential. When the TLV is used in a Request by the Server, this field is ignored. The values for this field are listed in Section 8.1.

IssuedOn (16 octets)

This field carries the GMT date for when this credential was issued. This field is 16 bytes long (the last byte must be set to '0x00') and contains the NULL-terminated ASCII string that represents the timestamp where the credential was issued. When the value is not set, the field should be set to { 0x00 }. The format of the string is as follows:

YYYYMMDDHHmmssZ

Where:

YYYY - is the 4 digits representation of the year

MM - is the 2 digits representation of the month

DD - is the 2 digits representation of the day of the month

HH - is the 2 digits representation of the hour of the day (24 hour format)

mm - is the 2 digits representation of the minutes of the hour

ss - is the 2 digits representation of the seconds of the minute

Z - is the character 'Z'

ExpiresOn (16 octets)

This field carries the GMT date for when this credential is to be considered expired. This field is 16 bytes long (the last byte must be set to '0x00') and contains the NULL-terminated ASCII string that represents the timestamp where the credential was issued. The format is the same as the ('IssuedOn') field. When the value is not set, the field should be set to { 0x00 }.

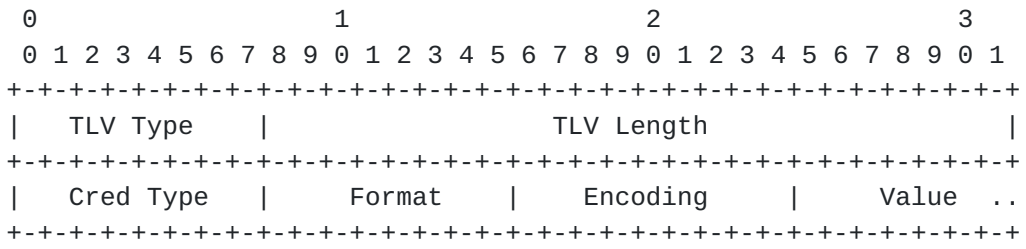
Credentials Length (uint16)

Length (in bytes) of the Credentials value. When used with a public-key type of credentials, this is the size of the key (e.g., for an RSA 2048 bit keys, this field should carry the value of 256). When used with a symmetric secret, this field carries the size of the secred (in bytes).

CredIDValue (> 1 octet)

The binary value of the credentials' identifier. This identifier can be the binary value of the SHA-256 calculated over the certificate, a username, or it could be a random handle. As long as the ID allows the peer and the server to uniquely (in its context) identify the credentials, the value of this field can be calculated in any way.

4.3.6. The Credentials-Data TLV



TLV Type (uint8)

<TBD> - Credentials-Data TLV

Length (uint24)

Provides the length of the TLV (> 3 octets)

Cred Type (uint8)

Provides the indication of the type of credentials. The allowed values for this field are listed in Section 8.3.

Format (uint8)

Provides the indication of the Format the credentials are in. The allowed values for this field are listed in Section 8.5.

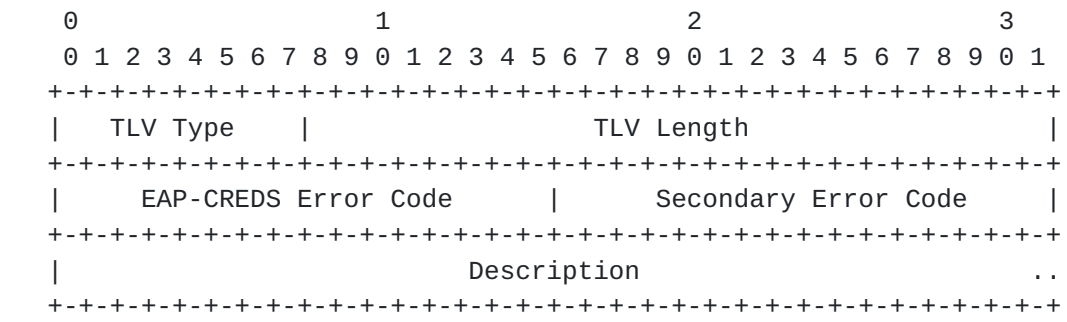
Encoding (uint8)

Provides the indication of the Encoding the credentials are in. The allowed values for this field are listed in Section 8.8.

Value (octet string)

This field carries the data for the credentials.

4.3.7. The Error TLV



TLV Type (uint8)

<TBD> - Challenge-Response-Data TLV

Length (uint24)

3 octets

EAP-CREDS Error Code (2 octets)

This field carries the EAP-CREDS error code. These code are related to the EAP-CREDS operations only and it should not be used to carry the Provisioning-Protocol specific error codes.

The error codes supported by this specifications are listed in Section 4.3.7.

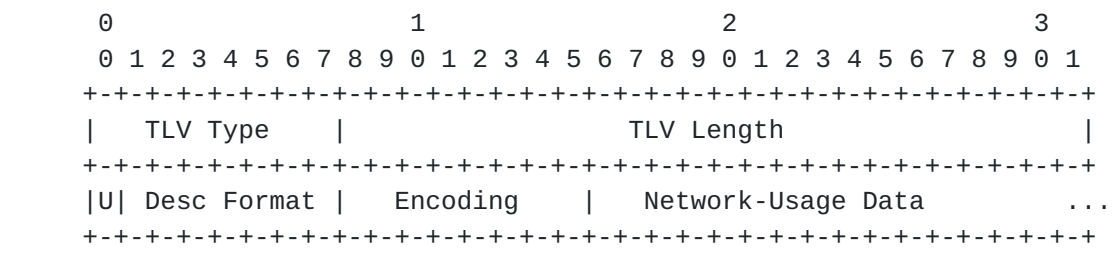
Secondary Error Code (2 octets)

This field is used to convery an error at the encapsulation layer (i.e., the provisioning protocol error). For example, this field can be used to convey a transport protocol error code (e.g., HTTP status code). Do not use this field to convery EAP-CREDS specific errors.

Description (> 1 octet)

The Description field is optional (i.e., when the Description Size is set to zero) and carries information about the error that occurred. The message may or may not be used by a user or an automated process for debugging purposes.

4.3.8. The Network-Usage TLV



TLV Type (uint8)

<TBD> - Network-Usage TLV

Length (uint24)

Variable Length TLV (Value must be > 2)

Description Format (uint8)

The Type of data encoded in the Peer Description Data. The initial values for this field are listed in Section 8.10.

Encoding (uint8)

Provides the indication of the Encoding the network usage description data is in. The allowed values for this field are listed in Section 8.8.

The 'U' field (1 bit)

The 'URL' bit ('U') is used to indicate if the value of the Network-Usage Data field is to be interpreted as a URL or as the actual data. In particular, if the value in the 'URL' bit is '1', then the value in the Network-Usage Data field is to be interpreted as the URL where the actual data can be downloaded from. Otherwise, if the 'URL' bit is set to '0', then the value in the Netowrk-Usage Data field is to be interpreted as the actual data (not a URL referencing it).

An example use of this bit is when the Peer wants to convey the URL of the MUD file [RFC8520]. In this case, the Peer can set the Network-Usage Data field to the Url of the MUD file related to the Peer.

Desc Format (7 bits)

This field provide the expected data format for the Network-Usage Data. For example, the value in this field could be set to 'MUD'

and have the 'U' bit set to '1' to provide the MUD-related information at credentials management time instead of at network-provisioning time (DHCP option). This possibility could help the Network controller to decide if the device shall be allowed to register its credentials or not.

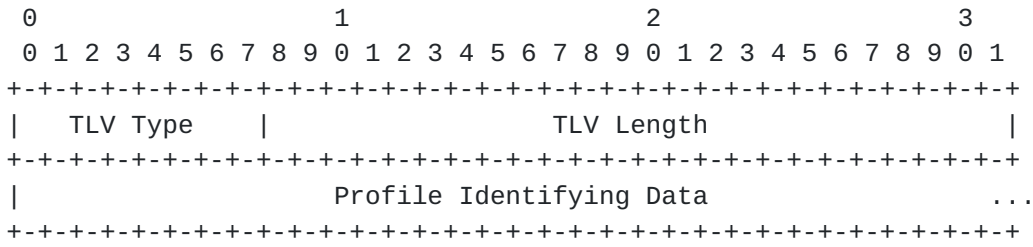
The list of initial values for this field is provided in Section 8.7.

Network-Usage Data (octet string)

This is additional information related to the device. In particular, this TLV can be used by the Peer to provide the Server with the description of the intended network usage or a URL that points to the same information.

For example, this field can be used to convey a MUD file (Manufacturer Usage Description) or the latest firmware-update manifest.

4.3.9. The Profile TLV



TLV Type (uint8)

<TBD> - Profile Identifying Data TLV

Length (uint24)

Length value should be >= 1

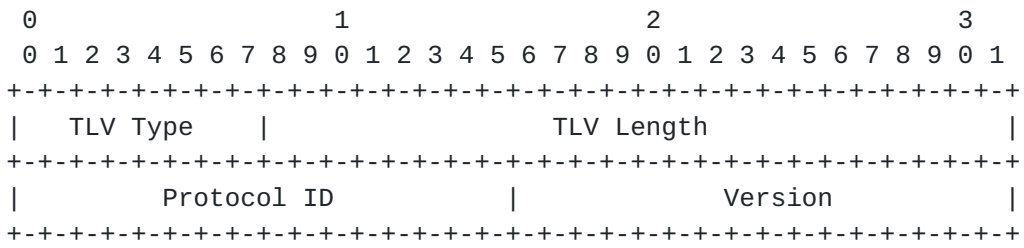
Profile Identifying Data (octet string)

The Profile Identifying Data is used to provide indication to the other party about which profiles are supported when requesting credentials management.

Also in this case, the data used in this field is left to be interpreted by the end-point and it is orthogonal to EAP-CREDS data types.

An example of values for this field, an end-point could use the string representation (i.e., dotted representation) of the Object Identifier (OID) of the specific profile supported (e.g., could be defined in the Certificate Policy of the credentials' provider).

4.3.10. The Protocol TLV



TLV Type (uint8)

<TBD> - Protocol TLV

TLV Length (uint24)

Fixed TLV Length value of 4.

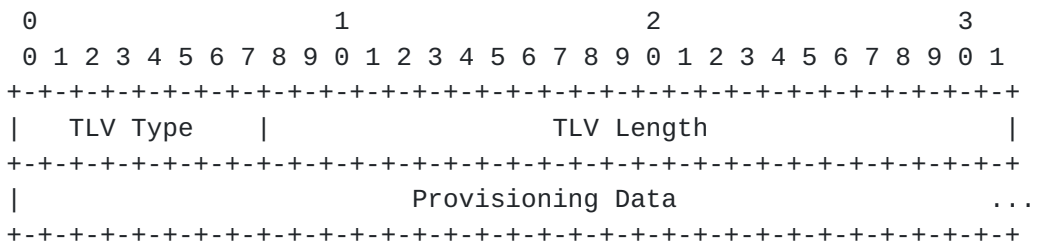
Protocol ID (uint16)

The Protocol ID value carries the id of a supported provisioning protocol. The initial list of values for the provisioning protocol identifiers can be found in Section 8.1.

Version (uint16)

The Version (Protocol Version) value represents the specific version of the identified provisioning protocol. When no version is specified for a protocol (i.e., either it does not support multiple versions or it does not matter), the value of this field should be set to '0x0'.

4.3.11. The Provisioning-Data TLV



TLV Type (uint8)

<TBD> - Provisioning-Data TLV

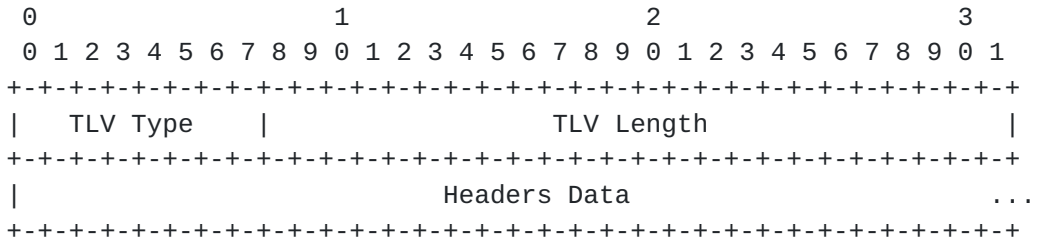
Length (uint24)

3 octets

Headers Data (> 1 octet)

This field carries the provisioning protocol's messages.

4.3.12. The Provisioning-Headers TLV



TLV Type (uint8)

<TBD> - Provisioning-Headers TLV

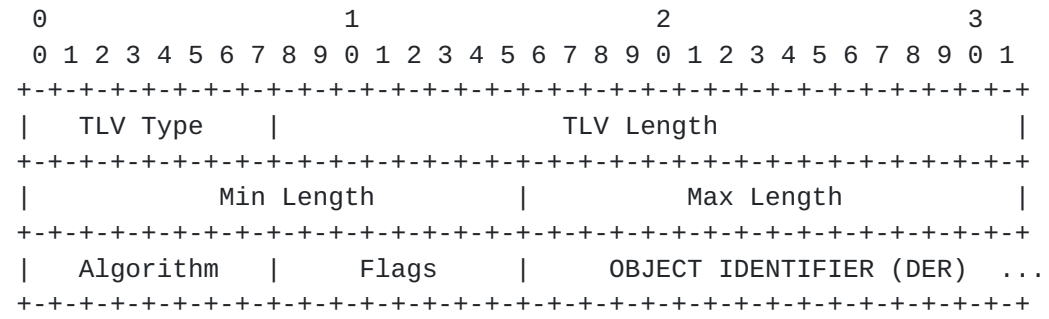
Length (uint24)

3 octets

Headers Data (> 1 octet)

This field carries the meta-data (if any) that might be associated with the transport-layer normally used with the provisioning protocol. For example, this TLV can carry the set of HTTP headers required by EST or ACME.

4.3.13. The Provisioning-Params TLV



TLV Type (uint8)

<TBD> - Provisioning-Params TLV

Length (uint24)

Provides the length of the TLV (>= 6 octets)

Min Length (uint16)

Provides the minimum allowed size size for the credentials. This value has meaning depending on the context of the credentials, however sizes are always expressed in bytes.

For example, when used with a symmetric key or a password, the ('Min Length') and ('Max Length') refer to the minimum and maximum size of the password data. The ('Algor OID') field can be omitted in this case.

On the other hand, when referring public-key credentials, this field should carry the size of the modulus of the key. For example, for an RSA 2048 bit keys, the field should carry the value of 256. For an ECDSA that uses the prime256r1 curve, this field should carry the value of 32 and the Algor OID should be the DER representation of the specific value of the curve (i.e., the DER representation of '1.2.840.10045.3.1.7').

Max Length (uint16)

Provides the indication maximum size of the credentials. This value has meaning depending on the context of the credentials, however sizes are always expressed in bytes.

The same considerations apply to this field as well as the ('Min Length') one discussed above.

Flags (uint8)

Provides a BITMASK that can be used to provide information about the status of the credentials (e.g., if the use marks the credentials to be compromised). The bits have the following meaning:

Bit 0 - Credentials (or part of it) are to be generated on the server

Bit 1 - Credentials (or part of it) are to be generated on the peer

Bit 2 - Credentials are to be generated on dedicated hardware

Bit 3 - Reserved

Bit 4 - Reserved

Bit 5 - Reserved

Bit 6 - Reserved

Bit 7 - Reserved

When using public-key based credentials, the bits 0 and 1 are mutually exclusive.

When using passwords or shared secrets, if bit 0 is set, then the secret is generated by the server and then sent to the client. On the other hand, if bit 1 is set, then the secret is generated by the peer and then sent to the server. Ultimately, if both bits are set, then the Server generates the first part of the password and sends it to the Peer, while the Peer generates the second part of the password and sends it to the Server. The password to be used for future authentication is the concatenation of the two shares of the password: first the one from the Server, then the one from the Client.

NOTE WELL: Last but not least, since these passwords/secrets are meant to be used in a automated fashion, there is no restriction around the character set to use or their interpretation. Therefore,it is good practice to generate

random passphrases that use the full 8-bit character set (on client and server) to maximize the secret's search space.

Algorithm (uint8)

Provides the indication of the algorithm used for the generation of the credentials. The allowed values for this field are listed in Section 8.4.

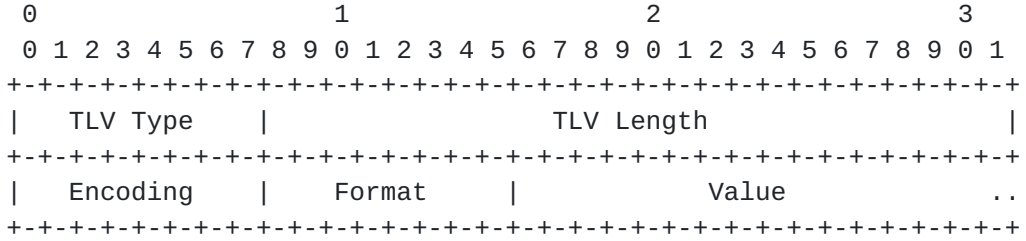
Object Identifier (binary; > 1 octet)

Provides the indication of additional parameters that are needed to be encoded for the credentials. This value is used only when the credentials use public-key cryptography - this field carries additional information about the generation algorithm to be used. We provide some useful values that can be used as reference:

OID Name	Dotted Representation	Binary Encoding
secp256r1 curve	1.2.840.10045.3.1.7	06 08 2A 86 48 CE 3D 03 01 07
secp384r1 curve	1.2.840.10045.3.1.34	06 08 2A 86 48 CE 3D 03 01 22
secp521r1 curve	1.2.840.10045.3.1.35	06 08 2A 86 48 CE 3D 03 01 23
X25519 curve	1.3.101.110	06 03 2B 65 6E
X25519 curve	1.3.101.110	06 03 2B 65 6E
X448 curve	1.3.101.111	06 03 2B 65 6F
Ed25519 curve	1.3.101.112	06 03 2B 65 70
Ed448 curve	1.3.101.113	06 03 2B 65 71

Table 3: Object Identifiers Examples

4.3.14. The Certificate-Request TLV



TLV Type (uint8)

<TBD> - Token-Data TLV

TLV Length (uint24)

Provides the length of the TLV (> 3 octets)

Encoding (uint8)

Provides the indication of the Encoding the credentials are in. The allowed values for this field are listed in Section 8.8.

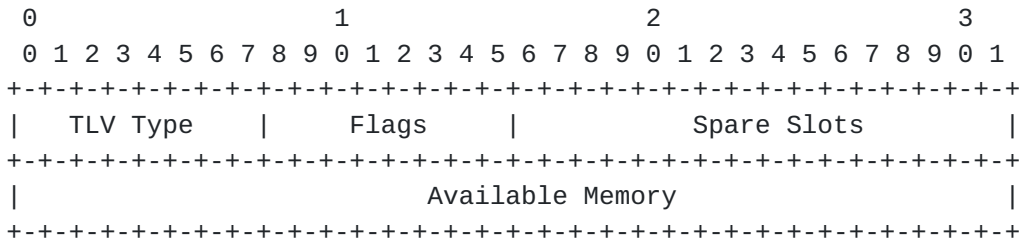
Format (uint8)

Provides the indication of the type of credentials. The allowed values for this field are listed in Section 8.5.

Value (octet string)

This field carries the data for the credentials.

4.3.15. The Storage-Info TLV



TLV Type (uint8)

<TBD> - Store-Info TLV

Flags (8 bits)

Provides information about the status and type of store and limited information about its capabilities. The bits have the following meaning:

Bit 0 - If set, the store supports RSA keys (software)

Bit 1 - If set, the store supports RSA keys (hardware)

Bit 2 - If set, the store supports ECDSA keys (software)

Bit 3 - If set, the store supports ECDSA keys (hardware)

Bit 4 - If set, the store supports symmetric keys

Bit 5 - If set, the store supports generic tokens

Bit 6 - If set, the store is immutable (no key generation or deletion)

Bit 7 - Not Used

Spare Slots (uint16)

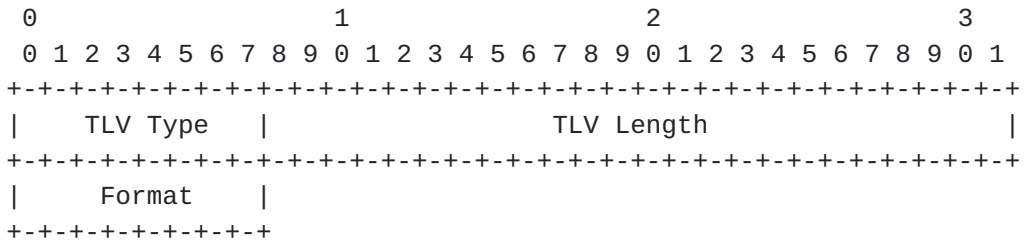
Provides the number of available slots where to store credentials. When no more slots are available, the value of '0' should be used to indicate to the Server that a credential must be deleted before a new one can be created.

When the number of slots is not fixed or not known, the value of { 0xFF, 0xFF } shall be used.

Available Memory (uint32)

This field carries the size (in bytes) of the spare memory on the Peer's secrets' store.

4.3.16. The Supported-Formats TLV



TLV Type (uint8)

<TBD> - Supported-Format TLV

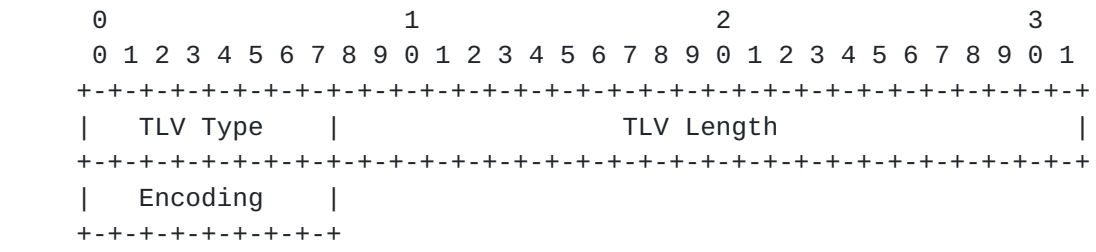
TLV Length (uint24)

Provides the length of the TLV. This field must be set to 1.

Format (uint8)

Provides the details about the supported format. Multiple formats TLVs can be used in the Peer's ('Init') message to provide the Server with the Peer's capabilities.

4.3.17. The Supported-Encoding TLV



TLV Type (uint8)

<TBD> - Store-Info TLV

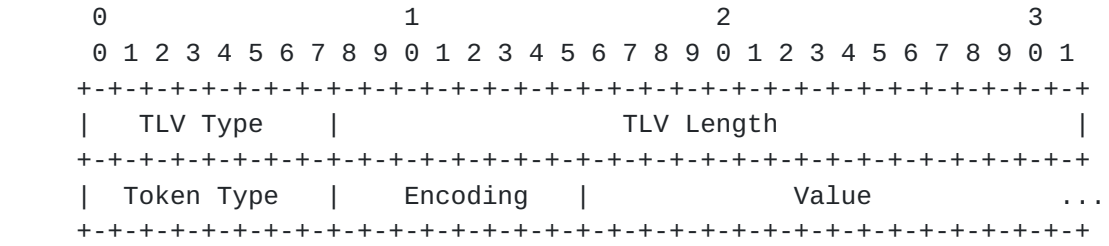
TLV Length (uint24)

Provides the length of the TLV. The field has a fixed value of 1.

Encoding (uint8)

Provides the indication of the supported Encoding by the End Point. This provides the indication to the Server of the capability of the Peer. The allowed values for this field are listed in Section 8.8.

4.3.18. The Token-Data TLV



TLV Type (uint8)

<TBD> - Token-Data TLV

TLV Length (uint24)

Provides the length of the TLV (> 3 octets)

Token Type (uint8)

Provides the indication of the type of credentials. The allowed values for this field are listed in Section 8.2.

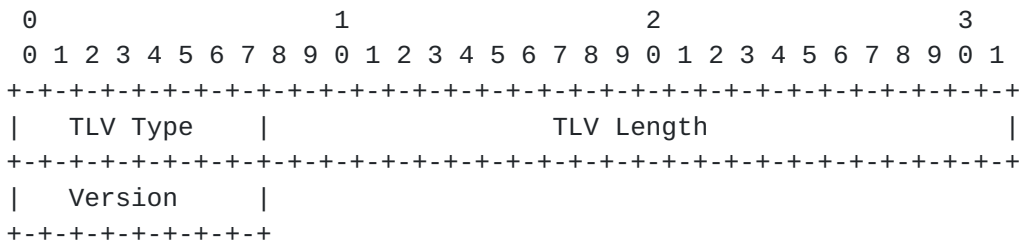
Encoding (uint8)

Provides the indication of the Encoding the credentials are in. The allowed values for this field are listed in Section 8.8.

Value (octet string)

This field carries the data for the credentials.

4.3.19. The Version TLV



TLV Type (uint8)

<TBD> - Version TLV

TLV Length (uint24)

Provides the length of the TLV. The field has a fixed value of 1.

Version (uint8)

The Version field represents the specific version of the EAP-CREDS protocol that are supported by the end point. When multiple versions of EAP-CREDS are supported, multiple ('Version') TLVs can be used.

When no version is specified (i.e., either it does not support multiple versions or it does not matter), the value of this field should be set to '0x0' (any version).

5. EAP-CREDS Messages

This section describes each message and what TLVs are allowed or required. EAP-CREDS defines the following values for the Message Type (Type):

Message Type	Name	Description
0	EAP-CREDS-Init	Initialization Phase
1	EAP-CREDS-Provisioning	Carries Provisioning Protocol Messages
2	EAP-CREDS-Validate	Validates newly installed credentials

Table 4: EAP-CREDS Message Types

5.1. The EAP-CREDS-Init Message

The EAP-CREDS-Init message type is used in Phase One only of EAP-CREDS. The message flow is depicted in Section 3.3. This message supports the following TLVs: Version, Protocol, Credentials-Info, and Error.

5.1.1. EAP Server's Init Message

EAP-CREDS starts with an ('EAP-CREDS-Init') message from the server. This message MAY contain zero, one, or more ('Version') TLVs and, optionally, a ('Challenge-Data') TLV.

The first message from the server is the one that starts Phase One, therefore the Server MUST set the headers' 'S' bit to '1' (Start) and the headers' 'Phase' value to '1' (Phase One).

The Server uses one or more ('Version') TLVs in the EAP-Request/EAP-CREDS(Type=Init) message to provide the Peer with the list of EAP-CREDS versions supported. If omitted, the implict version of EAP-CREDS used in the session is one ('0x1'). If the Server detects multiple occurrences of this TLV in the reply from the Peer, an error shall be issued and the EAP-CREDS session should be terminated.

In case Token-Based registration is enabled on the Server, the Server MUST include, in its Init message, a ('Challenge-Data') field that can be used by the client to provide challenge data for proof-of-possession of secrets.

5.1.2. EAP Peer's Init Message

The Peer MUST reply to the Server's ('EAP-CREDS-Init') message with its own ('EAP-CREDS-Init') one. The Peer SHOULD include one ('Version') TLV in its first message to indicate the version of EAP-CREDS that the client wants to use for the session. The Peer MUST also provide the list of supported provisioning protocols (via one or more the 'Protocol' TLV), the list and status of the installed credentials (via the 'Credentials-Info' TLV). The Peer MAY include authorization data when registering new credentials (e.g., an authorization token or a device certifcate) via the ('Token-Data') and ('Challenge-Response') TLV.

The Peer MUST include one ('Credentials-Info') TLV for each credential the Network is authorized to manage. Typically, a Peer will include only one ('Credentials-Info') TLV in its ('EAP-CREDS-Init') message, but there might be cases where multiple types of credentials are available and selected depending on the location and other factors (e.g., X.509 certificate and username/password combination).

In case the Peer does not have any credentials available yet, it does not add any ('Credentials-Info') TLV - leaving the Server with the only action possible: Registration. In this case, the Peer SHOULD include authorization information via the ('Token-Data') TLV as described in Section 5.1.2.1. Additionally, the Peer can add the ('Profile') TLV to indicate a preferred profile for the credentials.

5.1.2.1. Bootstrapping Peer's Trustworthiness

When the Peer does not have any valid credentials for the Network that it is authenticating to, it does not provide any ('Credentials-Info') TLV. This indicates to the Server that new credentials MUST be registered before the Peer is allowed on the network.

The Registration process might rely on information exchanged during the Provisioning Process in Phase Two. However, if an authorization mechanism is not available from the supported provisioning protocol and no credentials are available on the Peer, EAP-CREDS provides a simple machanism for the Peer to leverage an out-of-band token/passphrase/ott that may be already available on the Peer (e.g., a device certificate or a 'spendable' credentials token like a kerberos ticket or a crypto-currency transaction) and that can be verified by the Server.

In particular, when the Peer wants to register new credentials (and the Server requires the use of additional authorization data) it may need to provide (a) a Token, (b) a challenge value, and (c) a

response to the challenge value. To do so, the Peer MUST encode the token in a ('Token-Data') TLV, the challenge value in a ('Challenge-Data') TLV, and, finally, the response to the challenge in the ('Challenge-Response') TLV.

The use of ('Challenge-Data') and ('Challenge-Response') TLVs is optional, however it is suggested that if a token is used for bootstrapping the trust, it should provide a way to verify a secret associated with it.

It is also very important that the authorization token is disclosed only to authorized servers - the Peer MUST NOT disclose authorization tokens that are not meant for the network that is being accessed. This can be done, usually, by verifying the identity of the Server first (in the outer mechanism) and then verify that the target of the Token is the Server the Client is talking to.

5.1.3. The EAP-CREDS-Provisioning Message

The EAP-CREDS-Provisioning message type is used in Phase Two only of EAP-CREDS. The message flow is depicted in Section 3.4. This message type supports the following TLVs: Protocol, Profile, Credentials-Info, Provisioning-Headers, Provisioning-Data, Token-Data, and Error.

After the exchange of phase one messages, the Server MAY start phase two by issuing an ('EAP-CREDS-Provisioning') message for the Peer where it encodes all the required details for starting the provisioning process. In particular, the server sends the selected ('Action'), ('Protocol'), and metadata to the client in a EAP-Request/EAP-CREDS(Type=Provisioning) message. The header's 'S' bit MUST be set to '1' (Start) and the 'Phase' value set to '2' (Phase Two begins).

> NOTE WELL: After the initial message, the only TLVs that are allowed in messages coming from the server are the usual ('Provisioning-Headers') ('Provisioning-Data'), and ('Error').

The client checks that all the selected parameters are supported for the selected credentials and, if no errors are detected, it sends its first ('EAP-CREDS-Provisioning') message to the Server with the ('Provisioning-Headers') and ('Provisioning-Data') TLVs only.

From now on, the conversation between the Peer and the Server continues until an error is detected or the provisioning protocol completes successfully.

If no other actions, the server MAY continue with phase three or issue a success message and terminate the EAP session.

NOTE WELL: When the SPP protocol is used, the protocol messages that are encoded inside the ('Protocol-Data') TLV are composed of sets of TLVs as defined in this document. The overall message size is provided by the size of the ('Protocol-Data') TLV that encapsulates the SPP-specific TLVs. This design choice provides symmetry in implementing support for SPP when compared to other provisioning protocols.

5.1.4. The EAP-CREDS-Validate Message

The EAP-CREDS-Validate message type is used in Phase Three only of EAP-CREDS. The message flow is depicted in Section 3.5. This message type supports the following TLVs: Protocol, Credentials-Info, Provisioning-Headers, Provisioning-Data, Token-Data, and Error.

After Phase One (and/or Phase Two) ends, the Server MAY start phase three by issuing an ('EAP-CREDS-Validate') message for the Peer where it encodes all the required details for starting the validation process. In particular, the server sends the ('Credentials-Info'), a ('Challenge'), and the ('Phase-Control') TLVs in a EAP-Request/EAP-CREDS(Type=Validate) message. The ('Phase-Control') TLV should carry the '1' value for the 'S' bit (Start) and the number '3' for its value (Phase Three begins).

The Peer generates the answer to the Challenge and sends back a EAP-Response/EAP-CREDS(Type=Validate) message with the ('Challenge-Response') and an optional ('Challenge') field (only for server-side validation of the symmetric credentials). If the Peer requested server-side validation of the credentials, the Server MUST include (if a symmetric secret) the response to the Peer-issued ('Challenge') TLV by computing the response and adding it to the ('Challenge-Response') TLV in its reply.

Finally, in the last message, the Server (if Phase Three is to be ended) SHALL include the ('Phase-Control') TLV with the 'S' bit set to '0' (end of phase) and the value set to '3' (Phase Three ended).

At this point, EAP-CREDS has terminated all possible operations and can be terminated. The Server can now terminate the EAP session successfully. In case the Peer was not authenticated during the tunnel establishment (i.e., no credentials were already available on the Peer), the Server should terminate the EAP session with a Failure (thus requiring the device to re-attach and authenticate to the network - phase two should have provided the Peer with the credentials to use for authenticating to the Network).

6. Error Handling in EAP-CREDS

This section provides a description of the error handling by using the CREDS-Error-TLV in a CREDS message.

7. The Simple Provisioning Protocol (SPP)

EAP-CREDS supports a Simple Provisioning Protocol (SPP) which comprises of a series of messages that enable the management not only of certificates, but also of other types of credentials like username/password pairs, asymmetric keys, and symmetric keys.

The Simple Provisioning Protocol (SPP), described in this section, behaves as any other provisioning protocol: its messages are encapsulated in the ('Provisioning-Data') TLVs in the second phase of the protocol. SPP does not make use of any ('Provisioning-Headers') TLVs because its messages are all self-contained (no transport-protocol specific options are needed).

When no ('Credentials-Info') TLVs have been provided by the client, the Server knows that the device does not have valid credentials it wants to use to access the Network. In this case, EAP-CREDS/SPP supports the use of Tokens to kick-off the registration process. The type, format, or encoding of the Token is orthogonal to EAP-CREDS/SPP which treats the token as a black-box field (i.e., it SHOULD NOT try to interpret or parse its contents).

> NOTE WELL: During Phase One, the Peer MAY include the ('Token-Data') TLV in its EAP-CREDS-Init message to provide the needed authorization to register a new set of credentials. The Server might not allow the registration of new credentials if the required authorization (i.e., the Token) was not provided during the initialization phase.

In the case where an authorization token is used, different usage patterns are supported. For tokens that require an associated verifiable proof-of-possession, the Peer can include a ('Challenge-Response') TLVs.

The ('Challenge-Data') TLV provided by the Server MUST be used to convey the challenge data (usually some random value) to compute the contents of the ('Challenge-Response') TLV.

The ('Challenge-Response') TLV is used, instead, to encode the response to the challenge data. The ('Challenge-Response') TLV is generated by the Peer and verified by the Server. At minimum, the ('Challenge-Response') TLV SHOULD be calculated over the values of

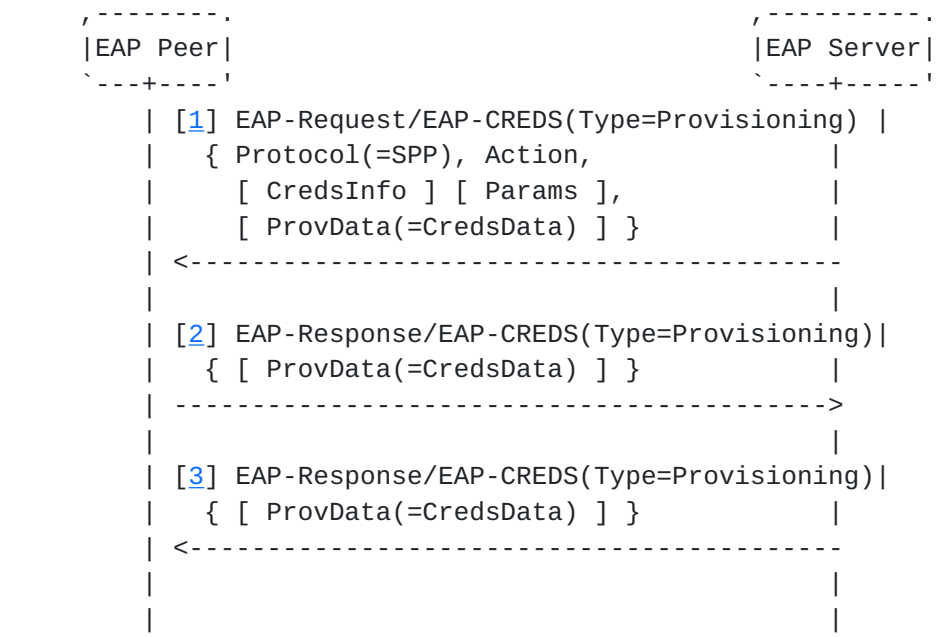
the ('Token-Data') and the ('Challenge-Data') TLVs to make sure that the authentication covers the token's data as well.

NOTE WELL: The use of the ('Token-Data'), ('Challenge-Data'), and ('Challenge-Response') TLVs in the Peer's Init message is be used only to bootstrap trust between the Server and the Peer. If the Server accepts the authorization information as valid, the Peer is enabled for registering new credentials. This should happen only when the Peer does not have valid credentials or when the server wants to provision a different type of credentials (i.e., Action=(Register)). Other methods to provide authorization information might be provided by the selected provisioning protocol: in this case, the Server MAY enable registration of new credentials when no authorization data is provided in the 'Init' message from the client and delegate the validation of the authorization data during Phase Two.

7.1. SPP Message Format

The SPP Messages are constructed with zero, one, or more TLVs and encoded in the ('Provisioning-Data') TLV in EAP-CREDS/Provisioning message types. The size of the encpasulating ('Provisioning-Data') TLV provides the size of the whole message.

7.2. SPP Message Flow



SPP was designed to provide an easy alternative to more complex provisioning protocols. When no extra flexibility is needed, SPP provides an easy-to-implement alternative that can handle not only certificates, but also symmetric secrets and access tokens provisioning. In this section we provide the generic flow of messages for SPP and specific examples for certificates, username/password, and token provisioning.

EAP-CREDS defines several actions for a set of credentials and they are listed in Section 8.9.

When a Peer wants to join a network it may or may not have have the needed credentials to do so. In case the Peer does not have valid credentials yet, the Server MAY start Phase Two with the intention of registering a new set of credentials. Alternatively, the Server MAY start Phase Two when the presented credentials information from the Peer triggers the Renew or the Remove action.

[1] The Server sends EAP-Request/EAP-CREDS(Type=Provisioning)

When registering new credentials, the first message from the Server, MUST not carry a ('Credentials-Info') TLV since there is no targeted credentials to apply the action on (i.e., for other actions - like 'renew' or 'remove' - the TLV would be required to identify the right set of credentials to renew or delete).

In SPP, the Server sets the ('Protocol') TLV to SPP, the ('Action') TLV to 'Register', 'Renew', or 'Remove'. When provisioning (or registering) new credentials for the Peer, the Server also sets the ('Provisioning-Params') TLV (or Params) to the type of credentials to be provisioned. The Server also sets any relevant constraints, and, optionally, the ('Profile') TLV.

NOTE WELL: If the Peer is authorized to register a new set of credentials, then the first message from the Server will have the ('Action') TLV set to 'register' and no ('Credentials-Info') TLV is present in the Server's message. In case server-side generation is used, an additional ('Credentials-Info') TLV MAY be encoded inside the ('Provisioning-Data') TLV.

If the type of credentials is symmetric and the parameters call for server-side generation of a symmetric key share, the Server MUST also include its own generated share in a ('Credentials-Data') TLV inside the ('Provisioning-Data') one (the data for the provisioning protocol are encapsulated in the 'Provisioning-Data'

TLV for any protocol used during Phase Two - SPP is no exception to this rule).

In case Server-side only is selected, the Server MUST send the new credentials in its message and include the ('Credentials-Info') TLV. If no other credentials need to be managed, the Server MUST end Phase Two by setting the appropriate bits in the EAP-CREDS headers as well.

[2] The Peer sends EAP-Response/EAP-CREDS(Type=Provisioning)

When Peer-generation is selected (either Peer-only or combined Peer and Server side) and Phase Two has not terminated yet, the Peer MUST reply to the Server's message with its own 'Provisioning' response. The response MUST carry either (a) its own generated share of the key in a ('Credentials-Data') TLV (if the credentials that are provisioned are symmetric and the configuration calls for a share of the key to be provided by the Peer) or (b) a PKCS#10 request in a ('Certificate-Request') TLV (also in this case, only if client-side generation was enabled by the Server) that is generated by using the parameters provided by the Server in the ('Provisioning-Params') TLV.

[3] The Server sends EAP-Request/EAP-CREDS(Type=Provisioning)

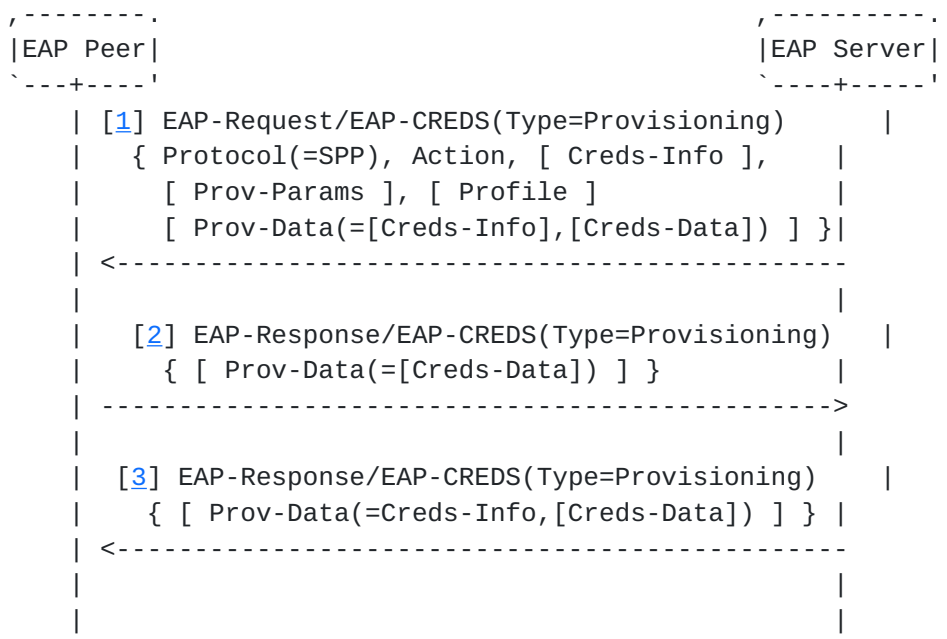
The last message of SPP is from the Server and it is used to deliver the finalized value of the credentials and/or associated metadata. In case the credentials being provisioned are Certificate-based, the Server MUST include the issued certificate in its reply. The issued credentials shall be encoded in a ('Credentials-Data') TLV inside the ('Provisioning-Data') one. In case that the selected format supported/selected by the Peer and the Server does not provide the possibility to encode the full chain (i.e., intermediate and Root CAs) in the response, the Server MUST add one ('Certificate-Data') TLV for each certificate in the chain (including the Root CA's certificate).

The Server MUST include the ('Credentials-Info') TLV in its message. This provide the Peer with some additional data (e.g., the 'Profile' or the 'Identifier' associated with the credentials that were provisioned/managed).

In the case where additional credentials need to be managed, the Server can continue Phase Two by issuing a new [1] message where the tuple Action/Credentials must be unique for the current EAP-CREDS session.

The Server can now decide to start Phase Three (suggested if new credentials were provisioned or renewed) or to terminate the EAP session successfully.

7.2.1. SPP Symmetric Secrets Management



EAP-CREDS/SPP can provision symmetric secrets (e.g, username/password, API keys, or SIM-based keys), tokens (e.g., username/password OAuth or Kerberos tokens), or asymmetric credentials (e.g., X.509 certificates or Key Pairs). This section focuses on provisioning symmetric secrets only. The message flow is provided in Section 7.2.1

EAP-CREDS/SPP provides the possibility for shared secret to be generated in different ways:

1. Server-Side Generated

2. Client-Side Generated

3. Both Client-Side and Server-Side Generated

In particular, when initiating the second phase of the protocol, the ('Provisioning-Params') TLV is used to specify how to generate the secret (see Section 4.3.13).

7.2.1.1. Server Side Only Generation

[TO BE EDITED]

Figure 1: SPP Message Flow for Server-Side only secrets provisioning

The message flow for deploying a server-side only credential (i.e., during registration or renewal) consists of only one message from the server. The flow is depicted in Figure 1.

In this case, the Server sends the first Provisioning message (which is also the last one), which MUST carry, the following data:

- The ('Credentials-Info') TLV that specifies the info for the provisioned secret, and
- The ('Protocol') TLV that specifies the provisioning protocol to be used, and
- The ('Action') TLV that provides the action to be performed ('Registration') or ('Renew'), and
- The ('Provisioning-Params') TLV that provides the generation parameters to the Peer, and

The Server also includes, encoded in the ('Provisioning-Data') TLV, the following data:

The ('Credentials-Info') TLV that provides the metadata associated with teh generated secret

The ('Credentials-Data') TLV that provides the secret that is provisioned to the Peer

Server-side secrets' generation can be used to generate username/ password combinations, API Keys, SIM-based credentials, or tokens.

7.2.1.2. Client Side Only Generation

[TO BE EDITED]

Figure 2: SPP Message Flow for Client-Side only secrets provisioning

The message flow for deploying a client-side only credential (i.e., during registration or renewal) consists of the full three messages exchange. The flow is depicted in Figure 2.

In this case, the Server MUST include, in its first Provisioning message and encoded in the ('Provisioning-Data') TLV, the following data:

- The ('Credentials-Info') TLV that specifies the target credentials, and
- The ('Protocol') TLV that specifies the provisioning protocol to be used, and
- The ('Action') TLV that provides the action to be performed ('Registration') or ('Renew'), and
- The ('Provisioning-Params') TLV that provides the generation parameters to the Peer, and

Notice that the Server does not include any ('Credentials-Data') TLV in its first message because the Server is not involved in the secret generation (client-side only).

The Peer MUST reply with its own Provisioning message where the Peer MUST encode the following data in the ('Provisioning-Data') TLV:

The ('Credentials-Data') TLV that provides the secret that is being registered

The credentials data MUST conform to the specifications the Server provided in the ('Provisioning-Params') TLV.

The final message is from the Server and it MUST contain (if no errors were detected), the following TLVs encoded, as usual, in the ('Provisioning-Data') TLV:

The ('Credentials-Info') TLV that specifies the metadata associated with the generated secret, and

The ('Credentials-Data') TLV that provides the secret that is provisioned to the Peer

Client-side secrets' generation should be used with caution and an evaluation of the quality of the generated credentials MUST be performed to make sure that the security of the generated secret is adequate for accessing the network. Since evaluating the quality of a secret is quite a difficult tasks, the use of this generation mode MUST be evaluated carefully and selected accordingly to acceptable risk profiles.

7.2.1.3. Client and Server Side Generation

When registering or renewing credentials and the secret generation is split between the Server (1st share) and the Peer (2nd share), the message flow is the same as Section 7.2.1.2 with the following exceptions:

- The Server MUST send its own share of the secret by including a ('Credentials-Data') TLV in its first message.

All other parameters remain the same.

Co-generation of the secret is the most secure option because both parties can provide the required randomness in their own share of the secret.

7.2.2. SPP Key Pair Provisioning

EAP-CREDS/SSP defines the following flow of messages for requesting the provisioning of key pairs (public and private keys).

7.2.2.1. Server Side Only Generation

[This case covers the server-side generation of KeyPair and Certificate]

7.2.2.2. Client Side Only Generation

[This case covers the registration of a self-signed or already available (e.g., device) certificate]

7.2.2.3. Client and Server Side Generation

This use-case is not supported. In other words, for the provisioning of Key Pairs, the ('Provisioning-Params') can not have both the peer-generation and server-generation bits set.

7.2.3. SPP Certificate Provisioning

EAP-CREDS/SSP defines the following flow of messages for requesting the provisioning of credentials.

7.2.3.1. Server Side Only Generation

[This case covers the server-side generation of KeyPair and Certificate]

7.2.3.2. Client Side Only Generation

[This case covers the registration of a self-signed or already available (e.g., device) certificate]

7.2.3.3. Client and Server Side Generation

[This case covers the generation of the KeyPair on the Peer and the generation of the certificate on the Server]

7.2.4. SPP Token Provisioning

EAP-CREDS/SSP defines the following flow of messages for requesting the provisioning of token-based credentials.

7.2.4.1. Server Side Only Generation

[This case covers the server-side generation of the Token and possibly associated key]

7.2.4.2. Client Side Only Generation

[This case covers the registration of a self-signed or already available (e.g., device) certificate]

7.2.4.3. Client and Server Side Generation

[This case covers the generation of the KeyPair on the Peer and the generation of the Token that cointains the reference to the key on the Server]

8. IANA Considerations

This document uses a new EAP type, EAP-CREDS, whose value (TBD) MUST be allocated by IANA from the EAP TYPEs subregistry of the RADIUS registry. This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to the EAP-CREDS protocol, in accordance with [RFC8126].

The EAP Method Type number for EAP-CREDS needs to be assigned.

This document also requires IANA to create new registries as defined in the following subsections.

8.1. Provisioning Protocols

Message Type	Purpose
0	Unspecified
1	Simple Provisioning Protocol (SPP)
2	Basic Certificate Management Protocol (CMP-S)
3	Full Certificate Management Protocol (CMP-F)
4	Enrollment over Secure Transport (EST)
5	Certificate Management over CMS (CMC)
6	Automatic Certificate Management Environment (ACME)
...	...
49141 ... 65534	Vendor Specific

Table 5: EAP-CREDS Inner Protocol Identifiers

Assignment of new values for new cryptosuites MUST be done through IANA with "Specification Required" and "IESG Approval" as defined in [RFC8126].

8.2. Token Types

Token Type	Description
0	Unspecified
1	JWT
2	Kerberos
3	OAuth
4	Certificate
200..254	Vendor Specific

Table 6: Token Types

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

8.3. Credentials Types

Credentials Type	Description
0	X.509 Certificate
1	Public Key
2	Symmetric Key
3	Username and Password
4	AKA Subscriber Key
5	Bearer Token
6	One-Time Token
7	API Key

Table 7: Credentials Types

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

8.4. Credentials Algorithms

ID	Algorithm
0	None
1	RSA
2	ECDSA
3	XMMS
4	AKA Subscriber Key
5	OAuth
6	Kerberos4
7	Kerberos5
200-254	Reserved

Table 8: Credentials Algorithms

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

8.5. Credentials Datatypes

ID	Data Type
0	None (Binary)
1	PKCS#8
2	PKCS#10
3	PKCS#12
4	PublicKeyInfo
200-254	Reserved

Table 9: Credentials Datatypes

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

8.6. Challenge Types

ID	Data Type
0	Not Specified
1	EAP-CREDS-ASYMMETRIC
2	EAP-CREDS-SYMMETRIC

Table 10: Challenge Type

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

8.7. Network Usage Datatypes

ID	Data Type
0	Vendor-Specific
1	Manufacturer Usage Description [RFC8520]
2	Network Access Granting System
3	Firmware Manifest
4..127	Reserved for Future Use

Table 11: Network Usage Datatypes

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

8.8. Credentials Encoding

ID	Encoding
0	None (Raw)
1	DER
2	PEM
3	Base64
4	JSON
5	XML
6	ASCII
7	UTF-8
200-254	Reserved

Table 12: Credentials Encoding

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

8.9. Action Types

ID	Data Type	Description
0	Registration	Registers New Credentials
1	Renewal	Renew an Existing Credential
2	Remove	Removes an Existing Credential
200-254	n/a	Reserved

Table 13: Action Types

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

8.10. Usage Metadata Types

Type	Description
0	Binary (Unspecified)
1	MUD File
2	TEEP Manifest

Table 14: Usage Metadata Types

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

9. Security Considerations

Several security considerations need to be explicitly considered for the system administrators and application developers to understand the weaknesses of the overall architecture.

The most important security consideration when deploying EAP-CREDS is related to the security of the outer channel. In particular, EAP-CREDS assumes that the communication channel has been properly authenticated and that the information exchanged between the Peer and the Server are protected (i.e., confidentiality and integrity).

For example, if certificate-based authentication is used, the server presents a certificate to the peer as part of the trust establishment (or negotiation). The peer SHOULD verify the validity of the EAP server certificate and SHOULD also examine the EAP server name presented in the certificate in order to determine whether the EAP server can be trusted. When performing server certificate validation, implementations MUST provide support for the rules in [RFC5280] for validating certificates against a known trust anchor.

10. Acknowledgments

The authors would like to thank everybody who provided insightful comments and helped in the definition of the deployment considerations.

11. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <https://www.rfc-editor.org/info/rfc4210>.

[RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008, <https://www.rfc-editor.org/info/rfc5272>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6402] Schaad, J., "Certificate Management over CMS (CMC) Updates", RFC 6402, DOI 10.17487/RFC6402, November 2011, <https://www.rfc-editor.org/info/rfc6402>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/info/rfc7030>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <https://www.rfc-editor.org/info/rfc8520>.

Author's Address

Massimiliano Pala
CableLabs
858 Coal Creek Cir
Louisville, CO 80027
US

Email: m.pala@openca.org
URI: http://www.linkedin.com/in/mpala