Stream: Internet Engineering Task Force (IETF)
RFC: 9154
Category: Standards Track
Published: December 2021
ISSN: 2070-1721
Authors: J. Gould, *Verisign, Inc.*; R. Wilhelm, *Verisign, Inc.*

RFC 9154
Extensible Provisioning Protocol (EPP) Secure Authorization Information for Transfer

Abstract

The Extensible Provisioning Protocol (EPP) (RFC 5730) defines the use of authorization information to authorize a transfer of an EPP object, such as a domain name, between clients that are referred to as "registrars". Object-specific, password-based authorization information (see RFCs 5731 and 5733) is commonly used but raises issues related to the security, complexity, storage, and lifetime of authentication information. This document defines an operational practice, using the EPP RFCs, that leverages the use of strong random authorization information values that are short lived, not stored by the client, and stored by the server using a cryptographic hash that provides for secure authorization information that can safely be used for object transfers.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9154.

Copyright Notice

Table of Contents

1. Introduction 3
   1.1. Conventions Used in This Document 4
2. Registrant, Registrar, Registry 5
3. Signaling Client and Server Support 6
4. Secure Authorization Information 6
   4.1. Secure Random Authorization Information 7
   4.2. Authorization Information Time To Live (TTL) 7
   4.3. Authorization Information Storage and Transport 8
   4.4. Authorization Information Matching 8
5. Create, Transfer, and Secure Authorization Information 9
   5.1. <Create> Command 9
   5.2. <Update> Command 10
   5.3. <Info> Command and Response 13
   5.4. <Transfer> Request Command 15
6. Transition Considerations 16
   6.1. Transition Phase 1 - Features 17
   6.2. Transition Phase 2 - Storage 18
   6.3. Transition Phase 3 - Enforcement 18
7. IANA Considerations 19
   7.1. XML Namespace 19
   7.2. EPP Extension Registry 19
8. Security Considerations 19
9. References 20
   9.1. Normative References 20

9.2. Informative References 21

Acknowledgements 21

Authors' Addresses 21

1. Introduction

The Extensible Provisioning Protocol (EPP) [RFC5730] defines the use of authorization information to authorize a transfer of an EPP object, such as a domain name, between clients that are referred to as "registrars". The authorization information is object specific and has been defined in "Extensible Provisioning Protocol (EPP) Domain Name Mapping" [RFC5731] and "Extensible Provisioning Protocol (EPP) Contact Mapping" [RFC5733] as password-based authorization information. Other authorization mechanisms can be used, but in practice the password-based authorization information has been used at the time of object creation, managed with the object update, and used to authorize an object transfer request. What has not been considered is the security of the authorization information, which includes the complexity of the authorization information, the Time To Live (TTL) of the authorization information, and where and how the authorization information is stored.

The current/original lifecycle for authorization information involves long-term storage of encrypted (not hashed) passwords, which presents a significant latent risk of password compromise and is not consistent with current best practices. The mechanisms in this document provide a way to avoid long-term password storage entirely and to only require the storage of hashed (not retrievable) passwords instead of encrypted passwords.

This document defines an operational practice, using the EPP RFCs, that leverages the use of strong, random authorization information values that are short lived, not stored by the client, and stored by the server using a cryptographic hash to provide secure authorization information used for transfers. This operational practice can be used to support transfers of any EPP object, where the domain name object as defined in [RFC5731] is used in this document for illustration purposes. Elements of the practice may be used to support the secure use of the authorization information for purposes other than transfer, but any other purposes and the applicable elements are out of scope for this document.

The overall goal is to have strong, random authorization information values that are short lived and are either not stored or stored as cryptographic hash values by the non-responsible parties. In a registrant, registrar, and registry model, the registrant registers the object through the registrar to the registry. The registrant is the responsible party, and the registrar and the registry are the non-responsible parties. EPP is a protocol between the registrar and the registry, where the registrar is referred to as the "client" and the registry is referred to as the "server". The following are the elements of the operational practice and how the existing features of the EPP RFCs can be leveraged to satisfy them:

Strong Random Authorization Information: The EPP RFCs define the password-based authorization information value using an XML schema "normalizedString" type, so they don't restrict what can be used in any substantial way. This operational practice defines the recommended mechanism for creating a strong random authorization value that would be generated by the client.

Short-Lived Authorization Information: The EPP RFCs don't explicitly support short-lived authorization information or a TTL for authorization information, but there are EPP RFC features that can be leveraged to support short-lived authorization information. All of these features are compatible with the EPP RFCs, though not mandatory to implement. As stated in Section 2.6 of [RFC5731], authorization information is assigned when a domain object is created, which results in long-lived authorization information. This specification changes the nature of the authorization information from long lived to short lived. If authorization information is set only when a transfer is in process, the server needs to support an empty authorization information value on create, support setting and unsetting authorization information, and support automatically unsetting the authorization information upon a successful transfer. All of these features can be supported by the EPP RFCs.

Storing Authorization Information Securely: The EPP RFCs don't specify where and how the authorization information is stored in the client or the server, so there are no restrictions on defining an operational practice for storing the authorization information securely. The operational practice will require the client to not store the authorization information and will require the server to store the authorization information using a cryptographic hash with at least a 256-bit hash function, such as SHA-256 [FIPS-180-4], and with a per-authorization information random salt with at least 128 bits. Returning the authorization information set in an EPP info response will not be supported.

1.1. Conventions Used in This Document

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

XML [W3C.REC-xml-20081126] is case sensitive. Unless stated otherwise, XML specifications and examples provided in this document **MUST** be interpreted in the character case presented in order to develop a conforming implementation.

In examples, "C:" represents lines sent by a protocol client and "S:" represents lines returned by a protocol server. Indentation and empty space in examples are provided only to illustrate element relationships and are not a required feature of this protocol.

The examples reference XML namespace prefixes that are used for the associated XML namespaces. Implementations **MUST NOT** depend on the example XML namespaces and instead employ a proper namespace-aware XML parser and serializer to interpret and output the XML documents. The example namespace prefixes used and their associated XML namespaces include the following:

domain: urn:ietf:params:xml:ns:domain-1.0

contact: urn:ietf:params:xml:ns:contact-1.0

2. Registrant, Registrar, Registry

The EPP RFCs refer to "client" and "server", but when it comes to transfers, there are three types of actors that are involved. This document will refer to these actors as "registrant", "registrar", and "registry". [RFC8499] defines these terms formally for the Domain Name System (DNS). The terms are further described below to cover their roles as actors using the authorization information in the transfer process of any object in the registry, such as a domain name or a contact:

Registrant: [RFC8499] defines the registrant as "an individual or organization on whose behalf a name in a zone is registered by the registry." The registrant can be the owner of any object in the registry, such as a domain name or a contact. The registrant interfaces with the registrar for provisioning the objects. A transfer is coordinated by the registrant to transfer the sponsorship of the object from one registrar to another. The authorization information is meant to authenticate the registrant as the owner of the object to the non-sponsoring registrar and to authorize the transfer.

Registrar: [RFC8499] defines the registrar as "a service provider that acts as a go-between for registrants and registries." The registrar interfaces with the registrant for the provisioning of objects, such as domain names and contacts, and with the registries to satisfy the registrant's provisioning requests. A registrar may (1) directly interface with the registrant or (2) indirectly interface with the registrant, typically through one or more resellers. Implementing a transfer using secure authorization information extends through the registrar's reseller channel up to the direct interface with the registrant. The registrar's interface with the registries uses EPP. The registrar's interface with its reseller channel or the registrant is registrar specific. In the EPP RFCs, the registrar is referred to as the "client", since EPP is the protocol used between the registrar and the registry. The sponsoring registrar is the authorized registrar to manage objects on behalf of the registrant. A non-sponsoring registrar is not authorized to manage objects on behalf of the registrant. A transfer of an object's sponsorship is from one registrar, referred to as the "losing registrar", to another registrar, referred to as the "gaining registrar".

Registry: [RFC8499] defines the registry as "the administrative operation of a zone that allows registration of names within that zone." The registry typically interfaces with the registrars over EPP and generally does not interact directly with the registrant. In the EPP RFCs, the registry is referred to as the "server", since EPP is the protocol used between the registrar and the registry. The registry has a record of the sponsoring registrar for each object and provides the mechanism (over EPP) to coordinate a transfer of an object's sponsorship between registrars.

3. Signaling Client and Server Support

This document does not define a new protocol; rather, it defines an operational practice using existing EPP features, where the client and the server can signal support for the operational practice using a namespace URI in the login and greeting extension services. The namespace URI "urn:ietf:params:xml:ns:epp:secure-authinfo-transfer-1.0" is used to signal support for the operational practice. The client includes the namespace URI in an <svcExtension> <extURI> element of the <login> command [RFC5730]. The server includes the namespace URI in an <svcExtension> <extURI> element of the greeting [RFC5730].

A client that receives the namespace URI in the server's greeting extension services can expect the following supported behavior by the server:

1. Support for an empty authorization information value with a <create> command.
2. Support for unsetting authorization information with an <update> command.
3. Support for validating authorization information with an <info> command.
4. Support for not returning an indication of whether the authorization information is set or unset to the non-sponsoring registrar.
5. Support for returning an empty authorization information value to the sponsoring registrar when the authorization information is set in an info response.
6. Support for allowing the passing of a matching non-empty authorization information value to authorize a transfer.
7. Support for automatically unsetting the authorization information upon successful completion of a transfer.

A server that receives the namespace URI in the client's <login> command extension services can expect the following supported behavior by the client:

1. Support for the generation of authorization information using a secure random value.
2. Support for only setting the authorization information when a transfer is in process.

4. Secure Authorization Information

The EPP RFCs ([RFC5731] and [RFC5733]) use password-based authorization information to support transfer with the <domain:pw> element [RFC5731] and with the <contact:pw> element [RFC5733]. Other EPP objects that support password-based authorization information for transfer can use secure authorization information as defined in this document. For authorization information to be secure, it must be generated using a strong random value and have a short TTL. The security of the authorization information is defined in the following sections.

4.1. Secure Random Authorization Information

For authorization information to be secure, it **MUST** be generated using a secure random value. The authorization information is treated as a password, and the required length L of a password, rounded up to the largest whole number, is based on the size N of the set of characters and the desired entropy H, in the equation L = ROUNDUP(H / $\log_2$ N). Given a target entropy, the required length can be calculated after deciding on the set of characters that will be randomized. In accordance with current best practices and noting that the authorization information is a machine-generated value, the implementation **SHOULD** use at least 128 bits of entropy as the value of H. The lengths below are calculated using that value.

Calculation of the required length with 128 bits of entropy and with the set of all printable ASCII characters except space (0x20), which consists of the 94 characters 0x21-0x7E:

ROUNDUP(128 / $\log_2$ 94) =~ ROUNDUP(128 / 6.55) =~ ROUNDUP(19.54) = 20

Calculation of the required length with 128 bits of entropy and with the set of case-insensitive alphanumeric characters, which consists of 36 characters (a-z A-Z 0-9):

ROUNDUP(128 / $\log_2$ 36) =~ ROUNDUP(128 / 5.17) =~ ROUNDUP(24.76) = 25

The strength of the random authorization information is dependent on the random number generator. Suitably strong random number generators are available in a wide variety of implementation environments, including the interfaces listed in Sections 7.1.2 and 7.1.3 of [RFC4086]. In environments that do not provide interfaces to strong random number generators, the practices defined in [RFC4086] and Section 4.7.1 of the NIST Federal Information Processing Standards (FIPS) Publication 140-2 [FIPS-140-2] can be followed to produce random values that will be resistant to attack. (Note: FIPS 140-2 has been superseded by FIPS 140-3, but FIPS 140-3 does not contain information regarding random number generators.)

4.2. Authorization Information Time To Live (TTL)

The authorization information **SHOULD** only be set when a transfer is in process. This implies that the authorization information has a TTL by which the authorization information is cleared when the TTL expires. The EPP RFCs do not provide definitions for TTL, but since the server supports the setting and unsetting of the authorization information by the sponsoring registrar, the sponsoring registrar can apply a TTL based on client policy. The TTL client policy may be based on proprietary registrar-specific criteria, which provides for a transfer-specific TTL tuned for the particular circumstances of the transaction. The sponsoring registrar will be aware of the TTL, and the sponsoring registrar **MUST** inform the registrant of the TTL when the authorization information is provided to the registrant.

4.3. Authorization Information Storage and Transport

To protect the disclosure of the authorization information, the following requirements apply:

1. The authorization information **MUST** be stored by the registry using a strong one-way cryptographic hash with at least a 256-bit hash function, such as SHA-256 [FIPS-180-4], and with a per-authorization information random salt with at least 128 bits.
2. An empty authorization information value **MUST** be stored as an undefined value that is referred to as a "NULL" value. The representation of a NULL (undefined) value is dependent on the type of database used.
3. The authorization information **MUST NOT** be stored by the losing registrar.
4. The authorization information **MUST** only be stored by the gaining registrar as a "transient" value in support of the transfer process.
5. The plain-text version of the authorization information **MUST NOT** be written to any logs by a registrar or the registry, nor otherwise recorded where it will persist beyond the transfer process.
6. All communication that includes the authorization information **MUST** be over an encrypted channel (for example, see [RFC5734]) for EPP.
7. The registrar's interface for communicating the authorization information with the registrant **MUST** be over an authenticated and encrypted channel.

4.4. Authorization Information Matching

To support the authorization information TTL, as described in Section 4.2, the authorization information must have either a set or unset state. Authorization information that is unset is stored with a NULL (undefined) value. Based on the requirement to store the authorization information using a strong one-way cryptographic hash, as described in Section 4.3, authorization information that is set is stored with a non-NULL hashed value. The empty authorization information value is used as input in both the <create> command (Section 5.1) and the <update> command (Section 5.2) to define the unset state. The matching of the authorization information in the <info> command (Section 5.3) and the <transfer> request command (Section 5.4) is based on the following rules:

1. Any input authorization information value **MUST NOT** match an unset authorization information value. For example, in [RFC5731] the input <domain:pw>2fooBAR</domain:pw> must not match an unset authorization information value that used <domain:null/> or <domain:pw/>.
2. An empty input authorization information value **MUST NOT** match any set authorization information value.
3. A non-empty input authorization information value **MUST** be hashed and matched against the set authorization information value, which is stored using the same hash algorithm.

5. Create, Transfer, and Secure Authorization Information

To secure the transfer process using secure authorization information as described in Section 4, the client and server need to implement steps where the authorization information is set only when a transfer is actively in process and ensure that the authorization information is stored securely and transported only over secure channels. The steps for management of the authorization information for transfers include the following:

1. The registrant requests to register the object with the registrar. The registrar sends the <create> command with an empty authorization information value to the registry, as described in Section 5.1.
2. The registrant requests from the losing registrar the authorization information to provide to the gaining registrar.
3. The losing registrar generates a secure random authorization information value and sends it to the registry, as described in Section 5.2, and then provides it to the registrant.
4. The registrant provides the authorization information value to the gaining registrar.
5. The gaining registrar optionally verifies the authorization information with the <info> command to the registry, as described in Section 5.3.
6. The gaining registrar sends the transfer request with the authorization information to the registry, as described in Section 5.4.
7. If the transfer completes successfully, the registry automatically unsets the authorization information; otherwise, the losing registrar unsets the authorization information when the TTL expires; see Section 5.2.

The following sections outline the practices of the EPP commands and responses between the registrar and the registry that supports secure authorization information for transfer.

5.1. <Create> Command

For a <create> command, the registry **MUST** allow the passing of an empty authorization information value and **MAY** disallow the passing of a non-empty authorization information value. By having an empty authorization information value on create, the object is initially not involved in the transfer process. Any EPP object extension that supports setting the authorization information with an "eppcom:pwAuthInfoType" element can pass an empty authorization information value. Examples of such extensions are found in [RFC5731] and [RFC5733].

Example of passing an empty authorization information value in a domain name <create> command [RFC5731]:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:authInfo>
C:          <domain:pw/>
C:        </domain:authInfo>
C:      </domain:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example of passing an empty authorization information value in a contact <create> command [RFC5733]:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <contact:create
C:       xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
C:        <contact:id>sh8013</contact:id>
C:        <contact:postalInfo type="int">
C:          <contact:name>John Doe</contact:name>
C:          <contact:addr>
C:            <contact:city>Dulles</contact:city>
C:            <contact:cc>US</contact:cc>
C:          </contact:addr>
C:        </contact:postalInfo>
C:        <contact:email>jdoe@example.com</contact:email>
C:        <contact:authInfo>
C:          <contact:pw/>
C:        </contact:authInfo>
C:      </contact:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

5.2. <Update> Command

For an <update> command, the registry **MUST** allow the setting and unsetting of the authorization information. The registrar sets the authorization information by first generating a strong, random authorization information value, based on the information provided in Section 4.1, and setting it in the registry in the <update> command. The importance of generating strong

authorization information values cannot be overstated: secure transfers are very important to the Internet to mitigate damage in the form of theft, fraud, and other abuse. It is critical that registrars only use strong, randomly generated authorization information values.

Because of this, registries may validate the randomness of the authorization information based on the length and character set required by the registry -- for example, validating that an authorization value contains a combination of uppercase, lowercase, and non-alphanumeric characters in an attempt to assess the strength of the value and returning an EPP error result of 2202 ("Invalid authorization information") [RFC5730] if the check fails.

Such checks are, by their nature, heuristic and imperfect, and may identify well-chosen authorization information values as being not sufficiently strong. Registrars, therefore, must be prepared for an error response of 2202 and respond by generating a new value and trying again, possibly more than once.

Often, the registrar has the "clientTransferProhibited" status set, so to start the transfer process, the "clientTransferProhibited" status needs to be removed, and the strong, random authorization information value needs to be set. The registrar **MUST** define a TTL, as described in Section 4.2, and if the TTL expires, the registrar will unset the authorization information.

Example of removing the "clientTransferProhibited" status and setting the authorization information in a domain name <update> command [RFC5731]:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:rem>
C:          <domain:status s="clientTransferProhibited"/>
C:        </domain:rem>
C:        <domain:chg>
C:          <domain:authInfo>
C:            <domain:pw>LuQ7Bu@w9?%+_HK3cayg$55$LSft3MPP
C:            </domain:pw>
C:          </domain:authInfo>
C:        </domain:chg>
C:      </domain:update>
C:    </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

When the registrar-defined TTL expires, the sponsoring registrar **MUST** cancel the transfer process by unsetting the authorization information value and **MAY** add back statuses like the "clientTransferProhibited" status. Any EPP object extension that supports setting the authorization information with an "eppcom:pwAuthInfoType" element can pass an empty authorization information value. Examples of such extensions are found in [RFC5731] and

[RFC5733]. Setting an empty authorization information value unsets the authorization information. [RFC5731] supports an explicit mechanism of unsetting the authorization information, by passing the <domain:null> authorization information value. The registry **MUST** support unsetting the authorization information by accepting an empty authorization information value and accepting an explicit unset element if it is supported by the object extension.

Example of adding the "clientTransferProhibited" status and unsetting the authorization information explicitly in a domain name <update> command [RFC5731]:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:add>
C:          <domain:status s="clientTransferProhibited"/>
C:        </domain:add>
C:        <domain:chg>
C:          <domain:authInfo>
C:            <domain:null/>
C:          </domain:authInfo>
C:        </domain:chg>
C:      </domain:update>
C:    </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

Example of unsetting the authorization information with an empty authorization information value in a domain name <update> command [RFC5731]:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:add>
C:          <domain:status s="clientTransferProhibited"/>
C:        </domain:add>
C:        <domain:chg>
C:          <domain:authInfo>
C:            <domain:pw/>
C:          </domain:authInfo>
C:        </domain:chg>
C:      </domain:update>
C:    </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

Example of unsetting the authorization information with an empty authorization information value in a contact <update> command [RFC5733]:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <contact:update
C:        xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
C:        <contact:id>sh8013</contact:id>
C:        <contact:chg>
C:          <contact:authInfo>
C:            <contact:pw/>
C:          </contact:authInfo>
C:        </contact:chg>
C:      </contact:update>
C:    </update>
C:    <clTRID>ABC-12345-XYZ</clTRID>
C:  </command>
C:</epp>
```

5.3. <Info> Command and Response

For an <info> command, the registry **MUST** allow the passing of a non-empty authorization information value for verification. The gaining registrar can pre-verify the authorization information provided by the registrant prior to submitting the transfer request with the use of the <info> command. The registry compares the hash of the passed authorization information

with the hashed authorization information value stored for the object. When the authorization information is not set or the passed authorization information does not match the previously set value, the registry **MUST** return an EPP error result code of 2202 [RFC5730].

Example of passing a non-empty authorization information value in a domain name <info> command [RFC5731] to verify the authorization information value:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <domain:info
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.com</domain:name>
C:        <domain:authInfo>
C:          <domain:pw>LuQ7Bu@w9?%+_HK3cayg$55$LSft3MPP
C:          </domain:pw>
C:        </domain:authInfo>
C:      </domain:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

The info response in object extensions, such as those defined in [RFC5731] and [RFC5733], **MUST NOT** include the optional authorization information element with a non-empty authorization value. The authorization information is stored as a hash in the registry, so returning the plain-text authorization information is not possible, unless valid plain-text authorization information is passed in the <info> command. The registry **MUST NOT** return any indication of whether the authorization information is set or unset to the non-sponsoring registrar by not returning the authorization information element in the response. The registry **MAY** return an indication to the sponsoring registrar that the authorization information is set by using an empty authorization information value. The registry **MAY** return an indication to the sponsoring registrar that the authorization information is unset by not returning the authorization information element.

Example of returning an empty authorization information value in a domain name info response [RFC5731] to indicate to the sponsoring registrar that the authorization information is set:

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <domain:infData
S:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:        <domain:name>example.com</domain:name>
S:        <domain:roid>EXAMPLE1-REP</domain:roid>
S:        <domain:status s="ok"/>
S:        <domain:clID>ClientX</domain:clID>
S:        <domain:authInfo>
S:          <domain:pw/>
S:        </domain:authInfo>
S:      </domain:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5.4. <Transfer> Request Command

For a <transfer> request command, the registry **MUST** allow the passing of a non-empty authorization information value to authorize a transfer. The registry compares the hash of the passed authorization information with the hashed authorization information value stored for the object. When the authorization information is not set or the passed authorization information does not match the previously set value, the registry **MUST** return an EPP error result code of 2202 [RFC5730]. Whether the transfer occurs immediately or is pending is up to server policy. When the transfer occurs immediately, the registry **MUST** return the EPP success result code of 1000 ("Command completed successfully") [RFC5730], and when the transfer is pending, the registry **MUST** return the EPP success result code of 1001 ("Command completed successfully; action pending"). The losing registrar **MUST** be informed of a successful transfer request using an EPP <poll> message.

Example of passing a non-empty authorization information value in a domain name <transfer> request command [RFC5731] to authorize the transfer:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <transfer op="request">
C:      <domain:transfer
C:        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example1.com</domain:name>
C:        <domain:authInfo>
C:          <domain:pw>LuQ7Bu@w9?%+_HK3cayg$55$LSft3MPP
C:          </domain:pw>
C:        </domain:authInfo>
C:      </domain:transfer>
C:    </transfer>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Upon successful completion of the transfer, the registry **MUST** automatically unset the authorization information. If the transfer request is not submitted within the TTL (Section 4.2) or the transfer is canceled or rejected, the registrar **MUST** unset the authorization information, as described in Section 5.2.

6. Transition Considerations

The goal of the transition considerations is to minimize the impact to the registrars in supporting the Secure Authorization Information Model defined in this document by supporting incremental transition steps. The transition steps are dependent on the starting point of the registry. Registries may have different starting points, since some of the elements of the Secure Authorization Information Model may have already been implemented. The considerations assume a starting point, referred to as the "Classic Authorization Information Model", which incorporates the following steps for management of the authorization information for transfers:

1. The registrant requests to register the object with the registrar. The registrar sends the <create> command, with a non-empty authorization information value, to the registry. The registry stores the authorization information as an encrypted value and requires a non-empty authorization information value for the life of the object. The registrar may store the long-lived authorization information.
2. At the time of transfer, the registrant requests from the losing registrar the authorization information to provide to the gaining registrar.
3. The losing registrar retrieves the locally stored authorization information or queries the registry for authorization information using the <info> command, and provides it to the registrant. If the registry is queried, the authorization information is decrypted and the plain-text authorization information is returned in the info response to the registrar.
4. The registrant provides the authorization information value to the gaining registrar.

5. The gaining registrar optionally verifies the authorization information with the <info> command to the registry, by passing the authorization information in the <info> command to the registry.
6. The gaining registrar sends the transfer request with the authorization information to the registry. The registry will decrypt the stored authorization information to compare to the passed authorization information.
7. If the transfer completes successfully, the authorization information is not touched by the registry and may be updated by the gaining registrar using the <update> command. If the transfer is canceled or rejected, the losing registrar may reset the authorization information using the <update> command.

The gaps between the Classic Authorization Information Model and the Secure Authorization Information Model include the following:

1. Registry requirement for a non-empty authorization information value on create and for the life of the object versus the authorization information not being set on create and only being set when a transfer is in process.
2. Registry not allowing the authorization information to be unset versus providing support for unsetting the authorization information in the <update> command.
3. Registry storing the authorization information as an encrypted value versus a hashed value.
4. Registry support for returning the authorization information versus not returning the authorization information in the info response.
5. Registry not touching the authorization information versus the registry automatically unsetting the authorization information upon a successful transfer.
6. Registry possibly validating a shorter authorization information value using password complexity rules versus validating the randomness of a longer authorization information value that meets the required bits of entropy.

The transition can be handled in the three phases defined in Sections 6.1, 6.2, and 6.3.

6.1. Transition Phase 1 - Features

The goal of "Transition Phase 1 - Features" is to implement the needed features in EPP so that the registrar can optionally implement the Secure Authorization Information Model. The features to implement are broken out by the commands and responses below:

<Create> Command: Change the <create> command to make the authorization information optional, by allowing both a non-empty value and an empty value. This enables a registrar to optionally create objects without an authorization information value, as described in Section 5.1.

<Update> Command: Change the <update> command to allow unsetting the authorization information, as described in Section 5.2. This enables the registrar to optionally unset the authorization information when the TTL expires or when the transfer is canceled or rejected.

Transfer Approve Command and Transfer Auto-Approve: Change the transfer approve command and the transfer auto-approve to automatically unset the authorization information. This sets the default state of the object to not have the authorization information set. The registrar implementing the Secure Authorization Information Model will not set the authorization information for an inbound transfer, and the registrar implementing the Classic Authorization Information Model will set the new authorization information upon a successful transfer.

Info Response: Change the <info> command to not return the authorization information in the info response, as described in Section 5.3. This sets up the implementation of "Transition Phase 2 - Storage" (Section 6.2), since the dependency on returning the authorization information in the info response will be removed. This feature is the only one that is not an optional change to the registrar, and this change could potentially break the client, so it's recommended that the registry provide notice of the change.

<Info> Command and Transfer Request: Change the <info> command and the transfer request to ensure that a registrar cannot get an indication that the authorization information is set or not set by returning the EPP error result code of 2202 when comparing a passed authorization to a non-matching set authorization information value or an unset value.

6.2. Transition Phase 2 - Storage

The goal of "Transition Phase 2 - Storage" is to transition the registry to use hashed authorization information instead of encrypted authorization information. There is no direct impact on the registrars, since the only visible indication that the authorization information has been hashed is that the set authorization information is not returned in the info response, as addressed in "Transition Phase 1 - Features" (Section 6.1). Transitioning the authorization information storage includes the following three steps:

Hash New Authorization Information Values: Change the <create> command and the <update> command to hash rather than encrypt the authorization information.

Support Comparison against Encrypted or Hashed Authorization Information: Change the <info> command and the <transfer> request command to be able to compare a passed authorization information value with either a hashed or encrypted authorization information value. This requires that the stored values be self-identifying as being in hashed or encrypted form.

Hash Existing Encrypted Authorization Information Values: Convert the encrypted authorization information values stored in the registry database to hashed values. This update will not be visible to the registrar. The conversion can be done over a period of time, depending on registry policy.

6.3. Transition Phase 3 - Enforcement

The goal of "Transition Phase 3 - Enforcement" is to complete the implementation of the Secure Authorization Information Model, by enforcing the following:

Disallow Authorization Information on <Create> Command: Change the <create> command to not allow the passing of a non-empty authorization information value. This behavior could potentially break the client, so it's recommended that the registry provide notice of this change.

Validate the Strong Random Authorization Information: Change the validation of the authorization information in the <update> command to ensure at least 128 bits of entropy.

7. IANA Considerations

7.1. XML Namespace

This document uses URNs to describe XML namespaces conforming to the registry mechanism described in [RFC3688]. IANA has assigned the following URI in the "ns" subregistry within the "IETF XML Registry" for secure authorization information for the transfer namespace:

URI: urn:ietf:params:xml:ns:epp:secure-authinfo-transfer-1.0
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

7.2. EPP Extension Registry

IANA has registered the EPP operational practice described in this document in the "Extensions for the Extensible Provisioning Protocol (EPP)" registry as defined in [RFC7451]. The details of the registration are as follows:

Name of Extension: "Extensible Provisioning Protocol (EPP) Secure Authorization Information for Transfer"
Document status: Standards Track
Reference: RFC 9154
Registrant Name and Email Address: IESG (iesg@ietf.org)
TLDs: Any
IPR Disclosure: None
Status: Active
Notes: None

8. Security Considerations

Section 4.1 defines the use of a secure random value for the generation of authorization information. The client **SHOULD** choose a length and set of characters that result in at least 128 bits of entropy.

Section 4.2 defines the use of an authorization information TTL. The registrar **SHOULD** only set the authorization information during the transfer process by setting the authorization information at the start of the transfer process and unsetting the authorization information at

the end of the transfer process. The TTL value is left up to registrar policy, and the sponsoring registrar **MUST** inform the registrant of the TTL when providing the authorization information to the registrant.

Section 4.3 defines the storage and transport of authorization information. The losing registrar **MUST NOT** store the authorization information and the gaining registrar **MUST** only store the authorization information as a "transient" value during the transfer process, where the authorization information **MUST NOT** be stored after the end of the transfer process. The registry **MUST** store the authorization information using a one-way cryptographic hash of at least 256 bits and with a per-authorization information random salt with at least 128 bits. All communication that includes the authorization information **MUST** be over an encrypted channel. The plain-text authorization information **MUST NOT** be written to any logs by the registrar or the registry.

Section 4.4 defines the matching of the authorization information values. The registry stores an unset authorization information value as a NULL (undefined) value to ensure that an empty input authorization information value never matches it. The method used to define a NULL (undefined) value is database specific.

9. References

9.1. 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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <https://www.rfc-editor.org/info/rfc5731>.

[RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.17487/RFC5733, August 2009, <https://www.rfc-editor.org/info/rfc5733>.

[RFC5734] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Transport over TCP", STD 69, RFC 5734, DOI 10.17487/RFC5734, August 2009, <https://www.rfc-editor.org/info/rfc5734>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

[W3C.REC-xml-20081126] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <https://www.w3.org/TR/2008/REC-xml-20081126>.

9.2. Informative References

[FIPS-140-2] National Institute of Standards and Technology, U.S. Department of Commerce, "NIST Federal Information Processing Standards (FIPS) Publication 140-2", DOI 10.6028/NIST.FIPS.140-2, May 2001, <https://csrc.nist.gov/publications/detail/fips/140/2/final>.

[FIPS-180-4] National Institute of Standards and Technology, U.S. Department of Commerce, "Secure Hash Standard, NIST Federal Information Processing Standards (FIPS) Publication 180-4", DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://csrc.nist.gov/publications/detail/fips/180/4/final>.

[RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <https://www.rfc-editor.org/info/rfc7451>.

Acknowledgements

The authors wish to thank the following persons for their feedback and suggestions: Michael Bauland, Martin Casanova, Scott Hollenbeck, Benjamin Kaduk, Jody Kolker, Barry Leiba, Patrick Mevzek, Matthew Pozun, Srikanth Veeramachaneni, and Ulrich Wisser.

Authors' Addresses

James Gould
Verisign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States of America
Email: jgould@verisign.com
URI: https://www.verisign.com

Richard Wilhelm
Verisign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States of America
Email: 4rickwilhelm@gmail.com
URI: https://www.verisign.com