

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 13, 2012

S. Turner
IECA
January 10, 2012

Secure Object Delivery Protocol (SODP)
draft-turner-sodp-02.txt

Abstract

This document describes the Secure Object Delivery Protocol (SODP). SODP enables clients to obtain secured packages from a Secure Object Management System (SOMS). Packages supported by a SOMS server include but are not limited to: firmware packages [RFC4108], trust anchor (TA) packages [RFC5934], symmetric key packages [RFC6031], asymmetric key packages [RFC5958], encrypted key packages [RFC6031], public key certificate enrollment packages [RFC5272], public key certificates [RFC5280], and Certificate Revocation Lists (CRLs) [RFC5280]. Client access is ideally direct and web-based, but access via agents acting on behalf of clients is supported.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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

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

This Internet-Draft will expire on July 13, 2012.

Internet-Draft

SODP

January 10, 2012

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1	Definitions	4
1.2	Key Words	6
2.	Secure Object Management System Model	6
3.	Server	8
3.1.	Server Package Requirements	9
3.1.1.	URI: /certificates	9
3.1.2.	URI: /fullCMC	10
3.1.3.	URI: /crls	11
3.1.4.	URI: /symmetricKey	11
3.1.5.	URI: /firmware	12
3.1.6.	URI: /tamp	13
3.1.7.	Mixed Packages	13
3.2.	Server Authentication Requirements	13
4.	Client	14
4.1.	Client Package Requirements	14
4.1.1.	URI: /certificates	14
4.1.2.	URI: /fullCMC	14
4.1.3.	URI: /crls	15
4.1.4.	URI: /symmetricKey	15
4.1.5.	URI: /firmware	16
4.1.6.	URI: /tamp	17
4.1.7.	Mixed Packages	17
4.2.	Authentication Requirements	17
5.	Agents	18

6.	Universal Unique Identifiers	18
7.	Product Availability List	18
7.1.	PAL Format	20
7.2.	URIs	21
7.2.1.	URI Scheme	23

7.2.2.	URI Authority	23
7.2.3.	URI Path	24
7.2.4.	URI Query and Fragments	24
8.	SODP Transport Requirements	25
8.1.	Server Requirements	25
8.2.	Client Requirements	26
8.3.	Agent Requirements	27
9.	Message Sequences	27
9.1.	/certificates Package Types and Message Sequence	27
9.2.	/crls Package Type and Message Sequence	27
9.3.	/fullCMC Package Types and Message Sequence	28
9.4.	/symmetricKey Package Types and Message Sequences	30
9.5.	/firmware Package Type and Message Sequences	31
9.5.	/tamp Message Types and Sequence	32
10.	Cryptographic Algorithm Requirements	32
10.1.	Package Protection	32
10.2.	TLS Cipher Suites	33
10.3.	Certificates	33
11.	Security Considerations	34
12.	IANA Considerations	34
12.1.	SODP Name Space	35
12.2.	SODP Schema	35
12.3.	SODP Package Types	36
12.4.	SODP Path 1 String Values	37
13.	Acknowledgements	38
14.	References	38
14.1.	Normative References	38
14.2.	Informative References	41
Appendix A.	Example Encodings	41
Appendix B.	Change Log	42
B.1.	Changes from -01 to -02	42
B.2.	Changes from -00 to -01	42
	Authors' Addresses	43

1. Introduction

The Secure Object Delivery Protocol (SODP) enables clients to obtain secured packages from a Secure Object Management System (SOMS) server. Packages supported by a SOMS server include but are not limited to: firmware packages [[RFC4108](#)], Trust Anchor (TA) packages [[RFC5934](#)], symmetric key packages [[RFC6031](#)], asymmetric key packages [[RFC5958](#)], encrypted key packages [[RFC6033](#)], public key certificate management packages [[RFC5272](#)], public key certificates [[RFC5280](#)], and Certificate Revocation Lists (CRLs) [[RFC5280](#)]. Some are the end product of multiple client-server interactions while some are simply made by or on behalf of the SOMS for the client. All packages are at

Turner

Expires July 13, 2012

[Page 3]

Internet-Draft

SODP

January 10, 2012

a minimum digitally signed and some may be encrypted. Client interactions with a SOMS server is via the HyperText Transfer Protocol (HTTP) 1.1 [[RFC2616](#)] over Transport Security Layer (TLS) [[RFC5246](#)] (HTTPS) [[RFC2818](#)]. Clients can directly access the SOMS or an agent can act on the client's behalf.

A SOMS server provides access to packages based on Universal Resource Identifiers (URIs) [[RFC3986](#)]. The Enrollment over Secure Transport (EST) [[ID.pkix-est](#)] provides a framework that this document expands. In addition, this document provides a mechanism, called a Product Availability List (PAL), by which a client can be informed of the location of all of its packages. Processing the packages in the provided order ensures the client is provided the packages necessary for it to operate.

1.1 Definitions

Terms are defined below as they are used in this document. They are presented in alphabetical order.

Agent: An entity that performs functions on behalf of a client.

Asymmetric Key Package: A package that includes an asymmetric key content type [[RFC5958](#)].

Centrally-Generated Asymmetric Keys: Server-generated asymmetric key pairs. Server provides both private key and public key certificate in an asymmetric key package [[ID.pkix-cmc-serverkeygeneration](#)].

Client: A cryptographic device/module that ultimately consumes and uses the SOMS' products to enable communications.

Encrypted Key Package: A package that includes an encrypted key content type [[RFC6032](#)].

Firmware Package: A package that contains a firmware content type [[RFC4108](#)] [RFC5911].

NOTE: [[RFC4108](#)] defines the semantics for the firmware content type's fields. [[RFC5911](#)] provides the 2002 ASN.1 definitions. Henceforth when referring to Firmware Packages only [[RFC4108](#)] will be used.

Locally-Generated Asymmetric Key: Client-generated asymmetric key pairs. Client provides the public key to the server for enrollment.

Notifications: Special entries in a PAL that tell the client to initiate an action at the server (e.g., begin a certificate rekey).

Package: An object that contains one or more CMS content types. At a minimum, all packages are protected using the CMS [[RFC5652](#)] [RFC6268] SignedData structure. There are numerous types of packages: Asymmetric Key, Certificate Revocation Lists, Public Key Certificate Management, Encrypted Key, Firmware, Public Key Certificates, and Symmetric Key Packages. The notable exception to the CMS SignedData rule are public key certificates and CRLs; these are not always protected by CMS because they've already been signed by the Certification Authority (CA). They can however be included in a package that is protected using SignedData, which is often referred to as a degenerate CMS or "certs-only" message [[RFC5751](#)] [RFC6268].

NOTE: This document does not define any packages; they are all defined elsewhere.

NOTE: [[RFC5652](#)] defines the semantics for the SignedData content type's fields. [[RFC6268](#)] provides the 2008 ASN.1 definitions. Henceforth when referring to CMS SignedData only [[RFC5652](#)] will be used and when referring to "certs-only" messages only [[RFC5751](#)] will be used.

Product Availability List (PAL): A PAL is an XML (Extensible Markup

Language) [[XML](#)] file that furnishes information for packages that are currently available and authorized for retrieval by a client or agent.

NOTE: The exception to this are Notifications.

Public Key Certificate Management Packages: A package that contains a Public Key Infrastructure (PKI) Data or a PKI Data Response content type [[RFC5272](#)] [[RFC5912](#)] [[RFC5273](#)] [[RFC5274](#)].

NOTE: [[RFC5272](#)] defines the semantics for the PKI Data and Data Response content types' fields. [[RFC5912](#)] provides the 2002 ASN.1 definitions. [[RFC5273](#)] defines the HTTP binding for CMC. [[RFC5274](#)] defines the support requirements for CMC. Henceforth when referring to CMC only [[RFC5272](#)] will be used.

Secure Object Management System (SOMS): A set of one or more components that is designed to protect, manage, and distribute cryptographic products. In this document, cryptographic products are referred to as packages.

Source Authority: A source authority is trusted by clients to generate particular package types. Clients determine this by validating the digital signature on the package back to a Trust Anchor (TA).

Symmetric Key Package: A package that contains a symmetric key content type [[RFC6031](#)].

Trust Anchor (TA): From [[RFC5914](#)], a TA contains a public key that is used to validate digital signatures. In this document, a TA represents an authoritative entity via a public key and associated data. The public key is used to verify digital signatures and the associated data is used to constrain the types of information for which the TA is authoritative. A relying party uses TAs to determine if a digitally signed object is valid by verifying a digital signature using the TA's public key, and by enforcing the constraints expressed in the associated data for the TA.

[1.2](#) Key Words

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 [\[RFC2119\]](#).

[2.](#) Secure Object Management System Model

Figure 1 depicts the SODP model. It is comprised of three entities: the SOMS server, one or more clients, and agents acting on behalf of clients. Server-to-client and server-to-agent protocol interactions are in-scope; agent-to-client protocol interactions are out-of-scope.

<==> IP-Based Protocol Profile (in scope)
<- -> Client-Specified Access Protocol (out of scope)

```
+-----+
|       |
|  Secure  |
|  Object  |
| Management |
|       |
+-----+
```

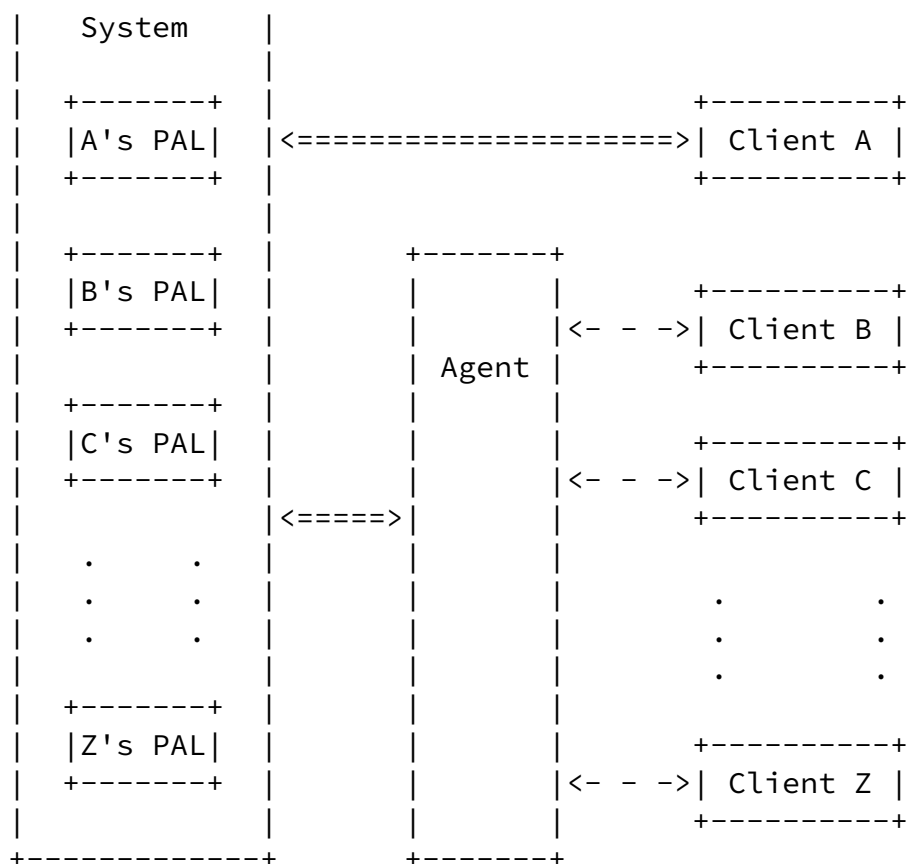


Figure 1 - SODP Model

While the SOMS is viewed as being a single entity, operationally the issuance of different packages can be assigned to different authorities within the SOMS. These authorities are referred to as source authorities. A source authority is trusted by clients to generate particular package types. Entities validate their source authorities when validating the digital signature(s) in/on packages. That is, when a client retrieves a package the client ensures that the signatures in/on the package validate to an installed trust anchor (TA).

Figure 2 depicts an example ladder diagram for a protocol flow. The first step is to establish a mutually authenticated HTTPS connection between the client/agent and a SOMS server. The client then requests their PAL, which is an XML file that contains URIs for client

packages, from the SOMS server via an HTTPS GET request. The server

replies with the client's PAL via an HTTPS GET response. Once a client has successfully downloaded their PAL, it will process it to retrieve the included packages(s). The processing provided will depend on the PAL entry. [Section 3](#) details the SOMS-package requirements, [Section 4](#) details clients-package requirements, and [Section 5](#) details agent-package requirements. [Section 7](#) details the PAL requirements.

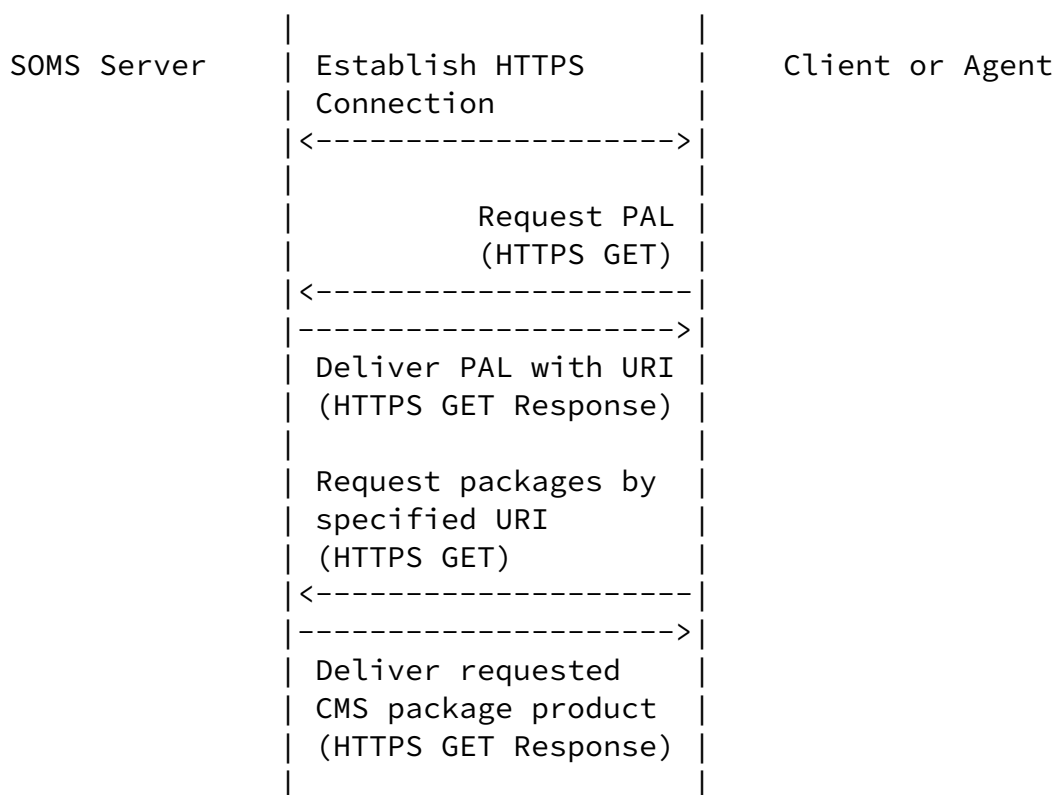


Figure 2 - Example SODP Message Sequence

3. Server

The SODP is the interface to the SOMS that clients use to access SOMS-provided packages on a SOMS server. The internal components of the SOMS server and their interactions are out-of-scope. However, if a SOMS provides all of the packages, it will need the capability to package trust anchors (TAs), generate and package symmetric keys, package firmware, generate and package asymmetric keys, issue and package public key certificates, and issue and package CRLs. It will also need to generate and receive packages, which includes generating and verifying digital signatures on packages as well as encrypting and decrypting of packages. Additionally, it will need a repository to store information about clients and to store the client's packages.

Prior to interaction with the SOMS, the client needs to be registered with the SOMS. During registration, information about the client is collected and an identity is assigned, which at a minimum includes an identifier. There are only two registration requirements:

- o The information collected MUST include a permanent identifier that is used to identify the client throughout its lifecycle. See [Section 6](#).
- o The SOMS MUST ensure that the client identity is SOMS-unique. That is, the collection of data that comprises the client identity MUST NOT match another client served by the SOMS.

The SOMS server MUST support the generation of a PAL. The SOMS server MUST support access to client packages directly and through a PAL.

After the client is registered the client is issued a public key certificate [[RFC5280](#)].

NOTE: 1) The process for delivering the certificate to the client is out-of-scope; 2) the format and protocol for communicating the registration data is out-of-scope; and 3) the client need not contribute to or respond to the supplied identity information.

The remainder of this section describes the SOMS server package requirements and the SOMS server authentication requirements.

[3.1](#). Server Package Requirements

SOMS servers provide packages based on a URI. EST [[ID.pkix-est](#)] defines 4 URIs. This document makes use of two defined therein (i.e, /certificates, /simpleEnroll, /simpleReEnroll, and /fullCMC) but this document specifies additional URIs: /crls, /symmetricKey, /firmware, and /tamp. There are no additional requirements on /simpleEnroll and /simpleReEnroll.

NOTE: I've suggested /crls be added to EST so if it's adopted there it'll come out of here.

NOTE: I also suggested collapsing /simpleEnroll and /simpleEnrollment in to one URI.

[3.1.1](#). URI: /certificates

NOTE: The URI defined in EST -02 is /CACerts. As you might guess

the URI is used to distribute CA certificates. I think it ought to be expanded to include arbitrary EE certificates. Therefore at

some point the name of this URI might change based on whether or not the comment is accepted.

NOTE: I've also suggested that the /certificates (or whatever it ends up being called) supports "certs-only" packages. If it's adopted, then I'd take it out of here.

Servers use the /certificates URI to deliver CA certificates that a client needs to validate package contents as well as EE certificates that the client might need when communicating with other clients. See section 5.1 of [[ID.pkix-est](#)].

Certificates can also be in a "certs-only" package [[RFC5751](#)], which is a CMS SignedData with no content just certificates. The SOMS server MUST support the "certs-only" package by replying to the HTTPS GET request with an HTTPS GET response that includes an HTTP Content-Type of application/pkcs7-mime and a file type of .p7c, [[RFC5751](#)].

Servers MUST support the /certificates URI.

[3.1.2](#). URI: /fullCMC

NOTE: A comment has been entered to rename this URI in EST as /fullEnrollment to line up better with /simpleEnrollment. Just saying the name might change.

Servers use the /fullCMC URI to perform public key certificate management using the Certificate Management over CMS (CMC) [[RFC5272](#)] with Full PKI Request and Responses. SOMS servers MUST use the HTTP binding defined in [[RFC5273](#)]. As a result, SOMS servers must support:

- o Receiving HTTPS POST requests with an HTTP Content-Type of application/pkcs7-mime that contains a ct-PKIData encapsulated in an ct-signed-data content type
- o Generating HTTPS POST response with an HTTP Content-Type of application/pkcs7-mime that contains a ct-PKIResponse encapsulated in an ct-signed-data content type.

SOMS servers MUST support locally-generated keys. SOMS servers that support centrally-generated keys MUST support [ID.pkix-cmc-serverkeygeneration].

The server MUST support validating signatures on /fullCMC packages back to a TA [[RFC5652](#)] [RFC5280].

Servers MUST support the /fullCMC URI.

[3.1.3](#). URI: /crls

Servers use the /crls URI to distribute CRLs, which are necessary to validate certification paths back to a TA. Clients are however free to elect to obtain the CRLs that they rely on from sources other than the SOMS (e.g., a local directory).

CRLs are offered in the form, or forms, produced by the responsible Certification Authority (CA). The form of the CRL is transparent to the SOMS. CAs may choose to publish compact versions of CRLs (e.g., partitioned CRLs) that are compatible with a disadvantaged client within the overall subscriber population.

Servers MUST support receiving HTTPS GET requests and MUST support generating HTTPS GET responses for CRLs. SOMS servers MUST support returning CRLs:

- o The HTTP Content-Type [[RFC2616](#)] is application/pkix-crl and a file type of .crl [[RFC2585](#)], which contains a single CRL.
- o The HTTP Content-Type [[RFC2616](#)] is application/pkcs7-mime and a file type of .p7c [[RFC5751](#)], which contains either a single CRL or multiple CRLs.

The PAL provided to a client will always contain a URI for the most current version of each CRL needed to verify the packages in the form used by the particular client. The SOMS will not list CRLs that a client does not need or cannot use. Based on its capabilities, the freshness of currently held CRLs, and the circumstances, the client will determine whether it needs to download each offered CRL.

Servers MUST support the /crls URI.

[3.1.4.](#) URI: /symmetricKey

Servers use the /symmetricKey URI to distribute symmetric key packages to clients and to receive receipts and errors about the distributed symmetric key packages. Symmetric key packages are defined in [\[RFC6031\]](#). A symmetric key package can contain one or more symmetric keys. It also can contain attributes that apply to one or more keys. The SOMS server MUST encapsulate the ct-symmetric-key-package content type in a ct-signed-data content type [\[RFC5652\]](#).

Distribution of the symmetric key packages requires that these keys be disclosed only to the client and to not to anyone else. The key packages need to be enveloped. The encrypted key package [\[RFC6032\]](#) supports encrypting key packages in one of three ways: with key exchange algorithms (i.e., using EnvelopedData), with previously

distributed symmetric algorithms (i.e., using EncryptedData), and with authenticated-encryption algorithms (i.e., using AuthEnvelopedData). The server MUST support the ct-encrypted-key-package content type and as specified in [\[RFC6032\]](#) the EnvelopedData choice must be supported (i.e., support ct-enveloped-data). The server MUST also encapsulate the ct-encrypted-key-package in a ct-signed-data content type.

Servers MUST support processing HTTPS GET requests and MUST support generating HTTPS GET responses for key packages. The server MUST support key packages by replying to the HTTPS GET request with an HTTPS GET response that includes an HTTP Content-Type of application/pkcs7-mime and a file type of .p7m, as specified in [\[RFC5751\]](#).

Servers MUST support processing HTTPS POST requests and MUST support generating HTTPS POST responses for both key package receipts and key package errors [ID.turner-ct-keypackage-receipt-n-error]. The server MUST support key package error and key package receipt packages by replying to the HTTPS GET request with an HTTPS GET response that includes an HTTP Content-Type of application/pkcs7-mime and a file type of .p7m, as specified in [\[RFC5751\]](#). The server MUST reject /symmetricKey errors and receipts whose signature fails to validate back to a TA [\[RFC5652\]](#) [\[RFC5280\]](#).

Servers MUST support the /symmetricKey URI.

[3.1.5.](#) URI: /firmware

Servers distribute object code for implementing one or more cryptographic algorithms in a cryptographic module and software to implement a communications protocol with the firmware package [[RFC4108](#)]. The server MUST support the ct-firmwarePackage content type. It MUST support receipt of the ct-firmwareLoadReceipt and ct-firmwareLoadError content types. The SOMS server MUST encapsulate the ct-firmwarePackage content type in a ct-signed-data content type [[RFC5652](#)].

Servers MUST support processing HTTPS GET requests and MUST support generating HTTPS GET responses for firmware packages. The server MUST support firmware packages by replying to the HTTPS GET request with an HTTPS GET response that includes an HTTP Content-Type of application/pkcs7-mime and a file type of .p7m, as specified in [[RFC5751](#)].

Servers MUST support HTTPS POST requests and MUST support HTTPS POST responses for firmware receipts and errors. The server MUST support firmware packages by replying to the HTTPS GET request with an HTTPS

GET response that includes an HTTP Content-Type of application/pkcs7-mime and a file type of .p7m, as specified in [[RFC5751](#)]. The server MUST reject /firmware error and receipt packages whose signature fails to validate back to a TA [[RFC5652](#)] [[RFC5280](#)].

Servers MUST support the /firmware URI.

[3.1.6.](#) URI: /tamp

The SOMS manages TAs to support validating packages with the Trust Anchor Management Protocol (TAMP) [[RFC5934](#)]. TAMP supports multiple formats for the TA. The SOMS MUST support the Certificate choice. The SOMS MUST support the HTTP binding described in [[RFC5934](#)]. As specified in [[RFC5934](#)]:

- o servers must support the tamp-update content type. And, the tamp-update must be encapsulated in a ct-signed-data content type.

- o servers must support the trust-anchor-update-confirm and tamp-error content types [[RFC5934](#)].

The server MUST reject /tamp update-confirm and error packages whose signature fail to validate back to a TA [[RFC5652](#)][RFC5280].

Servers MUST support the /tamp URI.

[3.1.7.](#) Mixed Packages

NOTE: certs-only packages allow servers to package up both certificates and CRLs in one package. Should mixed packages have their own URI? This section is obviously TBD.

To support sending multiple package types to a client, the SOMS can use the Content Collection [[RFC4073](#)] CMS content type. To allow the SOMS to apply additional attributes to the package the can use the Content With Attributes [[RFC4073](#)] CMS content type. The SOMS SHOULD support the ct-contentCollection and MAY support the ct-contentWithAttributes content type. The SOMS MUST support encapsulating these in a ct-signed-data content type.

[3.2.](#) Server Authentication Requirements

SOMS-to-client and SOMS-to-agent interactions MUST use mutual authentication, provide integrity, and optionally provide confidentiality through the use of HTTP over Transport Security Layer (HTTPS). Confidentiality for SOMS-to-client and SOMS-to-agent interactions is OPTIONAL because when confidentiality is needed the

packages are encrypted for the client. See [Section 10](#) for requirements on cryptographic suites.

The one exception to the requirement for mutual authentication is retrieval of certificates from /certificates and CRLs from /crls.

[4.](#) Client

Clients use SODP to access the SOMS. Clients need to be registered prior to interacting with the SOMS. Clients need not contribute to or respond to the supplied identity information. After registration

is completed, the client is supplied with a certificate. Prior to using this certificate, the client MUST verify the certificate back to an installed TA. The number of TAs a client supports is implementation specific, but the client MUST support at least one TA.

The remainder of this section addresses client package and client authentication requirements.

[4.1.](#) Client Package Requirements

Clients retrieve packages based on a URI. This section specifies the package requirements for clients use of the URIs: /certificates, /crls, /fullCMC, /symmetricKey, /firmware, and /tamp.

[4.1.1.](#) URI: /certificates

NOTE: The URI defined in EST -02 is /CACerts. As you might guess the URI is used to distribute CA certificates. I think it ought to be expanded to include arbitrary EE certificates. Therefore at some point the name of this URI might change based on whether or not the comment is accepted.

Clients use the /certificates URI to retrieve CA certificates needed to validate package contents as well as other EE certificates that the client might need. See section 5.1 of [[ID.pkix-est](#)].

Certificates can also be in a "certs-only" package [[RFC5751](#)], which is a CMS SignedData with no content just certificates. The client MUST support the "certs-only" package by generating an HTTPS GET request and receiving an HTTPS GET response that includes an HTTP Content-Type of application/pkcs7-mime and a file type of .p7c, as specified in [[RFC5751](#)].

Clients SHOULD support the /certificates URI.

[4.1.2.](#) URI: /fullCMC

NOTE: A comment has been entered to rename this URI in EST as /fullEnrollment to line up better with /simpleEnrollment. Just saying the name might change.

Clients use the /fullCMC URI when they use the Full PKI Request and Full PKI Responses [[RFC5272](#)]. Clients MUST use the HTTP binding defined in [[RFC5273](#)]. As a result, clients must support:

- o Generating HTTPS POST requests with an HTTP Content-Type of application/pkcs7-mime that contains a ct-PKIData encapsulated in an ct-signed-data content type
- o Processing HTTPS POST response with an HTTP Content-Type of application/pkcs7-mime that contains a ct-PKIResponse encapsulated in an ct-signed-data content type

Clients that support centrally-generated keys MUST support [ID.pkix-cmc-serverkeygeneration].

Clients MUST reject /fullCMC packages whose signature fail to validate back to a TA [[RFC5652](#)][RFC5280].

Client MUST support the /fullCMC URI.

[4.1.3.](#) URI: /crls

Clients use the /crls URI to retrieve CRLs, which are necessary to validate certification paths back to a TA. Clients are however free to elect to obtain the CRLs that they rely on from sources other than the SOMS (e.g., a local directory).

Clients MUST support generating HTTPS GET requests and MUST support processing HTTPS GET responses for CRLs. Client MUST support receiving CRLs with:

- o The HTTP Content-Type [[RFC2616](#)] is application/pkix-crl and a file type of .crl [[RFC2585](#)], which contains a single CRL.
- o The HTTP Content-Type [[RFC2616](#)] is application/pkcs7-mime and a file type of .p7c [[RFC5751](#)], which contains either a single CRL or multiple CRLs.

Clients SHOULD support the /crls URI unless the client relies on an alternate mechanism to retrieve CRLs.

[4.1.4.](#) URI: /symmetricKey

Clients use the /symmetricKey URI to retrieve key packages and to

return receipts and errors about the distributed symmetric key packages.

Clients MUST support symmetric key packages defined in [\[RFC6031\]](#). Clients MUST support a ct-signed-data content type [\[RFC5652\]](#) encapsulating a ct-symmetric-key-package content type. Clients MUST reject ct-symmetric-key-package content types that are not encapsulated in a ct-signed-data content type. Clients MUST reject symmetric key packages whose signature fail to validate back to a TA [\[RFC5652\]](#) [\[RFC5280\]](#).

Clients MUST support the encrypted key package [\[RFC6032\]](#) and the EnvelopedData choice must be supported (i.e., support ct-enveloped-data). Clients MUST support a ct-signed-data content type [\[RFC5652\]](#) encapsulating a ct-encrypted-key-package content type. Clients MUST reject ct-encrypted-key-package content types that are not encapsulated in a ct-signed-data content type. Clients MUST reject encrypted key packages whose signature fail to validate back to a TA [\[RFC5652\]](#) [\[RFC5280\]](#).

Clients MUST support generating HTTPS GET requests and MUST support processing HTTPS GET responses for key packages. Clients MUST support key packages by submitting HTTPS GET requests and receiving HTTPS GET response that includes an HTTP Content-Type of application/pkcs7-mime and a file type of .p7m, as specified in [\[RFC5751\]](#).

Clients MUST support generating HTTPS POST requests and MUST support processing HTTPS POST responses for both key package receipts and key package errors [\[ID.turner-ct-keypackage-receipt-n-error\]](#). Clients MUST support key package error and key package receipt packages by generating the HTTPS GET request and processing the HTTPS GET response that includes an HTTP Content-Type of application/pkcs7-mime and a file type of .p7m, as specified in [\[RFC5751\]](#).

Clients MAY support the /symmetricKey URI.

[4.1.5.](#) URI: /firmware

Clients retrieve object code for implementing one or more cryptographic algorithms in a cryptographic module and software to implement a communications protocol with the Firmware package [\[RFC4108\]](#). Clients MUST support processing the ct-firmwarePackage content type. Clients MUST support generating the ct-firmwareLoadReceipt and ct-firmwareLoadError content types. Clients MUST support the ct-firmwarePackage content type encapsulated in a ct-signed-data content type [\[RFC5652\]](#). Clients MUST reject ct-

firmware-package content-type whose signature fails to validate back

Internet-Draft

SODP

January 10, 2012

to a TA [[RFC5652](#)] [RFC5280].

Clients MUST support generating HTTPS GET requests and MUST support processing HTTPS GET responses for firmware packages. Clients MUST support firmware packages by generating to the HTTPS GET request and processing an HTTPS GET response that includes an HTTP Content-Type of application/pkcs7-mime and a file type of .p7m, as specified in [[RFC5751](#)].

Clients MUST support generating HTTPS POST requests and MUST support processing HTTPS POST responses for firmware receipts and errors. Clients MUST support firmware packages by generating to the HTTPS GET request with an HTTPS GET response that includes an HTTP Content-Type of application/pkcs7-mime and a file type of .p7m, as specified in [[RFC5751](#)]. Clients MUST reject /firmware error and receipt packages whose signature fails to validate back to a TA [[RFC5652](#)] [RFC5280].

Clients MAY support the /firmware URI.

[4.1.6](#). URI: /tamp

Client's TAs are managed with the Trust Anchor Management Protocol (TAMP) [[RFC5934](#)]. TAMP supports multiple formats for the TA. Clients MUST support the Certificate choice. Clients MUST support the HTTP binding described in [[RFC5934](#)]. As specified in [[RFC5934](#)]:

- o Clients must support the tamp-update content type [[RFC5934](#)]. And, the tamp-update must be encapsulated in a ct-signed-data content type.
- o Clients must support the trust-anchor-update-confirm and tamp-error content types [[RFC5934](#)].

Clients MUST reject /tamp packages whose signature fail to validate back to a TA [[RFC5652](#)] [RFC5280].

Clients MAY support the /tamp URI.

[4.1.7](#) Mixed Packages

Client MAY support for the ct-contentCollection [[RFC4073](#)] and the ct-contentWithAttributes [[RFC4073](#)] content types.

[4.2.](#) Authentication Requirements

Clients MUST use client-side certificate-based TLS authentication when communicating with the server. Clients MUST support processing certificate-based TLS authentication. The exception to this rule is

Turner

Expires July 13, 2012

[Page 17]

Internet-Draft

SODP

January 10, 2012

when the client uses the /certificates and /crls URIs, clients need not use client authentication during these exchanges.

[5.](#) Agents

Agents act on behalf of the client. The requirements for agents are identical to those of clients with the following exceptions:

- o Agents MUST support PAL processing.
- o Agents MUST support the /symmetricKey URI.
- o Agents MUST support the /firmware URI.
- o Agents MUST support the /tamp URI.

Agent Authentication requirements go here.

[6.](#) Universal Unique Identifiers

The Universal Unique Identifier (UUID) is a permanent identifier that is used to identify the client throughout its lifecycle.

Certificates include the UUID with the Hardware Module Name from [[RFC4108](#)] in the Subject Alternative Name extension [[RFC5280](#)]. The hardware module name form is an hwType (an object identifier) and hwSerialNumber (octet string). The hwSerialNumber is a Universal Unique Identifier (UUID) [[RFC4122](#)]. The server, clients, and agents SHOULD support UUIDs at least 16 octets in length.

[7.](#) Product Availability List

The PAL provides clients with:

- o Advertisements for available packages that can be retrieved from the server;

- o Notifications for public key certificate management, and;
- o Advertisement for another PAL.

An example PAL is provided in Figure 3. The explanation of the fields is explained in the subsequent text and sections.

```
<?xml version="1.0"encoding="utf-8" ?>
<pal>
  <message>
    <type>TBD</type>
    <date>0000000000000000</date>
    <size>1996</size>
    <info>https://www.example.com/certificates/12</info>
  </message>
```

Turner

Expires July 13, 2012

[Page 18]

Internet-Draft

SODP

January 10, 2012

```
<message>
  <type>0100</type>
  <date>0000000000000000</date>
  <size>0</size>
  <info>DN of subject</info>
</message>
<message>
  <type>TBD</type>
  <date>0000000000000000</date>
  <size>2390</size>
  <info>https://www.example.com/symmetricKey/100</info>
</message>
<message>
  <type>0001</type>
  <date>0000000000000000</date>
  <size>0</size>
  <info>https://www.example.com/symmetricKey/12345</info>
</message>
</pal>
```

Figure 3 – Example PAL

PAL processing by clients is OPTIONAL, yet RECOMMENDED. PALs MUST use the application/xml media type [[RFC3023](#)]. PAL retrieval can be performed by a client or by an agent that is assisting the client. Agents that service clients which do not process PALs, MUST process

the PAL on behalf of the client. The agent MUST retrieve and process the PAL from the SOMS as well as the packages advertised within the PAL. Once delivered to the agent, the agent MUST provide the package to the target client in an implementation specific manner. The method of delivery of the package to the target client may or may not implement a PAL type distribution mechanism.

When a client or agent requests a PAL, the server dynamically assembles a PAL based on the current information and packages it has for the requesting client or agent. The server relies on the knowledge of the requesting client's ESN, in order to amass the proper list of items. PALs can contain zero (0) or more entries for a package type.

An order of precedence for PAL offerings is based on the following rationale:

- o /certificates and /crls packages are the most important because they support validation decisions on certificates used to sign and encrypt other listed PAL items.
- o /fullCMC packages items are next in importance, since they can

impact an certificate used by the device to sign CMS content or a certificate to establish keys for encrypting content exchanged with the client.

- * A client engaged in a certificate management SHOULD accept and process CA-provided transactions as soon as possible to avoid undue delays that might lead to protocol failure.
- o /symmetricKey, /firmware, /tamp packages containing keys and other types of products are last. Precedence SHOULD be given to packages that the client has not previously downloaded. The items listed in a PAL may not identify all of the packages available for a device. This can be for any of the following reasons:
 - * The server may temporarily withhold some outstanding PAL items to simplify client processing.
 - * /fullCMC PAL entries linked to a near-real-time CA device

protocol (i.e., not staged through an agent) will be limited to one-at-a-time.

- * If a CA has more than one certificate ready to begin a certificate management protocol with a client, the server will provide a notice for one at a time. Pending notices will be serviced in order of the earliest date when the certificate will be used.
- * The SOMS will complete a certificate management activity for one certificate, before beginning the process for another. At most one pending certificate management transaction will be advertised in the PAL at a time.
- o A PAL is limited to a maximum of thirty-two entries. If more than thirty-two entries are available for the client, additional PALs will be identified in the last entry of the PAL. The first PAL in the chain is identified as the Initial PAL.
- o Packages will be removed when their contents are superseded or at the direction of a server administrator.

The remainder of this section describes the PAL format and its use of URIs.

[7.1.](#) PAL Format

The PAL furnishes information for SOMS packages that are currently available and authorized for retrieval by a client or an agent. The

initially offered PAL, will contain anywhere from zero to thirty-two XML-encoded PAL entries following the XML Header. The PAL's XML schema can be found in [Section 12](#). Each PAL entry is composed of the following four REQUIRED elements:

- o The <type> element uniquely identifies each package defined within this specification that a client may retrieve from server with a 4-digit field. The Package Types are defined in [Section 9](#) and registered in [Section 12](#).
- o The <date> element is a 14-character field that contains either:

- o The date and time (expressed as Generalized Time: YYYY-MM-DDTHH:MM:SSZ) that the client last successfully downloaded the identified package from the server, or
- o 0001-01-01T00:00:00 (i.e., 0), if
 - o There is no indication the device has successfully loaded the identified package, or
 - o The PAL entry corresponds to a notification or pointer to a next PAL.
- o The <size> element indicates the size of the package. If the entry is for a notification, this element will be populated with a zero character (i.e., "0" without the quotes). Otherwise, it indicates the size of the identified package in bytes.
- o The <info> element provides either a Distinguished Name (DN) or a URI of where the identified package can be retrieved. When the entry is a notification, the subcomponent is a DN that identifies a certificate that is the subject of the notification.

When more than thirty-two PAL entries are available, an additional PAL is advertised in the thirty second PAL entry. The additional PAL will have between one and thirty-two PAL entries.

When the <date> element is not zero (i.e., 0001-01-01T00:00:00) it MUST be represented in a form that matches the dateTime production in "canonical representation" [[XMLSCHEMA](#)]. Implementations SHOULD NOT rely on time resolution finer than milliseconds and MUST NOT generate time instants that specify leap seconds.

[7.2.](#) URIs

A client that supports the PAL will use URIs to obtain both the packages they need from the SOMS, and to post device information the

SOMS requires. Clients and agents that support PALs MUST be capable of using URIs [[RFC3986](#)].

In order use HTTPS GET or HTTPS POST, the client or agent needs to have a currently valid URI associated with that information. The URI

can correspond to:

- o A PAL that provides a unique URI for each package that the SOMS holds for the client and URIs identifying client actions that need to be taken, or
- o A package that the client believes is being held by the SOMS. The data may contain product, a protocol-related transaction, or a collection of packages with various contents.

When a client performs an HTTPS POST operation, the URI indicates the specific package that is targeted to process the information. A client SHALL be capable of requesting information by providing a URI in an HTTPS GET request to a connected SOMS.

A client may know, or believe they know, a specific package URI, because:

- o They discovered the URI on a PAL,
- o They are anticipating the next step in a protocol initiated by a prior URI submission, or
- o They were provided with the URI out-of-band by a human or an agent.

Clients and agents MUST be capable of accepting a URI that uniquely identifies the location of a SOMS package that is available for delivery.

Clients and agents MUST be capable of accepting a URI that identifies an action that is to be taken by the client. In order to POST information, the client or agent supplies a URI that identifies associated information to the SOMS. For example, the URI could correspond to a request to initiate, furnish intermediate results for, or conclude a certificate management protocol.

Regardless of whether an HTTPS GET or HTTPS POST request is being made, URI components have consistent definitions and usage requirements. These are specified in the following subsections. Figure 4 provides a view of the URI components:

scheme://Authority/Path/query|fragment

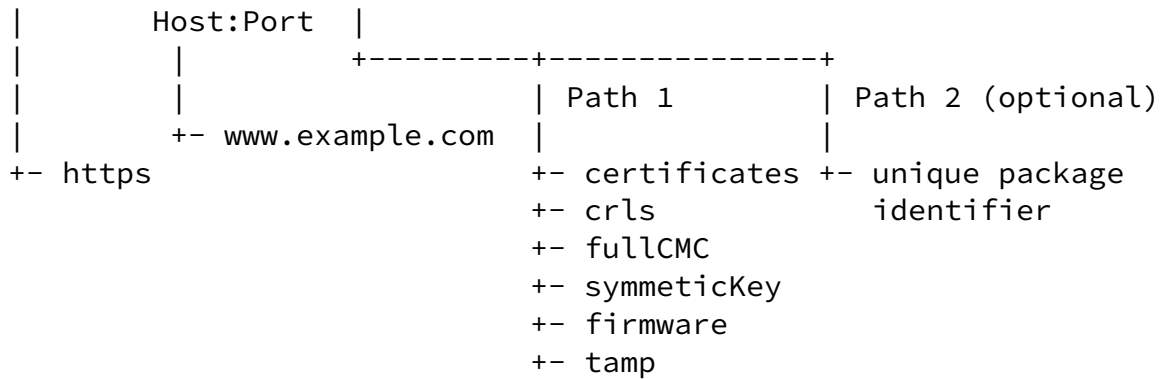


Figure 4 - PAL URI Components

[7.2.1.](#) URI Scheme

All HTTPS GET and POST requests and responses MUST use "https" as the scheme [[RFC2818](#)]. All processing of scheme data will be case-insensitive as required in [[RFC3986](#)].

PALs that do not specify "https" as the URI scheme for every PAL entry MUST be rejected.

[7.2.2.](#) URI Authority

The authority component of a URI identifies the server that the client is requesting the specific SOMS Service from. The authority component is in the form of a host name and an optional port number. The host name identifies the HTTPS server by name, and the port number identifies the HTTPS server port that will service the request. Inclusion of the port number is OPTIONAL, as the scheme component MUST always be "https".

Clients and agents that access servers are configured with the applicable registered name(s) or corresponding IP address(es) of the server with which they may establish a connection to.

When generating a URI, the server SHALL populate the Authority Component of the URI with the registered name of the target server. See Sections [7.2.3](#) and [12](#).

When generating a URI, clients and agents SHALL populate the Authority Component of the URI with the registered name of the target server.

Clients and agents SHALL reject the delivery of a received PAL, if any URI Authority Component contains a registered name that does not correspond to the connected server.

Internet-Draft

SODP

January 10, 2012

[7.2.3.](#) URI Path

The Path component of a URI identifies a resource that can be retrieved from, or a location that information can be posted to, at the SOMS. Path components are presented in the hierarchical form of SOMS Service Identifier followed by a Product Identifier. They adhere to the rules for path-absolute parsing as defined in [\[RFC3986\]](#).

Service Identifiers that constitute the first path (aka Path 1) segment in a URI received or generated by a device are:
/certificates, /crls, /fullCMC, /symmetricKey, /firmware, /tamp.

The Package Type (aka Path 2), when present, is always the second path segment. It is formatted as a four digit integer and represents the unique identifier the server has associated with the package to be retrieved. Package types are included in the Package Type registry found in [Section 12](#). The Product Identifier is only present in the URIs that will be included in HTTPS GET requests to obtain a package.

The Package Type is not included in:

- o The URI a client uses to obtain the initial PAL,
- o The URI portion of /certificates and /crls PAL entries or an entry the server uses to point to other PALs beyond the initial PAL,
- o The /fullCMC URIs that a SOMS server uses to provide the client notification for a suggested action, and
- o URIs that a client provides as a part of an HTTPS POST requests.

When generating a URI, clients SHALL populate the first Path component of the URI with one of the URIs defined by this specification. When generating a URI for the inclusion in a HTTPS POST operation, a client SHALL only populate the first Path component of the URI (i.e., the client does not include the package type). When generating a URI for the inclusion in a HTTPS GET operation for the initial PAL, a client SHALL only populate the first Path component of the URI (i.e., the client does not include the package

type). A client SHALL reject the delivery of any PAL received that contains a URI with the second path component not equal to an integer.

[7.2.4.](#) URI Query and Fragments

Servers do not use Query and Fragment elements. Servers MUST omit query and fragment components from PALs. Servers SHOULD reject the delivery of any PAL that contains a URI with a query or fragment components.

Query and Fragments are not supported by clients in the processing of received URIs, or in the generation of URIs. Clients and agents SHOULD reject the delivery of any PAL that contains a URI with a query or fragment component. When generating a URI, clients and agents MUST NOT populate the URI with any query or fragment components.

[8.](#) SODP Transport Requirements

This section provides the requirements for SODP interactions.

[8.1.](#) Server Requirements

The server MUST support processing HTTPS GET and POST requests and generating HTTPS GET and HTTPS POST responses [[RFC2818](#)]. TLS 1.2 [[RFC5246](#)][[RFC6176](#)] MUST be implemented in conjunction with HTTPS. To ensure only authorized clients and agents access the SOMS, the SOMS MUST support authentication with both client-side certificates and username/password. See [Section 10](#) for cipher suite requirements.

When the server receives and processes an HTTPS GET or POST request from a client, it will provide a response. HTTP responses include status information and may include a message body, when a request is successfully processed. The status information provided in responses to client requests will be restricted to the three-digit HTTP status code.

HTTP response status codes fall into five general classes (where the class is indicated by the first digit of the code).

- o Informational - The SOMS will not make use of the Informational class of status codes. Protocol switches and continued client processing are not expected.
- o Success - The SOMS will return this class when the GET results in the requested information being returned or the POST action is successfully completed.
- o Redirection - The SOMS will not make use of the Redirection class of status codes. The SOMS will not ask a client to take further action to fulfill a request.
- o Client Error - The SOMS will return this class when they cannot

fulfill the requested GET or POST because of a client error.

- o Server Error - The SOMS may return this class, when a valid POST or GET request was received, but the SOMS cannot fulfill the request for other reasons.

8.2. Client Requirements

Clients MUST support generating HTTPS GET and POST requests and receiving HTTPS GET and POST responses [[RFC2818](#)]. TLS 1.2 [[RFC5246](#)][[RFC6176](#)] MUST be implemented in conjunction with HTTPS. Clients MUST support client-side certificate authentication when connecting to the server. See [Section 10](#) for cipher suite requirements.

If a client receives an HTTPS response with an Informational or Redirection class status code, it SHALL interpret the response as a request failure and terminate its session with the SOMS.

When an Informational or Redirection class status code is received, a client MAY, if configured for an alternate SOMS server, terminate the current session and attempt to connect with an alternate SOMS.

If a client receives an HTTPS response with a Success class status code, it SHALL continue to process the response to determine the outcome of an HTTPS POST request or to use the information contained in the included package.

If a client receives an HTTP response with a Client Error class status code, it SHALL abandon the desired action and not repeat the same request to the same SOMS during the connection session.

The client can provide additional processing of Client Error class status codes for a given request; however, this is out-of-scope of this document.

A client can attempt other (different) HTTP requests after a request that failed with a Client Error class status code. However, the client incorporate a means to limit the number of consecutive requests that fail for any reason in a given connection session with the SOMS.

If a client receives an HTTPS response with a Server Error class status code, it SHOULD either:

- o Reattempt the request after a non-deterministic delay, or
- o Attempt the request with a different server.

[8.3.](#) Agent Requirements

Agent requirements are identical to those for clients with one exception and that is that agents MUST support either agent-side certificate authentication or username/password when connecting to the SOMS.

[9.](#) Message Sequences

This section depicts package types when using a PAL.

NOTE: Package types are not defined for packages that a client posts to a server: Key Package Receipts, Key Package Errors, Firmware Package Receipts, and Key Package Errors.

[9.1.](#) /certificates Package Types and Message Sequence

The /certificates URI is used to distribute certificates. The package types are defined as follows:

Package	Package
---------	---------

Type	
TBD	X.509 CA Public Key Certificate
TBD	X.509 EE Public Key Certificate

An example PAL entry for a /certificates package is as follows:

```
<package>
  <type>TBD</type>
  <date>0000000000000000</date>
  <size>1996</size>
  <info>https://www.example.com/certificates/mycert.cer</info>
</package>
```

The package type TBD indicates the message is an X.509 CA Public Key Certificate. The date and time indicates that the package has not been downloaded. The package size indicates the size of the package and the additional info element provides a link to the CA Public Key Certificate.

The message sequence is identical to Figure 2.

9.2. /crls Package Type and Message Sequence

The /crls URI is used to distribute CRLs. The package types are defined as follows:

Package Type	Package
TBD	Root ARL
TBD	non-Root CRL

An example PAL entry for a /crls package is as follows:

```
<package>
  <type>TBD</type>
  <date>0000000000000000</date>
  <size>1996</size>
  <info>https://www.example.com/crls/Root.crl</info>
</package>
```

The package type TBD indicates the package is a Root CRL. The date and time indicates that the package has not been downloaded. The package size indicates the size of the package and the additional info element provides a link to the Root CRL.

The message sequence is identical to Figure 2.

9.3. /fullCMC Package Types and Message Sequence

The /fullCMC URI is used to distribute notifications (i.e., start rekey) and to manage public key certificates. The package types are defined as follows:

Package Type	Package
-----	-----
0100	Certificate Rekey Notification
N/A	DS Certificate Rekey Transaction One
TBD	DS Certificate Rekey Transaction Two (Success)
TBD	DS Certificate Rekey Transaction Two (Failure)
N/A	KE Certificate Issuance Transaction One
TBD	KE Certificate Issuance Transaction Two (Success)
TBD	KE Certificate Issuance Transaction Two (Failure)

An example PAL entry for a /fullCMC package notification is as follows:

```
<package>
  <type>0100</type>
  <date>0000000000000000</date>
  <size>1996</size>
  <info>DN of DS certificate</info>
</package>
```

The package type TBD indicates the package is a Certificate Rekey Notification. The date and time indicates that the package has not been downloaded. The package size indicates the size of the package and the additional info element provides a link to the rekey notification. The info element tells the client which certificate to request a rekey for by the DN.

The message sequence for certificate rekey and issuance is a two-step process. The initial step is client/agent retrieval of the PAL and then retrieval of a notification for a certificate rekey. Step two is the client/agent posting of the CMC package followed by the certificate request response (success or failure) from the server. Prior to each interaction with the server, the client/agent authenticates itself with the server. The two steps are depicted in Figures 5 and 6.

Step 1

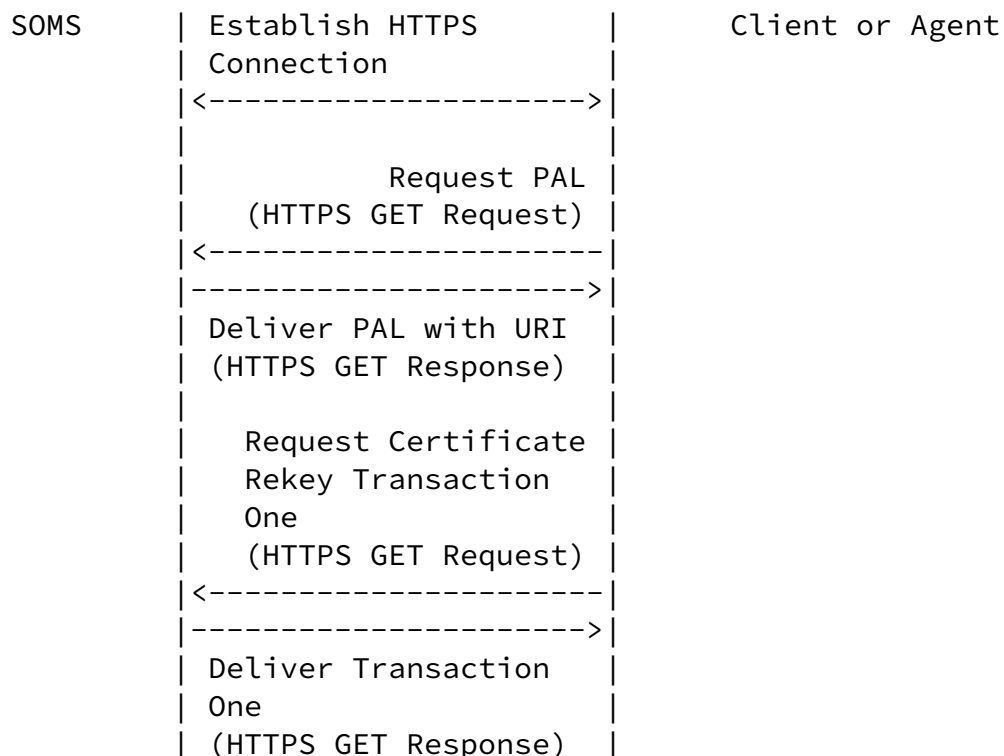


Figure 5 – SODP Certificate Management Service
Message Sequence – Step 1

Step 2

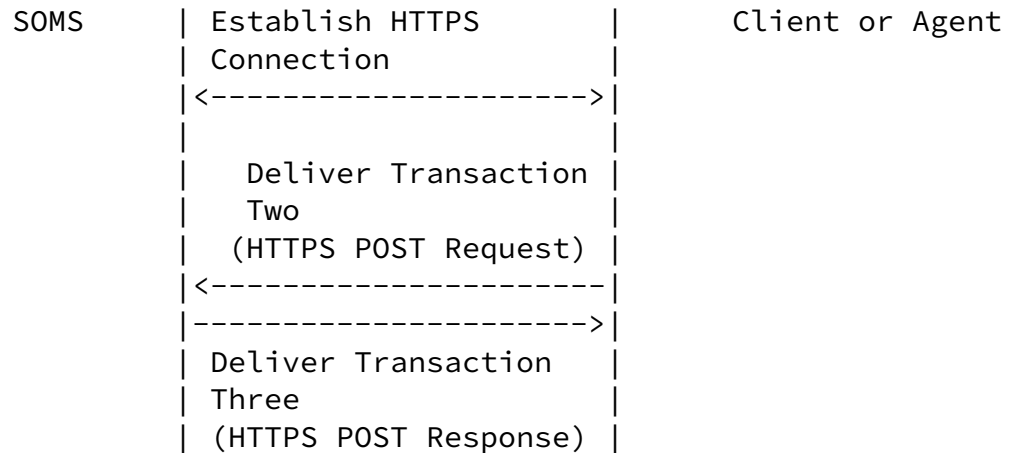


Figure 6 - SODP Certificate Management Service Message Sequence Step 2

9.4. /symmetricKey Package Types and Message Sequences

The /symmetricKey URI is used to distribute symmetric and centrally-generated asymmetric key packages. The package types are defined as follows:

Package Type	Package
-----	-----
TBD	Symmetric Key Package

An example PAL entry for a /symmetricKey package is as follows:

```
<package>
  <type>TBD</type>
  <date>0000000000000000</date>
  <size>1996</size>
  <info>https://www.example.com/symmetricKey/symmtrickey1</info>
</package>
```

The package type TBD indicates the message is a symmetric key. The date and time indicates that the package has not been downloaded. The package size indicates the size of the package and the additional info element provides a link to the symmetric key. The info element indicates the location of the package.

The sequence for both symmetric key and asymmetric key packages is identical, shown in Figure 2. The client or agent connects to the SOMS, retrieves their PAL, and the requests the package from the URI provided in the additional info component.

Internet-Draft

SODP

January 10, 2012

Error and receipt message sequence is as shown in Figure 7.

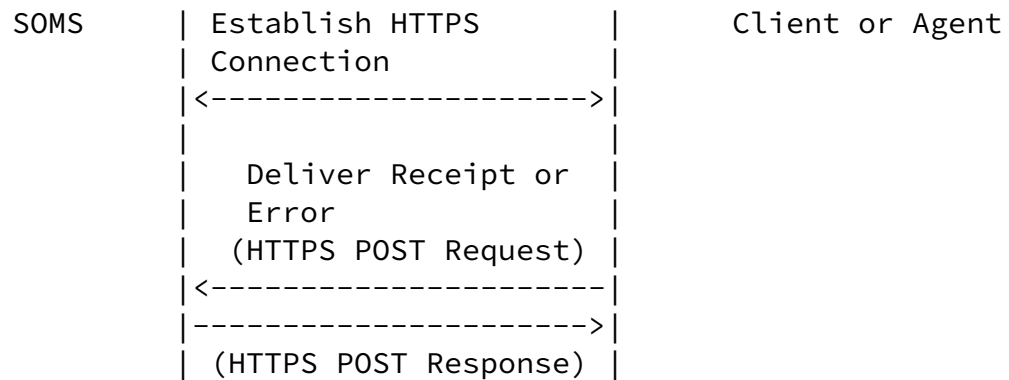


Figure 7 - SODP Certificate Management Service
Message Sequence Step 2

9.5. /firmware Package Type and Message Sequences

The /firmware URI is used to distribute firmware packages. The package types are defined as follows:

Package Type	Package
-----	-----
TBD	Firmware Package

An example PAL entry for a /firmware package is as follows:

```
<package>
  <type>TBD</type>
  <date>0000000000000000</date>
  <size>2056</size>
  <info>https://www.example.com/firmware/1234</info>
</package>
```

The message type TBD indicates the message is a firmware package. The date and time indicates that the package has not been downloaded. The message size indicates the size of the package and the additional info element provides a link to the symmetric key.

The sequence for firmware packages is identical, as shown in Figure 2. The client or agent connects to the SOMS, retrieves their PAL,

and the requests the package from the URI provided in the additional info component.

Errors and receipts message sequence are as shown in Figure 7.

[9.5.](#) /tamp Message Types and Sequence

The /tamp URI is used to distribute TAMP packages. The message types are defined as follows:

Message Type	Package
-----	-----
TBD	TAMP Package

An example PAL entry for a distribution package is as follows:

```
<package>
  <type>TBD</type>
  <date>0000000000000000</date>
  <size>2056</size>
  <info>https://www.example.com/tamp/1234</info>
</package>
```

The package type TBD indicates the message is a TAMP package. The date and time indicates that the package has not been downloaded. The package size indicates the size of the package and the additional info element provides a link to the TAMP package.

The sequence for TAMP packages is identical, as shown in Figure 2. The client or agent connects to the SOMS, retrieves their PAL, and the requests the package from the URI provided in the additional info component.

Errors and receipts message sequence are as shown in Figure 7.

[10.](#) Cryptographic Algorithm Requirements

This section defines the cryptographic algorithm requirements for SODP. There are three types: package protection requirements, TLS cipher suites, and certificate requirements.

10.1. Package Protection

For [[RFC5958](#)] algorithm requirements see [[RFC5959](#)][RFC6162].

For [[RFC6031](#)] algorithm requirements see [[RFC6160](#)].

For [[RFC6032](#)] algorithm requirements see [[RFC6033](#)][RFC6161].

NOTE: The "cert-only" package does not have algorithm requirements because no cryptographic operations are performed while generating this package (i.e., that is there is no signature applied to the

message - the signature is already on the certificates and/or CRLs included in the package).

For [[RFC4108](#)] algorithm requirements see [[RFC6160](#)].

NOTE: The algorithm requirements for [[RFC4108](#)] are for symmetric keys, but equally apply to firmware packages. Note the only required algorithm is RSA for generating the SignedData that encapsulates the firmware package.

For [[RFC5934](#)] algorithm requirements see [[RFC6160](#)].

NOTE: The algorithm requirements for [[RFC4108](#)] are for symmetric keys, but equally apply to tamp packages. Note the only required algorithm is RSA for generating the SignedData that encapsulates the tamp package.

For [[RFC5272](#)] algorithm requirements see [[RFC5274](#)].

10.2. TLS Cipher Suites

The following requirements apply to servers, clients, and agents:

- o Cipher suites supported MUST include: "TLS_RSA_WITH_", "TLS_DH_", "TLS_DHE_", and "TLS_ECDH_".
- o Cipher suites that include "anon" MUST NOT be used. These suites

do not support mutual authentication.

- o Cipher suite that include "EXPORT" and "DES" MUST NOT be used. These ciphers do not offer a sufficient level of protection; 40-bit crypto in '11 doesn't cut the mustard and the use of DES is deprecated.
- o When confidentiality is supported (recall that is optional), the "AES_128" ciphers MUST be supported and "AES_256" cipher SHOULD be supported.
- o Cipher suites that include "SHA256" MUST be supported and "SHA384" SHOULD be supported.

[10.3.](#) Certificates

Client, agents, and servers MUST support certificate path validation on all packages and HTTPS sessions [[RFC5280](#)].

Turner

Expires July 13, 2012

[Page 33]

Internet-Draft

SODP

January 10, 2012

To support the algorithm requirements in [Section 10.1](#), the following are the public key certificate requirements, which align with [[RFC5959](#)], [[RFC6033](#)], [[RFC6160](#)], [[RFC6161](#)], and [[RFC6162](#)]:

"If an implementation supports RSA, RSASSA-PSS, DSA, RSAES-OAEP, or Diffie-Hellman, then it MUST support key lengths from 1024-bit to 2048-bit, inclusive. If an implementation supports ECDSA or ECDH, then it MUST support keys on P-256."

[11.](#) Security Considerations

TO DO: Expand this section!

This document relies on many other specifications. For IP and TCP security considerations see [[RFC791](#)], [[RFC793](#)], and [[RFC2460](#)]; for HTTP, HTTPS, and TLS security considerations see [[RFC2616](#)], [[RFC2818](#)], and [[RFC5246](#)]; for URI security considerations see [[RFC3986](#)]; for content type security considerations see [[RFC4073](#)], [[RFC4108](#)], [[RFC5272](#)], [[RFC5652](#)], [[RFC5751](#)], [[RFC5958](#)], [[RFC5934](#)], [[RFC6031](#)], and [[RFC6032](#)]; for certificate security considerations see [[RFC5280](#)], [[RFC5480](#)], and [[RFC6010](#)], and; for algorithm security considerations see [[RFC5959](#)], [[RFC6033](#)], [[RFC6160](#)], [[RFC6161](#)],

[[RFC6162](#)].

TO DO: Probably more references are needed above for algorithms based on what gets added in [Section 10.1](#).

It is critical that the SODS encrypt symmetric keys and centrally-generated asymmetric private keys for the end client. Failure to encrypt these keys will allow intermediaries to intercept the key and eavesdrop and/or impersonate the client.

When packages are encrypted, the source of the package must randomly generate package-encryption keys. Also, the generation of public/private signature key pairs relies on random numbers. The use of inadequate pseudo-random number generators (PRNGs) to generate cryptographic keys can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys, searching the resulting small set of possibilities, rather than brute-force searching the whole key space.

The generation of quality random numbers is difficult. [[RFC4086](#)] offers important guidance in this area.

[12.](#) IANA Considerations

IANA is requested to perform four registrations: SODP Name Space, SODP XML Schema, SODP Message Types, and SODP URI String Types.

Turner

Expires July 13, 2012

[Page 34]

Internet-Draft

SODP

January 10, 2012

[12.1.](#) SODP Name Space

This section registers a new XML namespace, "urn:ietf:params:xml:ns:TBD" per the guidelines in [[RFC3688](#)]:

TO DO: Fill in TBDs after name space is registered.

URI: urn:ietf:params:xml:ns:TBD

Registrant Contact: Sean Turner (turners@ieca.com)

XML:

BEGIN

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
```

```

    <head>
      <title>SODP Messages</title>
    </head>
    <body>
      <h1>Namespace for SODP Messages</h1>
      <h2>urn:ietf:params:xml:ns:sodp</h2>
      <p>See RFC TBD</p>
    </body>
  </html>
END

```

[12.2.](#) SODP Schema

This section registers an XML schema as per the guidelines in [\[RFC3688\]](#).

TO DO: Fill in TBDs after the name space is registered.

URI: urn:ietf:params:xml:schema:sodp

Registrant Contact: Sean Turner turners@ieca.com

```

XML:
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
  xmlns:sodp="urn:ietf:params:xml:ns:sodp"
  targetNamespace="urn:ietf:params:xml:ns:sodp"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="1.0">

  <!-- ===== Element Declarations ===== -->

  <xsd:element name="pal" type="sodp:PalType" />

```

```

  <!-- ===== Complex Data Element Type Definitions ===== -->

  <xsd:complexType name="PalType">
    <xsd:sequence>
      <xsd:element name="message" type="sodp:SODPPackageType"
        minOccurs="0" maxOccurs="32">
      </xsd:element>
    </xsd:sequence>

```



```

</xsd:complexType>

<xsd:complexType name="SODPPackageType">
  <xsd:sequence>
    <xsd:element name="type" type="sodp:PackageType" />
    <xsd:element name="date" type="sodp:GeneralizedTimeType" />
    <xsd:element name="size" type="sodp:PackageSizeType" />
    <xsd:element name="info" type="sodp:PackageInfoType" />
  </xsd:sequence>
</xsd:complexType>

<!-- =====Simple Data Element Type Definitions ===== -->

<xsd:simpleType name="PackageType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[0-9]+" />
    <xsd:maxLength value="4" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="GeneralizedTimeType" type="xsd:string" />

<xsd:simpleType name="PackageSizeType">
  <xsd:pattern value="[0-9]+" />
  <xsd:maxLength value="19" />
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="PackageInfoType">
  <xsd:restriction base="xsd:string" />
</xsd:simpleType>

</xsd:schema>

```

[12.3.](#) SODP Package Types

This section registers the SODP Package Types. Future SODP Package Types registrations are to be subject to Specification Required, as defined in [RFC 5226](#) [[RFC5226](#)].

The registry has the following values:

Value	Package Type	Specification
0000	Reserved	This document
0001	Additional PAL value present	This document
0100	DS Rekey Notification	This document
TBD	X.509 CA Public Key Certificate	This document
TBD	X.509 EE Public Key Certificate	This document
TBD	Root CRL	This document
TBD	non-Root CRL	This document
TBD	DS Certificate Rekey Transaction Two - Success	This document
TBD	DS Certificate Rekey Transaction Two - Fail	This document
TBD	KE Certificate Issuance Transaction One	This document
TBD	KE Certificate Issuance Transaction Three - Success	This document
TBD	KE Certificate Issuance Transaction Three - Fail	This document
TBD	Symmetric Key Package	This document
TBD	Firmware Package	This document
TBD	TAMP Package	This document

[12.4.](#) SODP Path 1 String Values

This section registers SODP Path String Types as per [\[RFC3688\]](#). SODP Path 1 String Value registrations are to be subject to Specification Required, as defined in [RFC 5226](#) [\[RFC5226\]](#). The registry has the following structure:

+-----+		
	SODP Service Types	Specification
+-----+		
	certificates	This document
+-----+		
	crls	This document
+-----+		
	fullCMC	This document
+-----+		
	symmetricKey	This document
+-----+		
	firmware	This document
+-----+		
	tamp	This document
+-----+		

13. Acknowledgements

Thanks, in alphabetical order, go to Paul Hoffman, Brad McInnis, Max Pritikin, and Francois Rousseau for their helpful comments.

14. References

14.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", [RFC 2460](#), December 1998.
- [RFC2585] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", [RFC 2585](#), May 1999.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", [RFC 2616](#), June 1999.
- [RFC2818] Rescorla, E., "HTTP Over TLS", [RFC 2818](#), May 2000.
- [RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", [RFC 3023](#), January 2001.
- [RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#),

January 2004.

Turner

Expires July 13, 2012

[Page 38]

Internet-Draft

SODP

January 10, 2012

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, [RFC 3986](#), January 2005.
- [RFC4073] Housley, R., "Protecting Multiple Contents with the Cryptographic Message Syntax (CMS)", [RFC 4073](#), May 2005.
- [RFC4108] Housley, R., "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages", [RFC 4108](#), August 2005.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", [RFC 4122](#), July 2005.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#), May 2008.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), August 2008.
- [RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", [RFC 5272](#), June 2008.
- [RFC5273] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC): Transport Protocols", [RFC 5273](#), June 2008.
- [RFC5274] Schaad, J. and M. Myers, "Certificate Management Messages over CMS (CMC): Compliance Requirements", [RFC 5274](#), June 2008.
- [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](#), May 2008.
- [RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", [RFC 5480](#), March 2009.

- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, [RFC 5652](#), September 2009.
- [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", [RFC 5751](#), January 2010.
- [RFC5911] Hoffman, P. and J. Schaad, "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME", [RFC 5911](#),

Turner

Expires July 13, 2012

[Page 39]

Internet-Draft

SODP

January 10, 2012

June 2010.

- [RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", [RFC 5912](#), June 2010.
- [RFC5914] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Format", [RFC 5914](#), June 2010.
- [RFC5934] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Management Protocol (TAMP)", [RFC 5934](#), August 2010.
- [RFC5958] Turner, S., "Asymmetric Key Packages", [RFC 5958](#), August 2010.
- [RFC5959] Turner, S., "Algorithms for Asymmetric Key Package Content Type", [RFC 5959](#), August 2010.
- [RFC6010] Housley, R., Ashmore, S., and C. Wallace, "Cryptographic Message Syntax (CMS) Content Constraints Extension", [RFC 6010](#), September 2010.
- [RFC6031] Turner, S. and R. Housley, "Cryptographic Message Syntax (CMS) Symmetric Key Package Content Type", [RFC 6031](#), December 2010.
- [RFC6032] Turner, S. and R. Housley, "Cryptographic Message Syntax (CMS) Encrypted Key Package Content Type", [RFC 6032](#), December 2010.
- [RFC6033] Turner, S., "Algorithms for Cryptographic Message Syntax (CMS) Encrypted Key Package Content Type", [RFC 6033](#),

December 2010.

- [RFC6160] Turner, S., "Algorithms for Cryptographic Message Syntax (CMS) Protection of Symmetric Key Package Content Types", [RFC 6160](#), April 2011.
- [RFC6161] Turner, S., "Elliptic Curve Algorithms for Cryptographic Message Syntax (CMS) Encrypted Key Package Content Type", [RFC 6161](#), April 2011.
- [RFC6162] Turner, S., "Elliptic Curve Algorithms for Cryptographic Message Syntax (CMS) Asymmetric Key Package Content Type", [RFC 6162](#), April 2011.
- [RFC6176] Turner, S. and T. Polk, "Prohibiting Secure Sockets Layer (SSL) Version 2.0", [RFC 6176](#), March 2011.

Turner

Expires July 13, 2012

[Page 40]

Internet-Draft

SODP

January 10, 2012

- [XML] W3C, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", W3C Recommendation, November 2008, <<http://www.w3.org/TR/2006/REC-xml-20060816/>>.
- [XMLSCHEMA] Malhotra, A. and P. Biron, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041082, October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.
- [ID.pkix-est] Pritikin, M, and P. Yee, "Enrollment over Secure Transport", work-in-progress, [draft-ietf-pkix-est-00](#).
- [ID.pkix-cmc-serverkeygeneration] Schaad, J., Turner, S., and P. Timmel, "CMC Extensions: Server Key Generation", work-in-progress, [draft-ietf-pkix-cmc-serverkeygeneration-00](#).

[14.2](#). Informative References

- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", [BCP 106](#), [RFC 4086](#), June 2005.
- [RFC4880] Callas, J., Donnerhackle, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", [RFC 4880](#), November 2007.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", [RFC 5996](#), September 2010.

[XMLNS] Hollander, D., Bray, T., and A. Layman, "Namespaces in XML", World Wide Web Consortium First Edition REC-xml-names-19990114, January 1999, <<http://www.w3.org/TR/1999/REC-xml-names-19990114>>.

[Appendix A](#). Example Encodings

TO DO: Include BASE64 encodings of ASN.1 encodings of selected packages. They're a lot smaller than the ASN.1 pretty prints and there are tons of available tools to convert.

[Appendix B](#). Change Log

[[RFC EDITOR: Please delete this Appendix prior to publication.]]

[B.1](#). Changes from -01 to -02

Turner Expires July 13, 2012 [Page 41]

Internet-Draft SODP January 10, 2012

Removed the term Key Management System. SODP is about more than just keys.

Beefed up the introduction to provide more context.

Refined/Removed some definitions: ECU concept scrubbed, IA/KE Certificates, KMS, Operator, Sponsor, and Service Messages.

Added some definitions: Centrally-Generated Asymmetric Keys, Locally-Generated Asymmetric Keys, Notifications, and PAL.

Significantly shortened the description of the SODP Model, but removing the optional concept of having two sets of certificates: one for communicating with the infrastructure and another for communicating with peers.

Removed the distraction about services being instantiated by packages. Now it just talks about packages served from a URI. But,

maintained the split of Server, Client, and Agent.

Embraced the EST concept by replacing /pki with /fullCMC, adding /certificates (EST calls it /CAcerts), and recommending /crls.

Centrally-generated keys are now distributed via the /fullCMC URI. This made sense as the draft describing centrally-generated keys uses CMC.

Allowed clients to retrieve from /certificates and /crls without client authentication.

Added in Agent requirements.

Replaced ESN with UUID from [RFC 4122](#).

PAL changes: corrected the date fields in the PAL to be compliant with the XMLSCHEMA, removed 2.1 Gbyte size constraint, changed encoding from us-ascii to UTF-8.

[B.2](#). Changes from -00 to -01

Keep alive draft. Only the issue and expiry dates changed.

Authors' Addresses

Sean Turner
IECA, Inc.
3057 Nutley Street, Suite 106

Turner

Expires July 13, 2012

[Page 42]

Internet-Draft

SODP

January 10, 2012

Fairfax, VA 22031
USA

EMail: turners@ieca.com

