[MS-KPP-Diff]:

Key Provisioning Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
7/14/2016	1.0	New	Released new document.
6/1/2017	2.0	Major	Significantly changed the technical content.
6/13/2017	3.0	Major	Significantly changed the technical content.
9/15/2017	4.0	Major	Significantly changed the technical content.
12/1/2017	4.0	None	No changes to the meaning, language, or formatting of the technical content.
9/12/2018	5.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The Key Provisioning Protocol provides a mechanism for registering a set of cryptographic keys on a user and device pair.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 (Updated Section) Glossary

This document uses the following terms:

access control list (ACL): A list of access control entries (ACEs) that collectively describe the security rules for authorizing access to some resource; for example, an object or set of objects.

Active Directory: The Windows implementation of a general-purpose directory service, which uses LDAP as its primary access protocol. Active Directory stores information about a variety of objects in the network such as user accounts, computer accounts, groups, and all related credential information used by Kerberos [MS-KILE]. Active Directory is either deployed as Active Directory Domain Services (AD DS) or Active Directory Lightweight Directory Services (AD LDS), which are both described in [MS-ADOD]: Active Directory Protocols Overview.

application programming interface (API): A set of routines used by an application program to direct the performance of procedures used by the computer's operating system. Also called application program interface.

Augmented Backus-Naur Form (ABNF): A modified version of Backus-Naur Form (BNF), commonly used by Internet specifications. ABNF notation balances compactness and simplicity with reasonable representational power. ABNF differs from standard BNF in its definitions and uses of naming rules, repetition, alternatives, order-independence, and value ranges. For more information, see [RFC5234].

base64 encoding: A binary-to-text encoding scheme whereby an arbitrary sequence of bytes is converted to a sequence of printable ASCII characters, as described in [RFC4648].

Coordinated Universal Time (UTC): A high-precision atomic time standard that approximately tracks Universal Time (UT). It is the basis for legal, civil time all over the Earth. Time zones around the world are expressed as positive and negative offsets from UTC. In this role, it is also referred to as Zulu time (Z) and Greenwich Mean Time (GMT). In these specifications, all references to UTC refer to the time at UTC-0 (or GMT).

Cryptographic Message Syntax (CMS): A public standard that defines how to digitally sign, digest, authenticate, or encrypt arbitrary message content, as specified in [RFC3852].

directory: The database that stores information about objects such as users, groups, computers, printers, and the directory service that makes this information available to users and applications.

distinguished name (DN): A name that uniquely identifies an object by using the relative distinguished name (RDN) for the object, and the names of container objects and domains that contain the object. The distinguished name (DN) identifies the object and its location in a tree.

domain controller (DC): The service, running on a server, that implements Active Directory, or the server hosting this service. The service hosts the data store for objects and interoperates with other DCs to ensure that a local change to an object replicates correctly across all DCs. When Active Directory is operating as Active Directory Domain Services (AD DS), the DC contains full NC replicas of the configuration naming context (config NC), schema naming context (schema NC), and one of the domain NCs in its forest. If the AD DS DC is a global catalog server (GC server), it contains partial NC replicas of the remaining domain NCs in its

forest. For more information, see [MS-AUTHSOD] section 1.1.1.5.2 and [MS-ADTS]. When Active Directory is operating as Active Directory Lightweight Directory Services (AD LDS), several AD LDS DCs can run on one server. When Active Directory is operating as AD DS, only one AD DS DC can run on one server. However, several AD LDS DCs can coexist with one AD DS DC on one server. The AD LDS DC contains full NC replicas of the config NC and the schema NC in its forest. The domain controller is the server side of Authentication Protocol Domain Support [MS-APDS].

Domain Name System (DNS): A hierarchical, distributed database that contains mappings of domain names to various types of data, such as IP addresses. DNS enables the location of computers and services by user-friendly names, and it also enables the discovery of other information stored in the database.

globally unique identifier (GUID): A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the GUID. See also universally unique identifier (UUID).

JavaScript Object Notation (JSON): A text-based, data interchange format that is used to transmit structured data, typically in Asynchronous JavaScript + XML (AJAX) web applications, as described in [RFC7159]. The JSON format is based on the structure of ECMAScript (Jscript, JavaScript) objects.

JSON Web Token (JWT): A type of token that includes a set of claims encoded as a JSON object. For more information, see [~~IETFDRAFT-JWT~~RFC7519].

key: In cryptography, a generic term used to refer to cryptographic data that is used to initialize a cryptographic algorithm. Keys are also sometimes referred to as keying material.

object: A set of attributes, each with its associated values. For more information on objects, see [MS-ADTS] section 1 or [MS-DRSR] section 1.

private key: One of a pair of keys used in public-key cryptography. The private key is kept secret and is used to decrypt data that has been encrypted with the corresponding public key. For an introduction to this concept, see [CRYPTO] section 1.8 and [IEEE1363] section 3.1.

Representational State Transfer (REST): A class of web services that is used to transfer domain-specific data by using HTTP, without additional messaging layers or session tracking, and returns textual data, such as XML.

Transport Layer Security (TLS): A security protocol that supports confidentiality and integrity of messages in client and server applications communicating over open networks. TLS supports server and, optionally, client authentication by using X.509 certificates (as specified in [X509]). TLS is standardized in the IETF TLS working group.

UTF-8: A byte-oriented standard for encoding Unicode characters, defined in the Unicode standard. Unless specified otherwise, this term refers to the UTF-8 encoding form specified in [UNICODE5.0.0/2007] section 3.9.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents

in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[FIPS180-2] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-2, August 2002, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[ISO8601] ISO, "Data elements and interchange formats - Information interchange - Representation of dates and times", ISO 8601:2004, December 2004, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=40874

Note There is a charge to download the specification.

[MS-ADA2] Microsoft Corporation, "Active Directory Schema Attributes M".

[MS-ADA3] Microsoft Corporation, "Active Directory Schema Attributes N-Z".

[MS-ADSC] Microsoft Corporation, "Active Directory Schema Classes".

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification".

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MSKB-4022723] Microsoft Corporation, "June 27, 2017 - KB4022723 (OS Build 14393.1378)", https://support.microsoft.com/en-us/kb/4022723

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.rfc-editor.org/rfc/rfc2616.txt

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.rfc-editor.org/rfc/rfc2818.txt

[RFC4122] Leach, P., Mealling, M., and Salz, R., "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005, http://www.rfc-editor.org/rfc/rfc4122.txt

[RFC4346] Dierks, T., and Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006, http://www.ietf.org/rfc/rfc4346.txt

[RFC4514] Zeilenga, K., Ed., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4514, June 2006, http://www.ietf.org/rfc/rfc4514.txt

[RFC5280] Cooper, D., Santesson, S., Farrell, S., et al., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008, http://www.ietf.org/rfc/rfc5280.txt

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 5652, September 2009, http://www.rfc-editor.org/rfc/rfc5652.txt

[RFC5754] Turner, S., "Using SHA2 Algorithms with Cryptographic Message Syntax", RFC 5754, January 2010, https://tools.ietf.org/html/rfc5754

1.2.2 Informative References

None.

1.3 Overview

The Key Provisioning Protocol provides for registration of cryptographic keys on a user and device pair.

The Key Provisioning Protocol provides a single REST-based endpoint that returns JavaScript Object Notation (JSON)–formatted data in the response message.

This document defines and uses the following terms:

client: The entity that creates and sends the key provisioning request to the key provisioning server using the Key Provisioning Protocol.

key provisioning server: The server that implements the REST Web service that accepts and responds to key provisioning requests using the Key Provisioning Protocol.

1.4 Relationship to Other Protocols

The following figure illustrates the relationship of this protocol to other protocols.

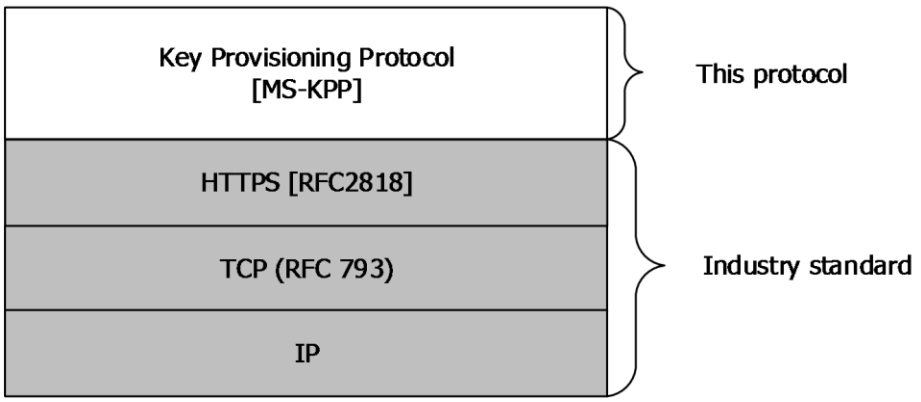

Figure 1: Protocols related to the Key Provisioning Protocol

1.5 Prerequisites/Preconditions

The Key Provisioning Protocol registers keys with User objects ([MS-ADSC] section 2.268) represented in a directory server. A server implementation of the Key Provisioning Protocol, or key provisioning server, requires a directory server.

This protocol requires that the following state changes be made to Active Directory.

1. Create an instance of the ms-DS-Device-Registration-Service-Container class ([MS-ADSC] section 2.138) in the directory.

2. Create an instance of the ms-DS-Device-Registration-Service class ([MS-ADSC] section 2.137) as a child of the container object created in the previous step. Generate a certificate-signing certificate and use it for the following attributes:

- The certificate and private key are stored in the ms-DS-Issuer-Certificates attribute ([MS-ADA2] section 2.342) of the ms-DS-Device-Registration-Service object. For details, see section 2.3.1.
- The public portion of the certificate is stored in the ms-DS-Issuer-Public-Certificates attribute ([MS-ADA2] section 2.343) of the ms-DS-Device-Registration-Service object. For details, see section 2.3.2.

3. Set the following directory access control list (ACL) entries:
    1. Grant the key provisioning server read access to the ms-DS-Device-Registration-Service object ([MS-ADSC] section 2.137).
    2. Grant the key provisioning server read/write access to ms-DS-Device objects ([MS-ADSC] section 2.135).
    3. Grant the key provisioning server read/write access to the ms-DS-Key-Credential-Link attribute ([MS-ADA2] section 2.347) on User objects.

1.6 Applicability Statement

The Key Provisioning Protocol is applicable only for requests for key provisioning.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The Key Provisioning Protocol consists of a single RESTful Web service.

- HTTPS [RFC2818] over TCP/IP [RFC2616]

The protocol operates on the following URI endpoint.

Web service	Location
Key Provisioning Service	https://<server>:<server port>/EnrollmentServer/key

All client messages to the key provisioning server MUST use Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS) and provide server authentication, which MUST use Transport Layer Security (TLS) 1.1 [RFC4346] or greater.

2.2 Common Data Types

2.2.1 HTTP Headers

This protocol accesses the HTTP headers listed in the following table.

Header	Description
client-request-id	Specifies the request identifier.
return-client-request-id	Specifies whether the key provisioning server is to include the given client-request-id in the server response.
request-id	Specifies the request identifier generated by the key provisioning server.
api-version	Specifies the application programming interface (API) version.
authorization	Specifies a JSON Web Token (JWT) used for client authorization.
accept	Specifies which media types ([RFC2616] section 3.7) are acceptable for the response.

2.2.1.1 client-request-id

The client-request-id HTTP header is optional and can appear in either the request or the response. This header is used to provide the key provisioning server with a unique request identifier, which is then used by the server to log error messages that were encountered while processing the lookup request. If present, the value of the client-request-id header MUST be a globally unique identifier (GUID) in standard string representation (see [RFC4122] section 3 for the format).

The format of the client-request-id header, in Augmented Backus-Naur Form (ABNF), is as follows.

```
String = *(%x20-7E)
```

```
client-request-id = String
```

2.2.1.2 return-client-request-id

The return-client-request-id HTTP header is optional. This header is sent in the request and is used by the key provisioning server to determine whether to return the client-specified client-request-id in the server response. If present, the value of the return-client-request-id header MUST be "true".

The format of the return-client-request-id header, in ABNF, is as follows.

```
return-client-request-id = "true"
```

2.2.1.3 request-id

The request-id HTTP header is a server-generated GUID in standard string representation (see [RFC4122] section 3 for the format). The key provisioning server SHOULD include this header in all server responses.

The format of the request-id header, in ABNF, is as follows.

```
String = *(%x20-7E)
request-id = String
```

2.2.1.4 api-version

The api-version header is an integer that indicates the API version that is expected by the client. Either this header or the api-version query parameter (section 2.2.2.1) MUST be included in all client requests.

Note The api-version header and the api-version query parameter defined in section 2.2.2.1 are mutually exclusive. The client is expected to specify an API version by using either one of these mechanisms, but not both.

The format of the api-version header, in ABNF, is as follows.

```
String = *(%x20-7E)
api-version = String
```

2.2.1.5 authorization

The authorization HTTP header is required in the request and contains a JSON Web Token (JWT) that the client passes to the key provisioning server. The JWT contains claims that the key provisioning server uses to authorize access to the relevant User object on the directory server.

The format of the authorization header, in ABNF, is as follows.

```
String = *(%x20-7E)
authorization =String
```

2.2.1.6 accept

The accept HTTP header is required in the request and specifies which media types ([RFC2616] section 3.7) are acceptable for the response. "application/json" is the only acceptable media type for the Key Provisioning Protocol.

The format of the accept header, in ABNF, is as follows.

```
accept = "application/json"
```

2.2.2 URI Parameters

The following table summarizes the set of common URI parameters defined by this protocol.

Parameter	Description
api-version	Specifies the API version.

2.2.2.1 api-version

The api-version parameter is an integer that indicates the API version that is expected by the client. Either this header or the api-version HTTP header (section 2.2.1.4) MUST be included in all client requests.

The format of the api-version parameter, in ABNF, is as follows.

```
String = *(%x20-7E)
api-version = String
```

2.2.3 Complex Types

The following table summarizes the set of complex type definitions included in this specification.

Complex Type	Description
ErrorDetails	An object that stores data related to a key provisioning server error.

2.2.3.1 ErrorDetails

This object contains a collection of human-readable details that describe an error encountered by the key provisioning server. It can be used by the client role of the Key Provisioning Protocol for logging purposes or for providing information to an administrator.

```
{
    "description": "error details",
    "type": "object",
    "properties": {
        "code": {  "type": "string", "optional": false },
        "message": { "type": "string", "optional": false },
        "response": {  "type": "string", "optional": false },
```

```
            "target": { "type": "string", "optional": false },
            "clientrequestid": {  "type": "string", "optional": true },
            "time": { "type": "string", "optional": false },
            "innererror": {
                "description": "error details",
                "type": "object",
                "optional": true,
                "properties": {
                    "trace": {  "type": "string", "optional": false },
                    "context": {  "type": "string", "optional": false },
                }
            }
        }
    }
```

code: A server-generated string representing a machine readable error.

message: A human-readable string explaining the error.

response: The client action to be taken when this error is received. This value MUST be set to "ERROR_FAIL".

target: A string representing the resource being acted upon.

clientrequestid: A GUID in standard string representation (see [RFC4122] section 3 for the format).

time: The [ISO8601]-formatted time that is assigned by the key provisioning server.

trace: MUST be "null".

context: MUST be "null".

2.3 Directory Service Schema Elements

This protocol makes use of the Directory Service schema classes and attributes that are listed in the following table.

For the syntax of <Class> or <Class><Attribute> pairs, refer to [MS-ADA2], [MS-ADA3], and [MS-ADSC].

Class	Attribute
User	ms-DS-Key-Credential-Link, User-Principal-Name
ms-DS-Device	ms-DS-Device-ID
ms-DS-Device-Registration-Service-Container	
ms-DS-Device-Registration-Service	ms-DS-Issuer-Certificates, ms-DS-Issuer-Public-Certificates

2.3.1 ms-DS-Issuer-Certificates

The ms-DS-Issuer-Certificates attribute is a multivalued OCTET_STRING attribute (see [MS-ADTS] section 3.1.1.2.2.2, the String(Octet) syntax). Each value of the attribute is stored as a binary blob containing the following formatted data:

[time]:[binary value of an X.509 certificate]

where [time] is the timestamp formatted as an integer representing the number of 100-nanosecond intervals that have elapsed since 12:00:00 midnight, January 1, 0001, UTC, and [binary value of an X.509 certificate] is the contents of an X.509 certificate [RFC5280] stored as an encrypted binary blob.

2.3.2 ms-DS-Issuer-Public-Certificates

The ms-DS-Issuer-Public-Certificates attribute is a multivalued OCTET_STRING attribute (see [MS-ADTS] section 3.1.1.2.2.2, the String(Octet) syntax). Each value of the attribute is stored as a binary blob containing an X.509 certificate [RFC5280].

2.3.3 ms-DS-Key-Credential-Link

The ms-DS-Key-Credential-Link attribute is a multivalued DN-Binary attribute (see [MS-ADTS] section 3.1.1.2.2.2, the Object(DN-Binary) syntax). Each value is formatted as follows:

B:[keylen]:[key]:[objectDN]

Where [keylen] is the length of [key]. [key] is a KEYCREDENTIALLINK_BLOB ([MS-ADTS] section 2.2.20.2). [objectDN] is an [RFC4514]-formatted distinguished name for the directory object that contains the ms-DS-Key-Credential-Link attribute.

3 Protocol Details

3.1 Key Provisioning Server Details

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The following resource is used by the Key Provisioning Protocol.

Resource	Description
key	An object that represents a key bound to a user and device pair.

3.1.5.1 Key

The following HTTP method is allowed to be performed on this resource:

HTTP Method	Description
POST	Provisions the supplied key on a User object.

3.1.5.1.1 POST

This method provisions a key on a User object.

The key resource is identified by the following URI:

```
https://server/EnrollmentServer/key?api-version=1.0 HTTP/1.1
```

The URI parameters supported by the POST request are the common URI parameters documented in section 2.2.2.

3.1.5.1.1.1 Request Body

The request body contains the following JSON-formatted object.

```
{
    "description": "key provisioning request object",
    "type": "object",
    "properties": {
        "kngc": {  "type": "string", "optional": false }
    }
}
```

kngc: Contains the base64-encoded public portion of an asymmetric key.

3.1.5.1.1.2 Response Body

If the key provisioning server successfully creates and provisions a key in the directory, the HTTP 200 status code is returned, along with a server-generated GUID in the request-id header. Additionally, the response body contains the following JSON-formatted object. See section 3.1.5.1.1.3 for processing details.

```
{
    "description": "key provisioning response object",
    "type": "object",
    "properties": {
        "kid": {  "type": "string", "optional": false },
        "upn": { "type": "string", "optional": false },
        "pctx" { "type": "string", "optional": true }
    }
}
```

kid: Contains a unique key identifier.

upn: Contains the User-Principal-Name ([MS-ADA3] section 2.349) of the User object onto which the key was written.

pctx: A base64-encoded, signed Cryptographic Message Syntax (CMS) message ([RFC5652] section 5) that contains the following JSON-formatted object with information about the Active Directory server that the key was registered with. The value MUST be opaque to the client.<1>

```
{
    "description": "Active Directory server information",
    "type": "object",
    "properties": {
        "DomainControllerFqdn": { "type": "string", "optional": false }
    }
}
```

DomainControllerFqdn: Contains the Domain Name System (DNS) address of the domain controller (DC) that the key was registered with.

3.1.5.1.1.3 Processing Details

The HTTP POST request is processed as follows.

1. The key provisioning server verifies that the following elements are included in the POST request.

    1. The api-version parameter MUST be present and contain the value "1.0".

2. The accept HTTP header MUST be present and contain the value "application/json".

3. The kngc property (section 3.1.5.1.1.1) MUST be present and base64-encoded.

If any of these constraints are not met, the server MUST respond to the HTTP POST with the HTTP status code set to 400 and the body of the response MUST contain an ErrorDetails object populated according to section 2.2.3.1.

2. The key provisioning server verifies that the authorization HTTP header is present and contains a JSON Web Token with the following claims:

Claim	Value
deviceid	The value MUST contain the ms-DS-Device-ID ([MS-ADA2] section 2.295) of an ms-DS-Device object in the Directory.
upn	The value MUST contain the User-Principal-Name of a user in the Directory.
amr	The value MUST contain "ngcmfa", "mfa", or "http://schemas.microsoft.com/claims/multipleauthn".<2>

If any of these constraints are not met, the server MUST respond to the HTTP POST with the HTTP status code set to 401, and the body of the response MUST contain an ErrorDetails object populated according to section 2.2.3.1.

3. The key provisioning server locates the User object in Active Directory where the User-Principal-Name attribute on the User object matches the upn claim in the JWT from step 2. If a User object is NOT found in the directory, the server MUST respond with an HTTP response with the HTTP status code set to 400. The body of the response MUST contain an ErrorDetails object populated according to section 2.2.3.1.

4. The key provisioning server sends a request to the directory server, which adds an ms-DS-Key-Credential-Link object as an additional value in the ms-DS-Key-Credential-Link attribute on the User object located in step 3. If the directory request cannot be successfully completed, the server MUST respond with an HTTP response with the HTTP status code set to 400. The body of the response MUST contain an ErrorDetails object populated according to section 2.2.3.1.

   The KEYCREDENTIALLINK_BLOB MUST be created according to [MS-ADTS] section 2.2.20 and section 2.3.3 in this document. In addition, the following KEYCREDENTIALLINK_ENTRY identifiers ([MS-ADTS] section 2.2.20.6) MUST be present.

KEYCREDENTIALLINK_ENTRY Identifier	Value
KeyMaterial	The base64-decoded value of the kngc property.
KeyUsage	KEY_USAGE_NGC ([MS-ADTS] section 2.2.20.1)
CustomKeyInformation	The **Version** field MUST be set to 1 and the **Flags** MUST be set to 0x02.
KeySource	KEY_SOURCE_AD ([MS-ADTS] section 2.2.20.1)
DeviceId	MUST be the value of the deviceid claim in the JWT from step 2, formatted according to [MS-ADTS] section 2.2.20.
KeyApproximateLastLogonTimeStamp	MUST be set to a time generated by the key provisioning server, represented in FILETIME format ([MS-DTYP] section 2.3.3).

KEYCREDENTIALLINK_ENTRY Identifier	Value
KeyCreationTime	MUST be set to a time generated by the key provisioning server, represented in FILETIME format.

5. The key provisioning server SHOULD provide a value in the **pctx** property of the response (section 3.1.5.1.1.2), which is created as follows.

    1. The key provisioning server creates an Active Directory server-information JSON-formatted object, populating the **DomainControllerFqdn** property with the DNS address of the DC that wrote the ms-DS-Key-Credential-Link attribute in step 4.

    2. The server creates a signed CMS message ([RFC5652] section 5), setting the message content to the UTF-8 encoding of the previously constructed JSON object. The server signs the CMS message by using the issuer certificate stored in the ms-DS-Issuer-Certificates attribute of the ms-DS-Device-Registration-Service object with the most recent timestamp (see section 2.3.1). The server MUST use a SHA256WithRSAEncryption signature algorithm ([RFC5754] section 3.2) and a SHA256 hash algorithm ([FIPS180-2] section 6.2.2).

    3. The **pctx** property of the response is set to the base64-encoding of the resulting CMS message.

6. The key provisioning server responds to the HTTP POST request with an HTTP response with the HTTP status code set to 200 ("OK") and a server-generated GUID in the request-id header. The response body MUST contain the UPN of the User object located in step 3 along with a server-generated GUID for the key identifier.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Key Provisioning Client Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

Note that the notation (Public) indicates that the element can be directly accessed from outside this protocol.

The following elements are defined by this protocol:

Data Store Information (Public): A string that is optionally returned from the server during key provisioning. It contains information about the data store that the key was registered with.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

The resources used for the Key Provisioning Protocol are defined in section 3.1.5.

3.2.5.1 Key

The HTTP methods allowed for this resource are defined in section 3.1.5.1.

3.2.5.1.1 POST

The POST message, including request and response body information, is defined in section 3.1.5.1.1 and subsections.

When the client receives the response from the POST message, it SHOULD<3> check for the **pctx** property. If this property exists in the response, the client stores its value in the **Data Store Information** abstract data model element (section 3.2.1).

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

The following section shows an example of the requests and responses that are defined by the Key Provisioning Protocol.

Note Throughout these examples, line breaks were added and irrelevant fields were removed to enhance readability.

4.1 Provision a Key

The following example shows a request to the key provisioning server to provision a key (section 3.1.5.1.1.1) and the response (section 3.1.5.1.1.2).

Request:

```
POST https://www.contoso.com/EnrollmentServer/key/?api-version=1.0 HTTP/1.1
Accept: application/json
Authorization: Bearer
eyJEZXZpY2VJRCI6IjNhNWY0NzQzLWQ0NTItNDQ2YS05NWY2LTRkYjFhNTZiOTJjYSIsInVwbiI6InVzZXJAY29udG9zb
y5jb20ifQ==
return-client-request-id: true
client-Request-Id: 006dd572-ca07-42ae-8472-01a00b045bb8
{
    "kngc":"VGhpc0lzQW5FeGFtcGxlQXN5bW1ldHJpY0tleQ=="
}
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
client-request-id: 006dd572-ca07-42ae-8472-01a00b045bb8
request-id: 650206f2-9a02-447d-9347-0cb7b4fee827

{
    "kid":"0ce67455-8ea1-4523-be5f-bfd09190fad6",
    "upn":"user@contoso.com",
    "pctx":"eyJEZXZpY2VJRCI6IjNhNWY0NzQzLWQ0NTIt=="
}
```

5 Security

5.1 Security Considerations for Implementers

The Key Provisioning Protocol uses HTTPS as a transport. Using Secure Sockets Layer (SSL) server certificate verification ensures that the client is communicating with the real key provisioning server and closes any possible man-in-the-middle attacks.

The input message uses an JSON Web Token for both authentication and authorization. The key provisioning server must validate that the security token is signed by a trusted identity provider, is within the token validity period, and that the target audience of the token is the server.

5.2 Index of Security Parameters

Security Parameter	Section
kngc	3.1.5.1.1.1
Authorization protocol	3.1.5.1.1.3

6 Appendix A: Full JSON Schema

```
{
    "description": "error details",
    "type": "object",
    "properties": {
        "code": {  "type": "string", "optional": false },
        "message": { "type": "string", "optional": false },
        "response": {  "type": "string", "optional": false },
        "target": { "type": "string", "optional": false },
        "clientrequestid": {  "type": "string", "optional": true },
        "time": { "type": "string", "optional": false },
        "innererror": {
            "description": "error details",
            "type": "object",
            "optional": true,
            "properties": {
                "trace": {  "type": "string", "optional": false },
                "context": {  "type": "string", "optional": false },
            }
        }
}

{
    "description": "key provisioning request object",
    "type": "object",
    "properties": {
        "kngc": {  "type": "string", "optional": false }
    }
}

{
     "description": "key provisioning response object",
     "type": "object",
     "properties": {
         "kid": {  "type": "string", "optional": false },
         "upn": { "type": "string", "optional": false },
         "pctx" { "type": "string", "optional": true }
     }
}

{
     "description": "Active Directory server information",
     "type": "object",
     "properties": {
         "DomainControllerFqdn": { "type": "string", "optional": false }
     }
}
```

7 (Updated Section) Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

The terms "earlier" and "later", when used with a product version, refer to either all preceding versions or all subsequent versions, respectively. The term "through" refers to the inclusive range of versions. Applicable Microsoft products are listed chronologically in this section.

The following tables show the relationships between Microsoft product versions or supplemental software and the roles they perform.

Windows Client releases	Client role	Key Provisioning Server role
Windows 10 v1607 operating system	Yes	No

Windows Server releases	Client role	Key Provisioning Server role
Windows Server 2016 operating system	Yes	Yes
Windows Server operating system	~~Yes~~ No	Yes
Windows Server 2019 operating system	Yes	Yes

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

<1> Section 3.1.5.1.1.2: The **pctx** property is not available in Windows Server 2016 unless [MSKB-4022723] is installed.

<2> Section 3.1.5.1.1.3: In Windows Server 2016 without [MSKB-4022723] installed, the value in the amr claim contains either "mfa" or "http://schemas.microsoft.com/claims/multipleauthn". However, in Windows Server 2016 with [MSKB-4022723] installed and in later versions of the product, the value contains "ngcmfa".

<3> Section 3.2.5.1.1: Windows 10 v1607, and also Windows Server 2016 without [MSKB-4022723] installed, ignore the **pctx** property if one is given in the response.

8 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
7 Appendix B: Product Behavior	Added Windows Server 2019 to the list of applicable products.	Major
7 Appendix B: Product Behavior	Updated support information for the protocol's client role.	Major

9 Index

K

M

N

O

P

R

S

T

U

V