Redfish Certificate Management White Paper
DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

Implementation of certain elements of this standard or proposed standard may be subject to third party patent rights, including provisional patent rights (herein “patent rights”). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party's reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

For information about patents held by third-parties which have notified the DMTF that, in their opinion, such patent may relate to or impact implementations of DMTF standards, visit http://www.dmtf.org/about/policies/disclosures.php.

This document's normative language is English. Translation into other languages is permitted.
## CONTENTS

1. Introduction ................................................................................................................................. 6
2. Model explanation .......................................................................................................................... 6
   2.1. Certificate service ................................................................................................................... 6
       2.1.1. Generate CSR ............................................................................................................... 7
       2.1.2. Replace certificate ........................................................................................................ 9
   2.2. Certificates .............................................................................................................................. 10
       2.2.1. Resources with certificates .......................................................................................... 12
   2.3. Certificate locations ................................................................................................................ 12
3. Client workflows ............................................................................................................................ 13
   3.1. Common operations .................................................................................................................. 13
       3.1.1. Generate CSR ............................................................................................................... 13
       3.1.2. Replace a certificate ..................................................................................................... 14
       3.1.3. Replace a certificate and provide a private key ............................................................. 15
       3.1.4. Install a certificate ........................................................................................................ 16
       3.1.5. Install a certificate and provide a private key ............................................................... 16
       3.1.6. Remove a certificate ..................................................................................................... 17
   3.2. Generating and installing a new certificate ............................................................................... 17
   3.3. Installing a new certificate that was not generated by the service ......................................... 18
   3.4. Possible side effects when installing certificates ..................................................................... 19
   3.5. Removing certificates ............................................................................................................. 19
4. Appendix ......................................................................................................................................... 20
   4.1. References .................................................................................................................................. 20
   4.2. Change log .................................................................................................................................. 20
Foreword

The Redfish Certificate Management White Paper was prepared by the Redfish Forum of the DMTF.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see http://www.dmtf.org.
Acknowledgments

The DMTF acknowledges the following individuals for their contributions to this document:

- Chris Davenport - Hewlett Packard Enterprise
- Michael Raineri - Dell Inc.
- Peter Svensson - Ericsson AB
1. Introduction

Redfish is an evolving hardware management standard that is designed to be flexible, extensible, and interoperable. Redfish contains a data model that is used to describe certificates for devices, services, and other resources. It also provides interface for clients to manage their certificates. This document helps implementers and clients understand the Redfish certificate data model as well as the common workflows clients might require to manage certificates.

2. Model explanation

If a Redfish service supports certificate management, the service root resource will contain the CertificateService property. It will also support the Certificates property in various resources to show where certificates can be installed.

Within the CertificateService resource, a client will find the available actions for managing certificates, such as generating signing requests, or doing atomic replacement of certificates. A client will also find the CertificateLocations resource, which is responsible for providing an inventory of all certificates the service is managing.

The following sections detail how these things are reported by a Redfish service.

2.1. Certificate service

The certificate service is represented by the CertificateService resource. It contains a set of actions for service level management of certificates. There are currently two actions defined for this:

- GenerateCSR (Certificate Signing Request)
- ReplaceCertificate

The CertificateService resource also contains a link to the CertificateLocations resource via the CertificateLocations property.

Example CertificateService resource:

```json
{
    "@odata.id": "/redfish/v1/CertificateService",
    "@odata.type": "#CertificateService.v1_0_0.CertificateService",
    "Id": "CertificateService",
    "Name": "Certificate Service",
    ...
}
```
"Actions": {
    "#CertificateService.GenerateCSR": {
        "target": "/redfish/v1/CertificateService/Actions/CertificateService.GenerateCSR"
    },
    "#CertificateService.ReplaceCertificate": {
    }
},
"CertificateLocations": {
    "@odata.id": "/redfish/v1/CertificateService/CertificateLocations"
}

2.1.1. Generate CSR

The GenerateCSR (Certificate Signing Request) action is used to create a signing request for a new certificate. The response of the action contains the CSR as a Privacy Enhanced Mail (PEM) encoded string. The CSR is then given to a CA (Certificate Authority) by the client, which then produces a signed certificate.

The parameters for the action specify many of the common fields that are put into the end certificate. The action also allows for the client to specify the type of key-pair that is generated for the certificate. The private portion of the key-pair is expected to be retained by the service and used with the certificate that is installed after the CA has signed it. It's possible one or more clients will invoke this action multiple times before any certificates are signed and installed. It's recommended that implementations retain sufficient information from each of the requests, such as the CSR string that was generated, so that it's possible to map a given certificate to a CSR that was produced by this action.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>X.509 attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommonName</td>
<td>commonName</td>
<td>The fully qualified domain name of the component that is being secured.</td>
</tr>
<tr>
<td>AlternativeNames</td>
<td>subjectAltName</td>
<td>Additional hostnames of the component that is being secured.</td>
</tr>
<tr>
<td>Organization</td>
<td>organizationName</td>
<td>The name of the organization making the request.</td>
</tr>
<tr>
<td>OrganizationalUnit</td>
<td>organizationalUnitName</td>
<td>The name of the unit or division of the organization making the</td>
</tr>
<tr>
<td>Parameter</td>
<td>X.509 attribute</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>City</td>
<td>localityName</td>
<td>The city or locality of the organization making the request.</td>
</tr>
<tr>
<td>State</td>
<td>stateOrProvinceName</td>
<td>The state, province, or region of the organization making the request.</td>
</tr>
<tr>
<td>Country</td>
<td>countryName</td>
<td>The country of the organization making the request.</td>
</tr>
<tr>
<td>Email</td>
<td>emailAddress</td>
<td>The email address of the contact within the organization making the request.</td>
</tr>
<tr>
<td>KeyUsage</td>
<td>keyUsage and extKeyUsage</td>
<td>The usage of the key contained in the certificate.</td>
</tr>
<tr>
<td>Surname</td>
<td>surname</td>
<td>The surname of the user making the request.</td>
</tr>
<tr>
<td>GivenName</td>
<td>givenName</td>
<td>The given name of the user making the request.</td>
</tr>
<tr>
<td>Initials</td>
<td>initials</td>
<td>The initials of the user making the request.</td>
</tr>
<tr>
<td>ChallengePassword</td>
<td>challengePassword</td>
<td>The challenge password to be applied to the certificate for revocation requests.</td>
</tr>
<tr>
<td>UnstructuredName</td>
<td>unstructuredName</td>
<td>The unstructured name of the subject.</td>
</tr>
<tr>
<td>ContactPerson</td>
<td>name</td>
<td>The name of the user making the request.</td>
</tr>
<tr>
<td>KeyPairAlgorithm</td>
<td>N/A</td>
<td>The type of key-pair for use with signing algorithms.</td>
</tr>
<tr>
<td>KeyBitLength</td>
<td>N/A</td>
<td>The length of the key in bits, if needed based on the value of the KeyPairAlgorithm parameter.</td>
</tr>
<tr>
<td>Parameter</td>
<td>X.509 attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>KeyCurveId</td>
<td>N/A</td>
<td>The curve ID to be used with the key, if needed based on the value of the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KeyPairAlgorithm parameter.</td>
</tr>
<tr>
<td>CertificateCollection</td>
<td>N/A</td>
<td>A link to the CertificateCollection resource where the certificate will be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>installed.</td>
</tr>
</tbody>
</table>

The KeyPairAlgorithm and KeyCurveId are both strings where the values are based on the contents of the Trusted Computing Group (TCG) Algorithm Registry. The TPM_ALG_ID and TPM_ECC_CURVE tables respectively contain the set of possible values for the two parameters. Services are not required to support the entire contents of the table, however, the following are recommended to be supported.

Recommended TPM_ALG_ID values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPM_ALG_RSA</td>
<td>The RSA algorithm.</td>
</tr>
<tr>
<td>TPM_ALG_ECDSA</td>
<td>Signature algorithm using elliptic curve cryptography (ECC).</td>
</tr>
</tbody>
</table>

Recommended TPM_ECC_CURVE values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPM_ECC_NIST_P256</td>
<td>The NIST P-256 curve.</td>
</tr>
<tr>
<td>TPM_ECC_NIST_P384</td>
<td>The NIST P-384 curve.</td>
</tr>
<tr>
<td>TPM_ECC_NIST_P521</td>
<td>The NIST P-521 curve.</td>
</tr>
</tbody>
</table>

A sample usage of this action can be found in the workflows section.

2.1.2. Replace certificate

The ReplaceCertificate action is used for cases where an atomic deletion of a Certificate resource followed by a creation of a new Certificate resource is required. For example, if an HTTPS service has a single certificate and a user would like to replace it, performing a delete operation on the old certificate first would create a window where the HTTPS service does not have a certificate, which the service might not be able to handle.
The request body for the ReplaceCertificate action contains the same fields required when creating a new Certificate resource. CertificateType contains the format of the certificate, and CertificateString contains the string encoding of the certificate based on its format. In addition, the client also provides the URI of the certificate to be replaced using the CertificateUri parameter. The response from the action uses the Location HTTP header to show where the new certificate was placed. Depending on the implementation, the Location header in the response might contain the same URI specified in the CertificateUri parameter, but if they are not the same, it's expected that the URI referenced by the CertificateUri parameter is no longer available at this point.

A sample usage of this action can be found in the workflows section.

2.2. Certificates

Devices, services, and other resources that support certificates have a Certificates property that references the CertificateCollection resource for the respective resource. This is done so that it's possible to associate one or more certificates with a particular usage. For example, the CertificateCollection resource found off of the HTTPS property within the ManagerNetworkProtocol resource is used to represent the certificates for the HTTPS service for a manager. Over time, resources that have use cases for managing certificates will have CertificateCollection resources added to them.

Users are able to add and remove members from the CertificateCollection resource like with any other type of resource collection. A user can submit a POST operation to the resource collection to install a new certificate, and a user can perform a DELETE operation to a Certificate resource to remove a certificate from the resource collection. When removing the last member of the CertificateCollection resource, the service might automatically install a default certificate. This could happen for cases where at least one certificate is always required to be present, such as with an HTTPS service.

Since there are different formats for certificates, the CertificateCollection resource uses the @Redfish.SupportedCertificates annotation to show which formats it will accept. The following example shows a CertificateCollection resource that only supports PEM style certificates.

```json
{
    "@odata.id": "/redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates",
    "@odata.type": "#CertificateCollection.CertificateCollection",
    "Name": "Certificate Collection",
    "Members@odata.count": 1,
    "Members": [
        {
            "@odata.id": "/redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates/1"
        }
    ]
}
```
Within the Certificate resource, there are two properties that are provided by the user when installing a new certificate:

- **CertificateType** describes the format of the certificate.
- **CertificateString** is the string encoding of the certificate based on the value of CertificateType.

If there is ever a portion of the certificate that contains private information, such as a private key, a service is not allowed to provide it when the certificate is being read by a client.

The remaining properties of the Certificate resource contain decodings of attributes found in the certificate itself. An example of this is shown below.

```json
{
    "@odata.id": "/redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates/1",
    "@odata.type": "#Certificate.v1_1_0.Certificate",
    "Id": "1",
    "Name": "HTTPS Certificate",
    "CertificateString": "-----BEGIN CERTIFICATE-----
...\n-----END CERTIFICATE-----",
    "CertificateType": "PEM",
    "Issuer": {
        "Country": "US",
        "State": "Oregon",
        "City": "Portland",
        "Organization": "Contoso",
        "OrganizationalUnit": "ABC",
        "CommonName": "manager.contoso.org"
    },
    "Subject": {
        "Country": "US",
        "State": "Oregon",
        "City": "Portland",
        "Organization": "Contoso",
        "OrganizationalUnit": "ABC",
        "CommonName": "manager.contoso.org"
    },
    "ValidNotBefore": "2018-09-07T13:22:05Z"
}
```
2.2.1. Resources with certificates

As of this publication, the following resources contain a Certificates property, which references their respective CertificateCollection resource:

- AccountService resource, within both the LDAP and ActiveDirectory properties
- ComputerSystem resource, within the Boot property
- ExternalAccountProvider resource
- ManagerAccount resource
- ManagerNetworkProtocol resource, within the HTTPS property

2.3. Certificate locations

The CertificateLocations resource is contains a set of links to all of the certificates managed by the service. This type of resource is to be used for security administrators who may need perform auditing of the service. The following is an example CertificateLocations resource that shows the service is managing three certificates.
3. Client workflows

The following section contains possible workflows that clients may follow for managing certificates.

3.1. Common operations

The following section shows common operations used in various client workflows.

3.1.1. Generate CSR

The following two examples show a client performing a GenerateCSR action and receiving a response. In the request, the client is specifying a new signing request using an RSA 4096bit key-pair, and that the end certificate will be installed for the HTTPS service for the Manager resource named BMC. The response contains the PEM encoded signing request in the property CSRString.

Example generate CSR request:

```
POST /redfish/v1/CertificateService/Actions/CertificateService.GenerateCSR HTTP/1.1
Content-Type: application/json
Content-Length: <computed-length>

{
    "Country": "US",
    "State": "Oregon",
    "City": "Portland",
    "Organization": "Contoso",
    "OrganizationalUnit": "ABC",
    "CommonName": "manager.contoso.org",
    "AlternativeNames": [
        "manager.contoso.com",
        "manager.contoso.us"
    ],
    "Email": "admin@contoso.org",
    "KeyPairAlgorithm": "TPM_ALG_RSA",
    "KeyBitLength": 4096,
    "KeyUsage": [
```
"KeyEncipherment",
"ServerAuthentication"
],
"CertificateCollection": {
  "@odata.id": "/redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates"
}
}

Example generate CSR response:

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: <computed-length>

{
  "CSRString": "-----BEGIN CERTIFICATE REQUEST-----...
          
          -----END CERTIFICATE REQUEST-----",
  "CertificateCollection": {
    "@odata.id": "/redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates"
  }
}

3.1.2. Replace a certificate

The following two examples show a client performing a ReplaceCertificate action and receiving a response. The request specifies that it's replacing the HTTPS certificate with the Id of 1 for the Manager resource called BMC with a new PEM encoded certificate. The Location header in the response shows that the new certificate was assigned the Id of 2.

Example replace certificate request:

POST /redfish/v1/CertificateService/Actions/CertificateService.ReplaceCertificate HTTP/1.1
Content-Type: application/json
Content-Length: <computed-length>

{
  "CertificateUri": {
    "@odata.id": "/redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates/1"
  },
  "CertificateString": "-----BEGIN CERTIFICATE-----
          
          -----END CERTIFICATE-----",
  "CertificateType": "PEM"
Example replace certificate response:

HTTP/1.1 204 No Content
Content-Type: application/json
Content-Length: <computed-length>
Location: /redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates/2

3.1.3. Replace a certificate and provide a private key

There are cases where a private key may need to be provided with the certificate. This could happen if the signing request for the certificate originated from another entity, so the key-pair was not generated by the service. In these cases, the private key can be concatenated with the certificate itself.

The following two examples show a client performing a ReplaceCertificate action and receiving a response. The request specifies that it's replacing the HTTPS certificate with the Id of 1 for the Manager resource called BMC with a new PEM encoded certificate. It's also including a private key portion in CertificateString. The Location header in the response shows that the new certificate was assigned the Id of 2.

Example replace certificate request:

```json
POST /redfish/v1/CertificateService/Actions/CertificateService.ReplaceCertificate HTTP/1.1
Content-Type: application/json
Content-Length: <computed-length>

{
    "CertificateUri": {
        "@odata.id": "/redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates/1"
    },
    "CertificateString": "-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
-----BEGIN PRIVATE KEY-----
...
-----END PRIVATE KEY-----",
    "CertificateType": "PEM"
}
```

Example replace certificate response:

HTTP/1.1 204 No Content
3.1.4. Install a certificate

The following two examples show a client installing a certificate to the CertificateCollection resource located at /redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates and receiving a response. The Location header in the response shows that the new certificate was assigned the Id of 1.

Example certificate installation request:

```plaintext
POST /redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates HTTP/1.1
Content-Type: application/json
Content-Length: <computed-length>

{
   "CertificateString": "-----BEGIN CERTIFICATE-----
   ...
   -----END CERTIFICATE-----",
   "CertificateType": "PEM"
}
```

Example certificate installation response:

```plaintext
HTTP/1.1 204 No Content
Content-Type: application/json
Content-Length: <computed-length>
Location: /redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates/1
```

3.1.5. Install a certificate and provide a private key

There are cases where a private key may need to be provided with the certificate. This could happen if the signing request for the certificate originated from another entity, so the key-pair was not generated by the service. In these cases, the private key can be concatenated with the certificate itself.

The following two examples show a client installing a certificate to the CertificateCollection resource located at /redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates and receiving a response. It’s also including a private key portion in CertificateString. The Location header in the response shows that the new certificate was assigned the Id of 1.

Example certificate installation request:
Example certificate installation response:

HTTP/1.1 204 No Content
Content-Type: application/json
Content-Length: <computed-length>
Location: /redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates/1

3.1.6. Remove a certificate

The following example shows a client removing a certificate located at /redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates/1.

Example certificate removal request:

DELETE /redfish/v1/Managers/BMC/NetworkProtocol/HTTPS/Certificates/1 HTTP/1.1

3.2. Generating and installing a new certificate

In this workflow, a client is using the service to generate a new Certificate Signing Request (CSR), and installing a new certificate on a given service or device.

1. Locate the appropriate CertificateCollection resource where the new certificate will be installed.
   - For example, if installing a new HTTPS certificate for a manager:
     - Identify the appropriate Manager resource in the ManagerCollection resource.
     - Navigate to the manager’s ManagerNetworkProtocol resource.
     - Navigate to the CertificateCollection resource found in the HTTPS object.
2. Perform the GenerateCSR action.
3. The service will generate a new key-pair and retain the private key.

3. The value of CSRString in the response is taken to a Certificate Authority (CA).
   - The CA uses this string to produce a signed certificate.

4. Install the new certificate.
   - If the installation is to replace an existing certificate in the CertificateCollection resource:
     ▪ Perform the ReplaceCertificate action.
     ▪ In the action request, specify the certificate to replace as found in the first step.
   - If the installation is to add a certificate to the CertificateCollection resource:
     ▪ Perform a POST to the CertificateCollection resource from the first step.

### 3.3. Installing a new certificate that was not generated by the service

In this workflow, a client is installing a new certificate that has been generated by an external source. This could be a common certificate provided by a client's IT group that needs to be installed on multiple systems. Depending on how the service will be using the certificate, the private key may need to be provided as part of the installation.

1. Locate the appropriate CertificateCollection resource where the new certificate will be installed.
   - For example, if installing a new HTTPS certificate for a manager:
     ▪ Identify the appropriate Manager resource in the ManagerCollection resource.
     ▪ Navigate to the manager's ManagerNetworkProtocol resource.
     ▪ Navigate to the CertificateCollection resource found in the HTTPS object.

2. Install the new certificate.
   - If the installation is to replace an existing certificate in the CertificateCollection resource, and the service does not require the use of the private key:
     ▪ Perform the ReplaceCertificate action.
     ▪ In the action request, specify the certificate to replace as found in the first step.
   - If the installation is to replace an existing certificate in the CertificateCollection resource, and the service does require the use of the private key:
     ▪ Perform the ReplaceCertificate action.
     ▪ In the action request, specify the certificate to replace as found in the first step.
     ▪ Also in the action request, the private key of the certificate will need to be provided.
   - If the installation is to add a certificate to the CertificateCollection resource, and the service does not require the use of the private key:
     ▪ Perform a POST to the CertificateCollection resource from the first step.
3.4. Possible side effects when installing certificates

In some cases, multiple CertificateCollection resources may share a common set of certificates internal to the service. For example, a service might implement a common set of certificates for LDAP and Active Directory. This would make it so that when one is disabled and the other is enabled, the same certificates already installed previously would still be available. In these cases, installing a certificate in one CertificateCollection resource will result in the Certificate resources appearing in two different collections, and deleting a Certificate resource found in one collection will cause it to be removed from both collections. This does not mean that two certificates are tracked in the service, but rather one certificate can be found in two different collections.

3.5. Removing certificates

In this workflow, a client is removing a certificate from the service. This could be done as part of a decommissioning process for a system where certificates need to be removed first, or other types of maintenance activities.

1. Locate the appropriate Certificate resource that will be removed.
   - For example, if removing a certificate for a ManagerAccount resource:
     - Identify the appropriate ManagerAccount resource in the ManagerAccountCollection resource.
     - Navigate to the CertificateCollection resource found in the ManagerAccount resource.
     - Identify the Certificate resource to remove in the CertificateCollection resource.


Removing the last Certificate resource from a CertificateCollection resource might result in different outcomes, depending on the design of the service, or the type of certificate being removed.

- In cases where an empty collection is not valid, such as with an HTTPS service:
  - The operation might fail, which would require steps to be taken to replace the certificate with a default certificate.
  - The operation might result in a new default certificate to populate the collection.
- In other cases, the collection will become empty.
4. Appendix

4.1. References


4.2. Change log

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0</td>
<td>2020-06-11</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>